Banking_Domain(Credit Card Approval Prediction)

Credit Card Approval Prediction
Abstract:
Banks receive a lot of applications for issuance of credit cards. Many of them are rejected for many reasons, like high-loan balances, low-income levels, or too many inquiries on an individual’s credit report. Manually analyzing these applications is error-prone and a time consuming process. This task can be automated with the power of machine learning. In this project, we will build an automatic credit card approval predictor using machine learning techniques, just like the real banks do.

Problem Statement:
The task is to analyze and build a predictive model that can accurately determine whether a credit card application should be approved or rejected based on various applicant attributes and historical credit data. The goal is to develop a model that can assist financial institutions in making informed decisions while minimizing the risk of default and maximizing profitability.

They have given a problem to identify the customers segments which are eligible for Credit Card approval, so that they can specifically target these customers.

Variable Description:

File - Application Record.csv

Column
Description
ID
Unique Id of the row
CODE_GENDER
Gender of the applicant. M is male and F is female.
FLAG_OWN_CAR
Is an applicant with a car? Y is Yes and N is NO.
FLAG_OWN_REALTY
Is an applicant with realty? Y is Yes and N is No.
CNT_CHILDREN
Count of children.
AMT_INCOME_TOTAL
the amount of the income.
NAME_INCOME_TYPE
The type of income (5 types in total).
NAME_EDUCATION_TYPE
The type of education (5 types in total).
NAME_FAMILY_STATUS
The type of family status (6 types in total).

DAYS_BIRTH
The number of the days from birth (Negative values).
DAYS_EMPLOYED
The number of the days from employment (Negative values). This column has error values.
FLAG_MOBIL
Is an applicant with a mobile? 1 is True and 0 is False
FLAG_WORK_PHONE
Is an applicant with a work phone? 1 is True and 0 is False.
FLAG_PHONE
Is an applicant with a phone? 1 is True and 0 is False.

FLAG_EMAIL
Is an applicant with an email? 1 is True and 0 is False.

OCCUPATION_TYPE
The type of occupation (19 types in total). This column has missing values.
CNT_FAM_MEMBERS
The count of family members.

File - Credit Record.csv

ID
Unique Id of the row in application record.

MONTHS_BALANCE
The number of months from record time.
STATUS
Credit status for this month.
X: No loan for the month

C: paid off that month
0: 1-29 days past due
1: 30-59 days past due
2: 60-89 days overdue
3: 90-119 days overdue
4: 120-149 days overdue
5: Overdue or bad debts, write-offs for more than 150 days

Note –
DAYS_BIRTH ---> Count backwards from current day (0), -1 means yesterday.
DAYS_EMPLOYED ---> Count backwards from current day (0). If positive, it means the person currently unemployed.
MONTHS_BALANCE ---> The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the previous month, and so on.
STATUS ---> 0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5: Overdue or bad debts, writeoffs for more than 150 days C: paid off that month X: No loan for the month

Scope:
Understand data by performing exploratory data analysis
Training and building classification algorithm to predict if a customer will be approved with credit card or not
Understand feature importance and improve the model
Understand various model performance metrics and measure the performance of each model

Objective & Learning Outcome:
The objective is to train a machine learning model using the provided dataset to predict the approval outcome for new credit card applications accurately. The model should generalize well to unseen data and provide a reliable assessment of the creditworthiness of applicants.
Students should be able to predict credit card approval from records with the help of a classification model. They should also be able to perform EDA and re-build the model and check if there is any significant change in the predictive scores.

In [1]:
```
import os, sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
plt.style.use('classic')
import warnings
warnings.filterwarnings('ignore')

import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
import plotly.offline as pyo
from plotly import tools
from plotly.subplots import make_subplots
```

In [2]:
```
data = pd.read_csv('application_record.csv')
data.head()
```

Out[2]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUC
0	5008804	M	Y	Y	0	427500.0	Working	Hig
1	5008805	M	Y	Y	0	427500.0	Working	Hig
2	5008806	M	Y	Y	0	112500.0	Working	Seconda
3	5008808	F	N	Y	0	270000.0	Commercial associate	Seconda
4	5008809	F	N	Y	0	270000.0	Commercial associate	Seconda

In [3]:
```
print("No of datapoints for application records : {}".format(len(data)))
print("No. of unique clients in dataset : {}".format(len(data.ID.unique())))
```

```
No of datapoints for application records : 438557
No. of unique clients in dataset : 438510
```

In [4]:
```
records = pd.read_csv('credit_record.csv')
records.head()
```

Out[4]:

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

In [5]:
```
print("No of datapoints for credit records : {}".format(len(records)))
print("No. of unique clients in dataset : {}".format(len(records.ID.unique())))
```

```
No of datapoints for credit records : 1048575
No. of unique clients in dataset : 45985
```

In [6]:
```
len(set(records['ID']).intersection(set(data['ID'])))
```

Out[6]: 36457

In [7]:
```
print(f'Duplicates in applications data : {data.iloc[:,1:].duplicated().sum()},({np.round(100*data.iloc[:,1:].duplic
```

```
Duplicates in applications data : 348472,(79.5%)
```

In [8]:
```
print(f'Duplicates in records data: {records.duplicated().sum()},({np.round(100*records.duplicated().sum()/len(recor
```

```
Duplicates in records data: 0,(0.0%)
```

Drop duplicates

In [9]:
```
data = data.drop_duplicates(subset=data.columns[1:], keep='first')
```

In [10]:
```
data.shape
```

Out[10]: (90085, 18)

In [11]:
```
len(set(records['ID']).intersection(set(data['ID'])))
```

Out[11]: 9709

In [12]:
```
unique_counts = pd.DataFrame.from_records([(col, data[col].nunique()) for col in data.columns],
                                          columns=['Column_name','Num_unique']).sort_values(by=['Num_unique'])
```

In [13]:
```
unique_counts
```

Out[13]:

	Column_name	Num_unique
12	FLAG_MOBIL	1
1	CODE_GENDER	2
2	FLAG_OWN_CAR	2
3	FLAG_OWN_REALTY	2
15	FLAG_EMAIL	2
14	FLAG_PHONE	2
13	FLAG_WORK_PHONE	2
8	NAME_FAMILY_STATUS	5
6	NAME_INCOME_TYPE	5
7	NAME_EDUCATION_TYPE	5
9	NAME_HOUSING_TYPE	6
4	CNT_CHILDREN	12
17	CNT_FAM_MEMBERS	13
16	OCCUPATION_TYPE	18
5	AMT_INCOME_TOTAL	866
11	DAYS_EMPLOYED	9406
10	DAYS_BIRTH	16379
0	ID	90085

In [14]:
```
unique_counts = pd.DataFrame.from_records([(col, records[col].nunique()) for col in records.columns],
                                          columns=['Column_name','Num_unique']).sort_values(by=['Num_unique'])
```

In [15]:
```
unique_counts
```

Out[15]:

	Column_name	Num_unique
2	STATUS	8
1	MONTHS_BALANCE	61
0	ID	45985

In [16]:
```
data.drop(['FLAG_MOBIL'], axis=1, inplace=True)
```

In [17]:
```
data.shape
```

Out[17]:
```
(90085, 17)
```

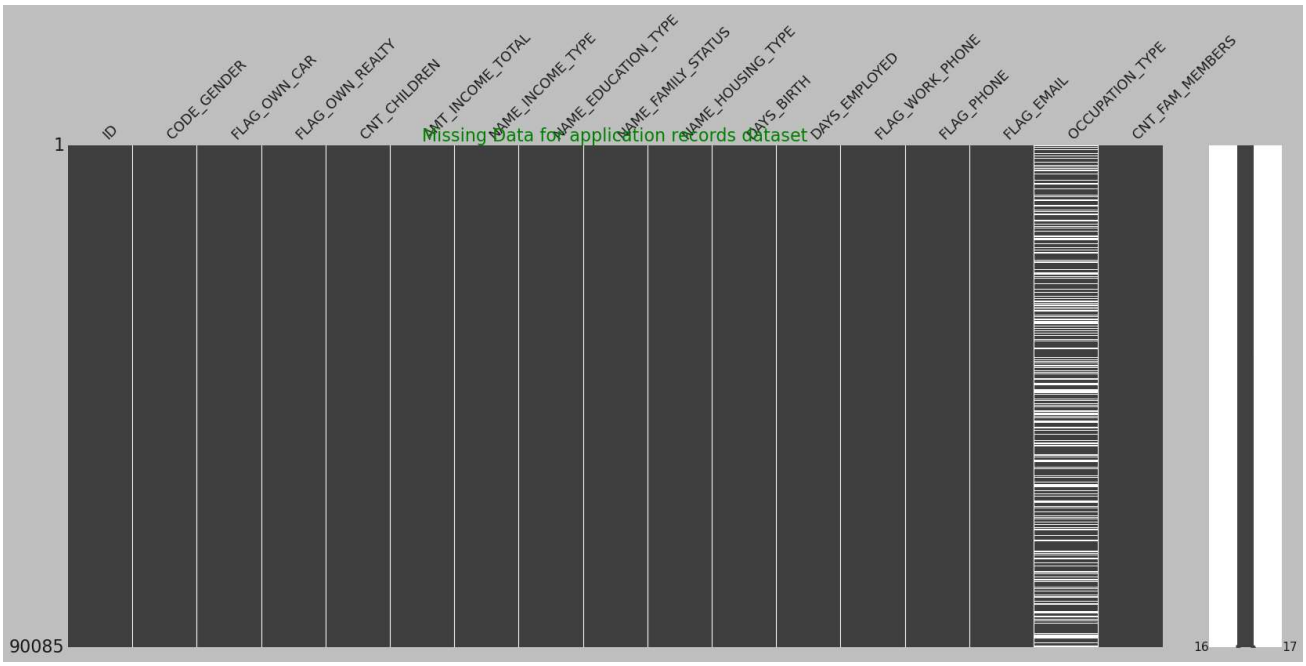
Handling missing values

In [18]:
```
data.isnull().sum()
```

Out[18]:
```
ID                        0
CODE_GENDER               0
FLAG_OWN_CAR              0
FLAG_OWN_REALTY           0
CNT_CHILDREN              0
AMT_INCOME_TOTAL          0
NAME_INCOME_TYPE          0
NAME_EDUCATION_TYPE       0
NAME_FAMILY_STATUS        0
NAME_HOUSING_TYPE         0
DAYS_BIRTH                0
DAYS_EMPLOYED             0
FLAG_WORK_PHONE           0
FLAG_PHONE                0
FLAG_EMAIL                0
OCCUPATION_TYPE       27477
CNT_FAM_MEMBERS           0
dtype: int64
```

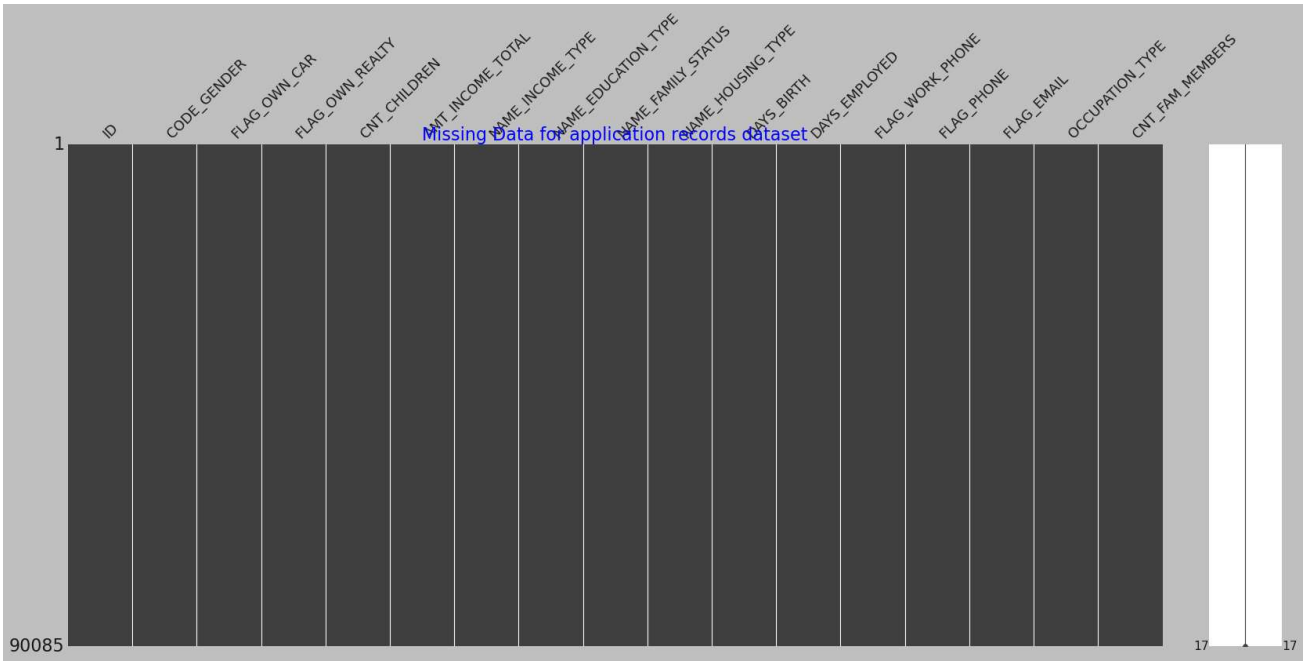
In [19]:
```
import missingno as msno
```

In [20]:
```
missing = msno.matrix(data)
missing.set_title("Missing Data for application records dataset", fontsize=20,color = 'Green')
plt.show()
```



In [21]:
```
data['OCCUPATION_TYPE'].fillna(value='Other', inplace=True)
```

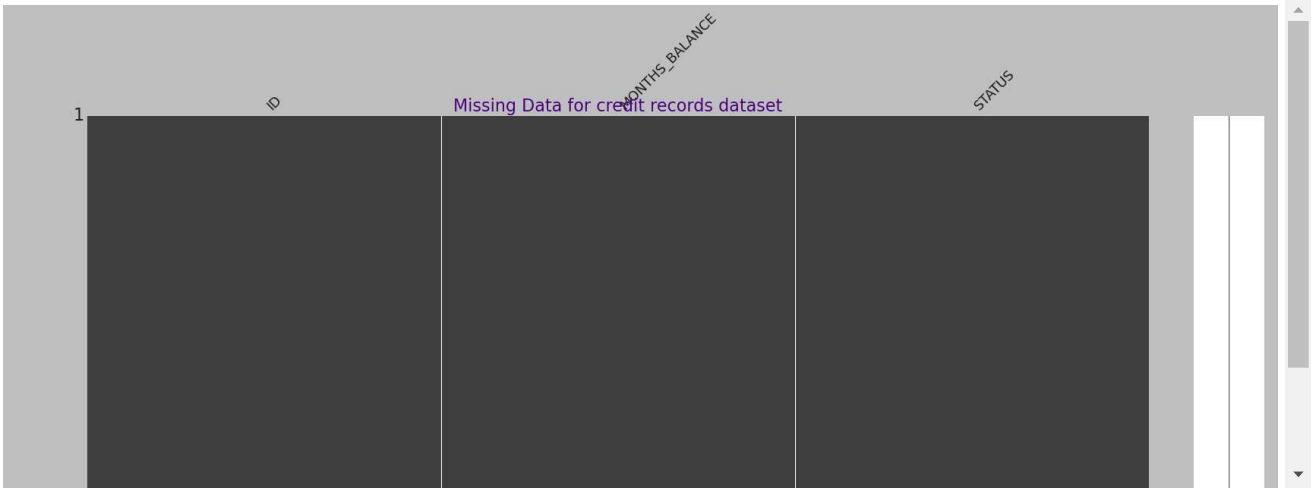
In [22]:
```
missing = msno.matrix(data)
missing.set_title("Missing Data for application records dataset", fontsize=20,color = 'blue')
plt.show()
```



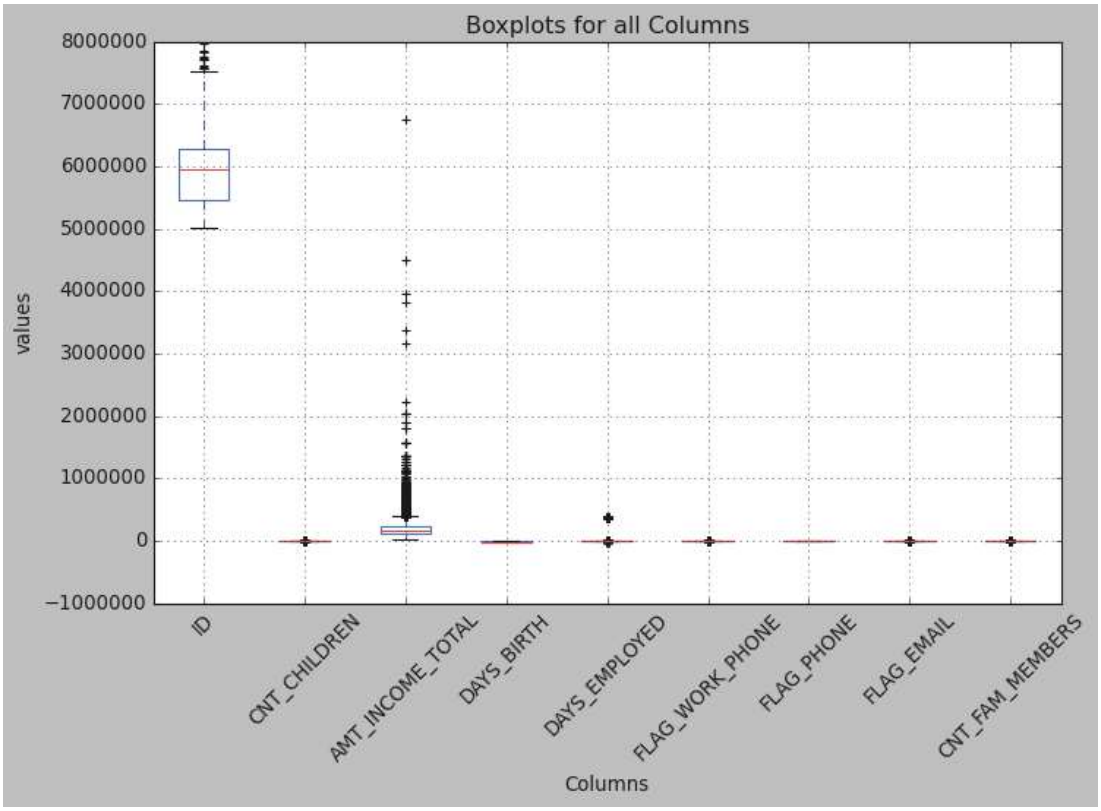
In [23]:
```
records.isnull().sum()
```

Out[23]:
```
ID              0
MONTHS_BALANCE  0
STATUS          0
dtype: int64
```

In [24]:
```
missing2 = msno.matrix(records)
missing2.set_title("Missing Data for credit records dataset", fontsize=20,color = 'indigo')
plt.show()
```



In [25]:
```
num_col = data.select_dtypes(exclude='object').columns.tolist()
plt.figure(figsize = (10,6))
data.boxplot(column = num_col)
plt.title('Boxplots for all Columns')
plt.xticks(rotation = 45)
plt.xlabel('Columns')
plt.ylabel('values')
plt.grid(True)
plt.show()
```



In [ ]:

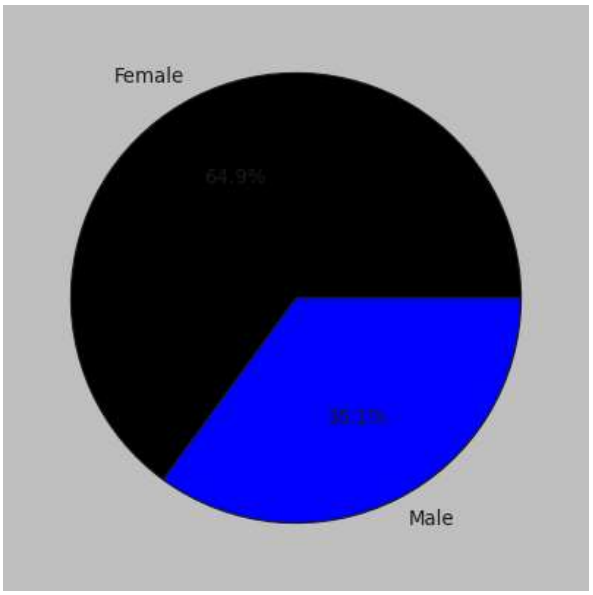
In [26]:
```
for i in data.columns:
    print("**************************",i,
          "**************************")
    print()
    print(set(data[i].tolist()))
    print()
```

```
9, -13628, -13627, -13626, -13625, -13624, -13623, -13622, -13621, -13620, -13619, -13618, -13617, -13616, -1361
5, -13614, -13613, -13612, -13611, -13610, -13609, -13608, -13607, -13606, -13605, -13604, -13603, -13602, -1360
1, -13600, -13599, -13598, -13597, -13596, -13595, -13594, -13593, -13592, -13591, -13590, -13589, -13588, -1358
7, -13586, -13585, -13584, -13583, -13582, -13581, -13580, -13579, -13578, -13577, -13576, -13575, -13574, -1357
3, -13572, -13571, -13570, -13569, -13568, -13567, -13566, -13565, -13564, -13563, -13562, -13561, -13560, -1355
9, -13558, -13557, -13556, -13555, -13554, -13553, -13552, -13551, -13550, -13549, -13548, -13547, -13546, -1354
5, -13544, -13543, -13542, -13541, -13540, -13539, -13538, -13537, -13536, -13535, -13534, -13533, -13532, -1353
1, -13530, -13529, -13528, -13527, -13526, -13525, -13524, -13523, -13522, -13521, -13520, -13519, -13518, -1351
7, -13516, -13515, -13514, -13513, -13512, -13511, -13510, -13509, -13508, -13507, -13506, -13505, -13504, -1350
3, -13502, -13501, -13500, -13499, -13498, -13497, -13496, -13495, -13494, -13493, -13492, -13491, -13490, -1348
9, -13488, -13487, -13486, -13485, -13484, -13483, -13482, -13481, -13480, -13479, -13478, -13477, -13476, -1347
5, -13474, -13473, -13472, -13471, -13470, -13469, -13468, -13467, -13466, -13465, -13464, -13463, -13462, -1346
1, -13460, -13459, -13458, -13457, -13456, -13455, -13454, -13453, -13452, -13451, -13450, -13449, -13448, -1344
7, -13446, -13445, -13444, -13443, -13442, -13441, -13440, -13439, -13438, -13437, -13436, -13435, -13434, -1343
3, -13432, -13431, -13430, -13429, -13428, -13427, -13426, -13425, -13424, -13423, -13422, -13421, -13420, -1341
9, -13418, -13417, -13416, -13415, -13414, -13413, -13412, -13411, -13410, -13409, -13408, -13407, -13406, -1340
5, -13404, -13403, -13402, -13401, -13400, -13399, -13398, -13397, -13396, -13395, -13394, -13393, -13392, -1339
1, -13390, -13389, -13388, -13387, -13386, -13385, -13384, -13383, -13382, -13381, -13380, -13379, -13378, -1337
7, -13376, -13375, -13374, -13373, -13372, -13371, -13370, -13369, -13368, -13367, -13366, -13365, -13364, -1336
3, -13362, -13361, -13360, -13359, -13358, -13357, -13356, -13355, -13354, -13353, -13352, -13351, -13350, -1334
```

In [27]:
```
data['CODE_GENDER'].value_counts()
```

Out[27]:
```
F    58509
M    31576
Name: CODE_GENDER, dtype: int64
```

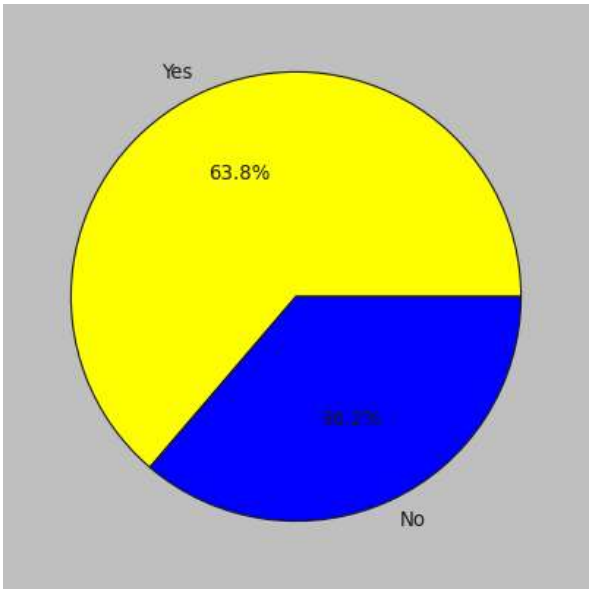
In [28]:
```
CODE_GENDER = ['Female','Male']
quantity = [58509,31576]
plt.pie(quantity,labels=CODE_GENDER,autopct='%0.1f%%',colors=['black','blue'])
plt.show()
```



In [29]:
```
data['FLAG_OWN_CAR'].value_counts()
```

Out[29]:
```
N    57430
Y    32655
Name: FLAG_OWN_CAR, dtype: int64
```

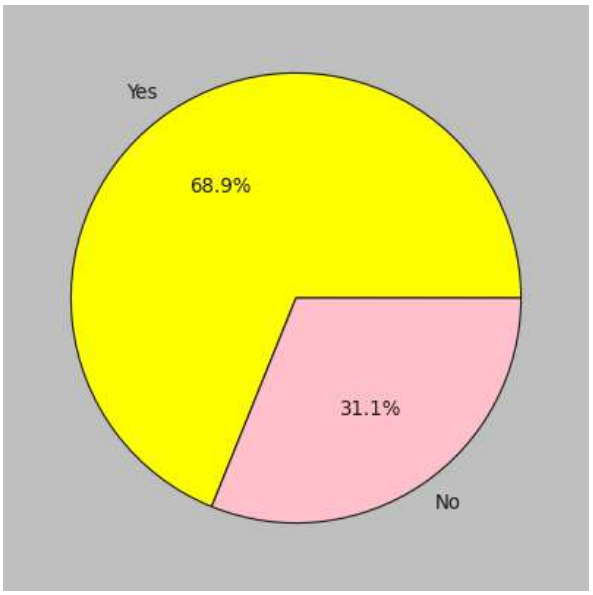
In [30]:
```
FLAG_OWN_CAR = ['Yes','No']
quantity = [57430, 32655]
plt.pie(quantity,labels=FLAG_OWN_CAR,autopct='%0.1f%%',colors=['yellow','blue'])
plt.show()
```



In [31]:
```
data['FLAG_OWN_REALTY'].value_counts()
```

Out[31]:
```
Y    62056
N    28029
Name: FLAG_OWN_REALTY, dtype: int64
```

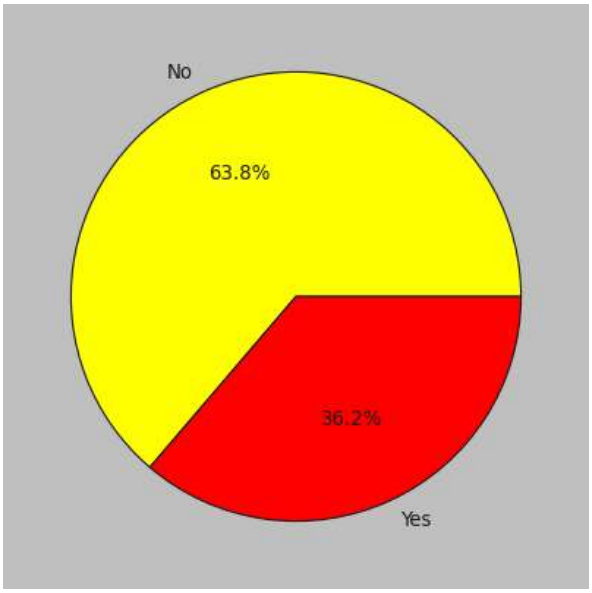
In [32]:
```
FLAG_OWN_REALTY = ['Yes','No']
quantity = [ 62056, 28029]
plt.pie(quantity,labels=FLAG_OWN_REALTY,autopct='%0.1f%%',colors=['yellow','pink'])
plt.show()
```



In [33]:
```
data['FLAG_OWN_CAR'].value_counts()
```

Out[33]:
```
N    57430
Y    32655
Name: FLAG_OWN_CAR, dtype: int64
```

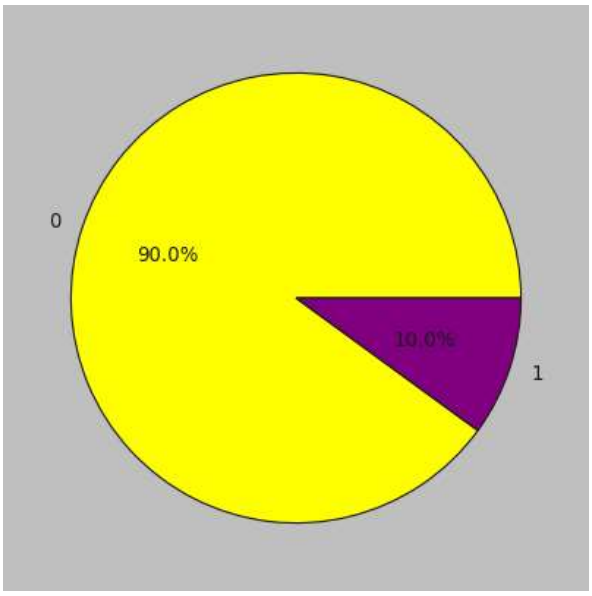
In [34]:
```
FLAG_OWN_CAR = ['No','Yes']
quantity = [57430, 32655]
plt.pie(quantity,labels=FLAG_OWN_CAR,autopct='%0.1f%%',colors=['yellow','red'])
plt.show()
```



In [35]:
```
data['FLAG_EMAIL'].value_counts()
```

Out[35]:
```
0    81056
1     9029
Name: FLAG_EMAIL, dtype: int64
```

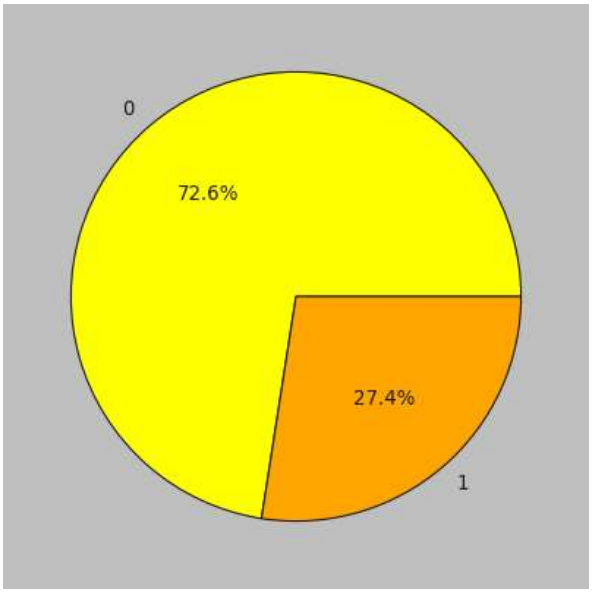
In [36]:
```
FLAG_EMAIL = ['0','1']
quantity = [81056, 9029]
plt.pie(quantity,labels=FLAG_EMAIL,autopct='%0.1f%%',colors=['yellow','purple'])
plt.show()
```



In [37]:
```
data['FLAG_PHONE'].value_counts()
```

Out[37]:
```
0    65357
1    24728
Name: FLAG_PHONE, dtype: int64
```

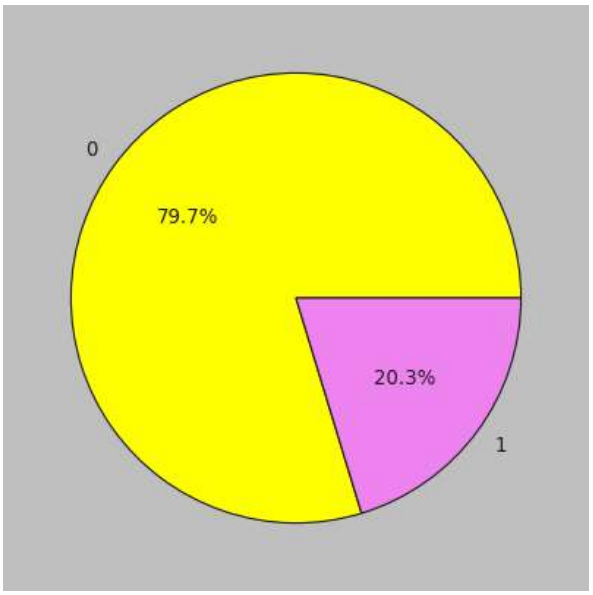
In [38]:
```
FLAG_PHONE = ['0','1']
quantity = [65357,  24728]
plt.pie(quantity,labels=FLAG_PHONE,autopct='%0.1f%%',colors=['yellow','orange'])
plt.show()
```



In [39]:
```
data['FLAG_WORK_PHONE'].value_counts()
```

Out[39]:
```
0    71802
1    18283
Name: FLAG_WORK_PHONE, dtype: int64
```

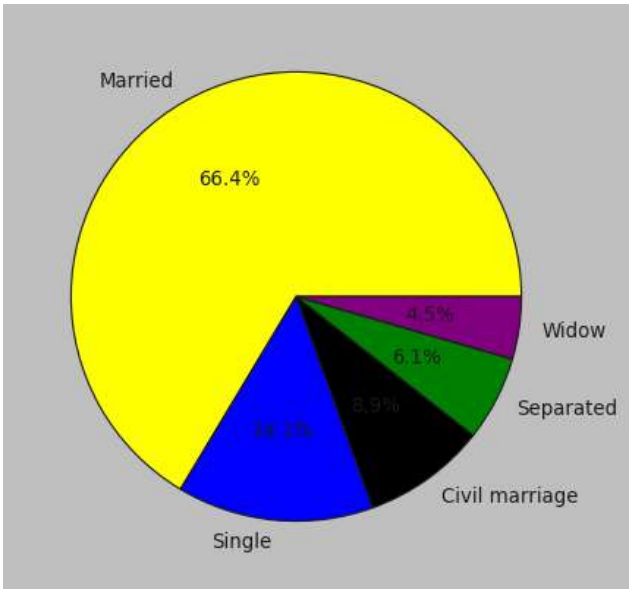
In [40]:
```
FLAG_WORK_PHONE = ['0','1']
quantity = [71802,18283]
plt.pie(quantity,labels=FLAG_WORK_PHONE,autopct='%0.1f%%',colors=['yellow','violet'])
plt.show()
```



In [41]:
```
data['NAME_FAMILY_STATUS'].value_counts()
```

Out[41]:
```
Married                 59825
Single / not married    12708
Civil marriage           8009
Separated                5492
Widow                    4051
Name: NAME_FAMILY_STATUS, dtype: int64
```

In [42]:
```
NAME_FAMILY_STATUS = ['Married','Single','Civil marriage','Separated','Widow']
quantity = [59825, 12708,8009,5492,4051]
plt.pie(quantity,labels=NAME_FAMILY_STATUS,autopct='%0.1f%%',colors=['yellow','blue','black','green','purple'])
plt.show()
```



In [43]:
```
data['NAME_INCOME_TYPE'].value_counts()
```

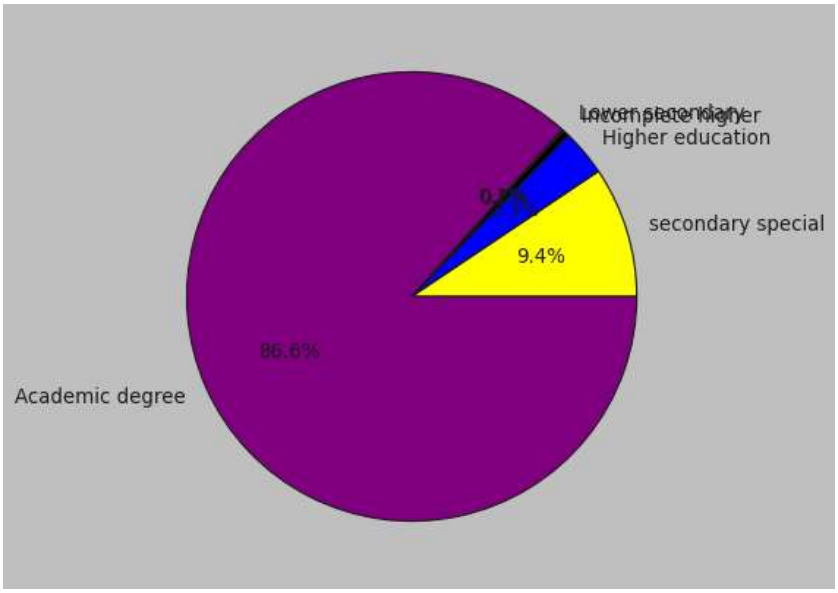
Out[43]:
```
Working                 46633
Commercial associate    20957
Pensioner               15839
State servant            6650
Student                     6
Name: NAME_INCOME_TYPE, dtype: int64
```

In [44]:
```
#NAME_INCOME_TYPE = ['Working','Commercial associate','Pensioner','State servant','Student']
#quantity = [46633, 20957,15839,66506]
#plt.pie(quantity,labels=NAME_INCOME_TYPE,autopct='%0.1f%%',colors=['yellow','blue','black','green','purple'])
#plt.show()
```

In [45]:
```
data['NAME_EDUCATION_TYPE'].value_counts()
```

Out[45]:
```
Secondary / secondary special    63479
Higher education                 22477
Incomplete higher                 3100
Lower secondary                    971
Academic degree                     58
Name: NAME_EDUCATION_TYPE, dtype: int64
```

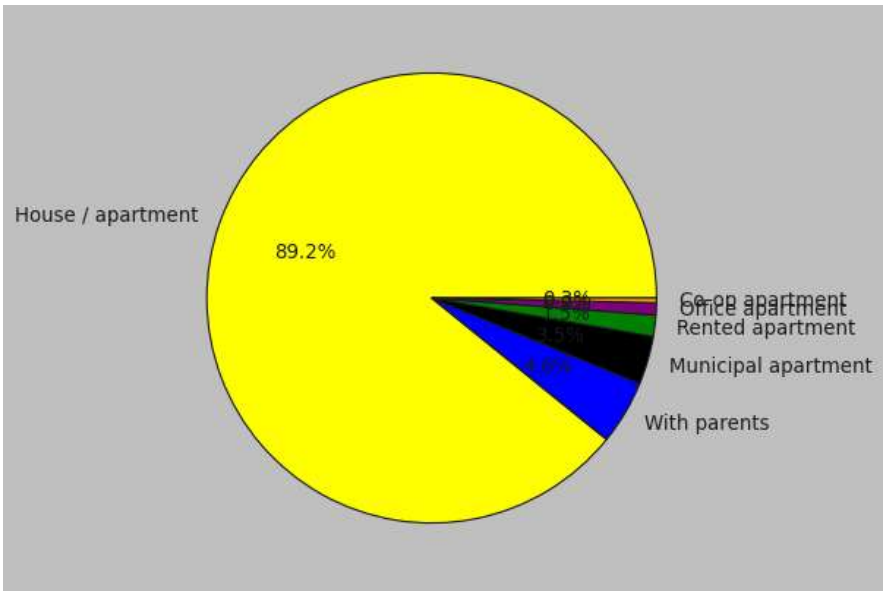
In [46]:
```
NAME_EDUCATION_TYPE = ['secondary special','Higher education','Incomplete higher','Lower secondary','Academic degree
quantity = [63479, 22477,3100,971,584051]
plt.pie(quantity,labels=NAME_EDUCATION_TYPE,autopct='%0.1f%%',colors=['yellow','blue','black','green','purple'])
plt.show()
```



In [47]:
```
data['NAME_HOUSING_TYPE'].value_counts()
```

Out[47]:
```
House / apartment      80335
With parents            4156
Municipal apartment     3130
Rented apartment        1373
Office apartment         790
Co-op apartment          301
Name: NAME_HOUSING_TYPE, dtype: int64
```

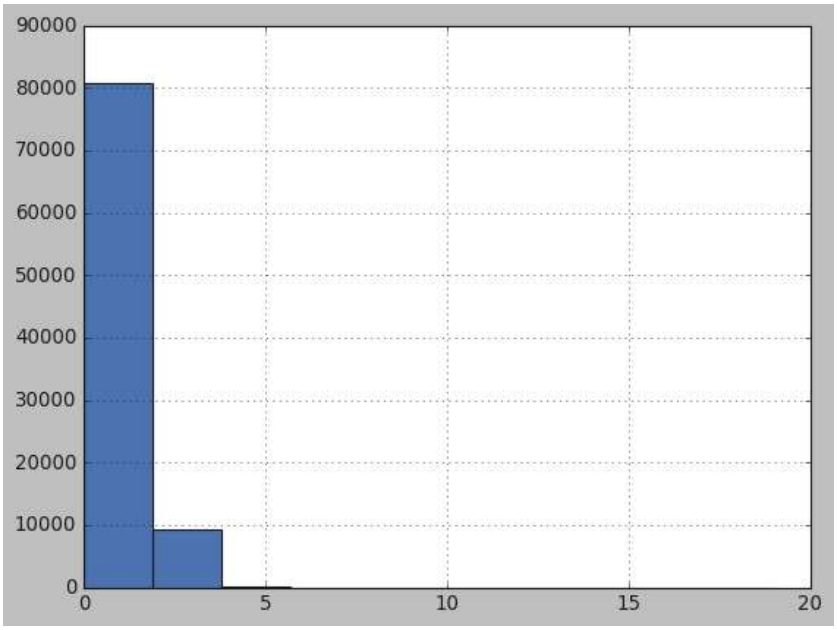
In [48]:
```
HOUSING_TYPE = ['House / apartment','With parents','Municipal apartment','Rented apartment','Office apartment','Co-op
ity = [80335, 4156,3130,1373,790,301]
ie(quantity,labels=NAME_HOUSING_TYPE,autopct='%0.1f%%',colors=['yellow','blue','black','green','purple','orange'])
how()
```



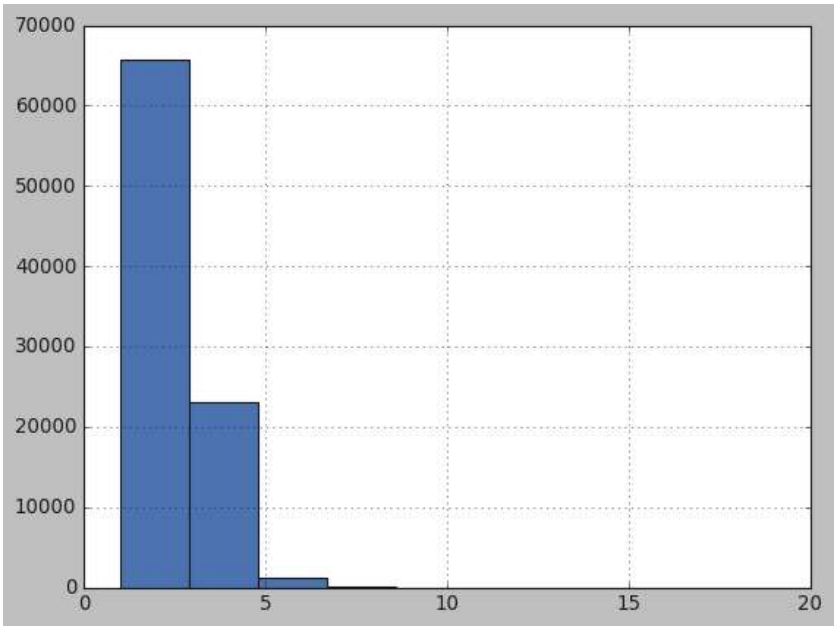
In [49]:
```
data['CNT_CHILDREN'].value_counts()
```

Out[49]:
```
0     62723
1     18026
2      8075
3      1114
4       114
5        25
7         2
6         2
14        1
19        1
9         1
12        1
Name: CNT_CHILDREN, dtype: int64
```

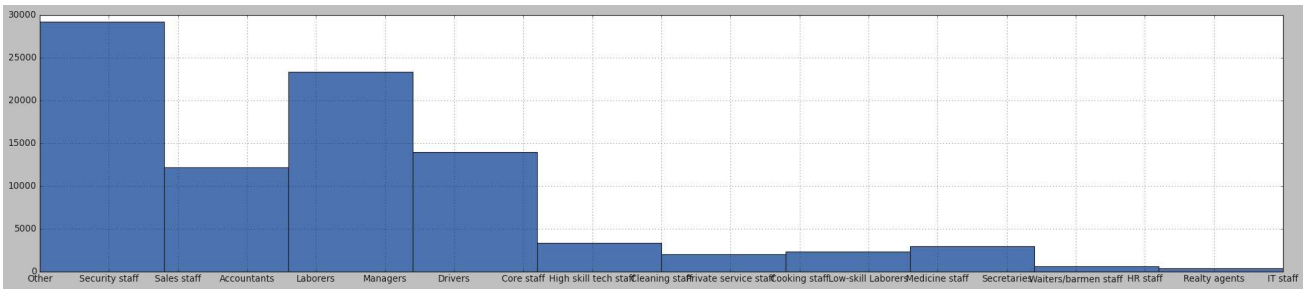
In [50]:
```
data['CNT_CHILDREN'].hist()
plt.show()
```



In [51]:
```
data['CNT_FAM_MEMBERS'].hist()
plt.show()
```

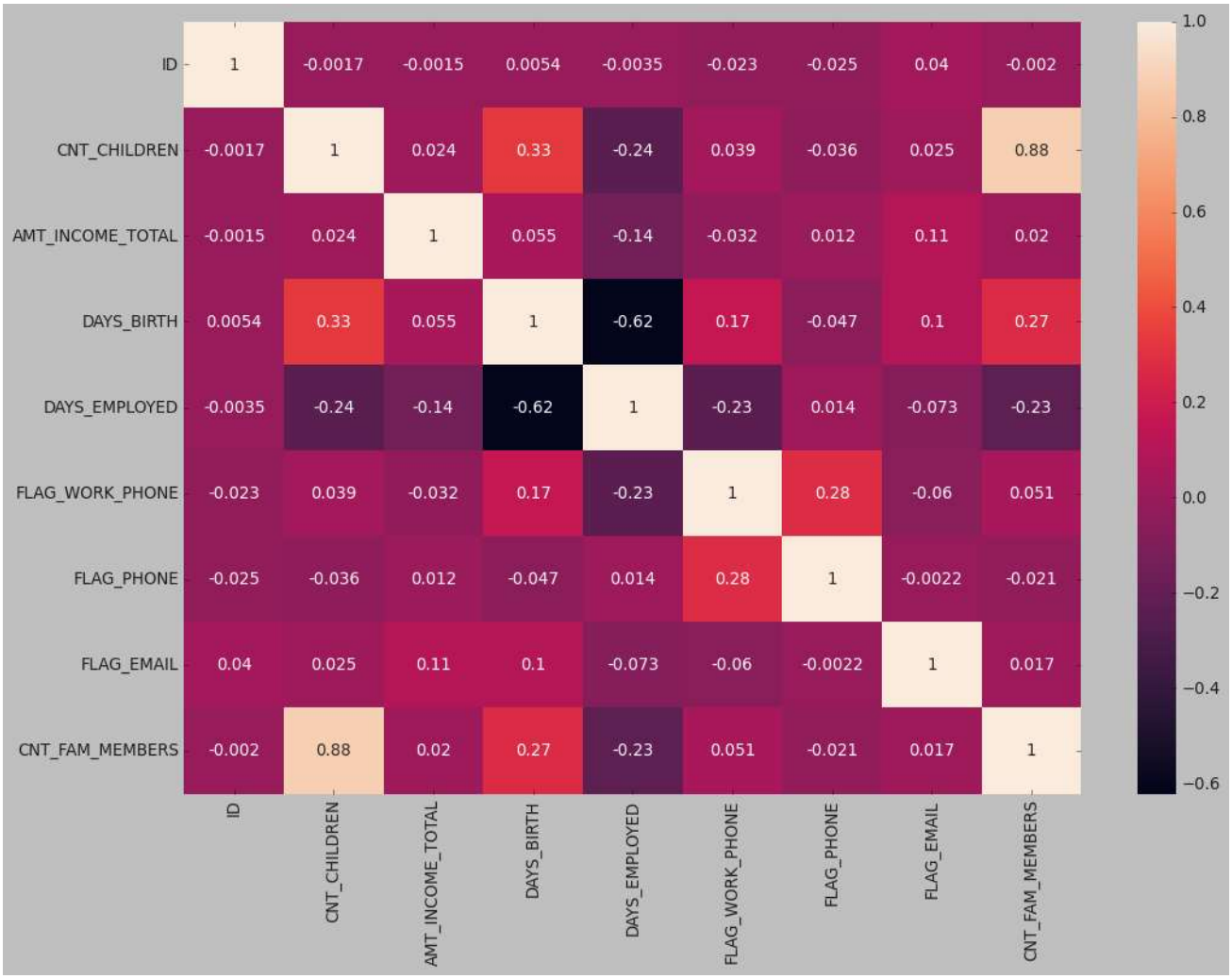


In [120]:
```
plt.figure(figsize=(30,6))
data['OCCUPATION_TYPE'].hist()
plt.show()
```

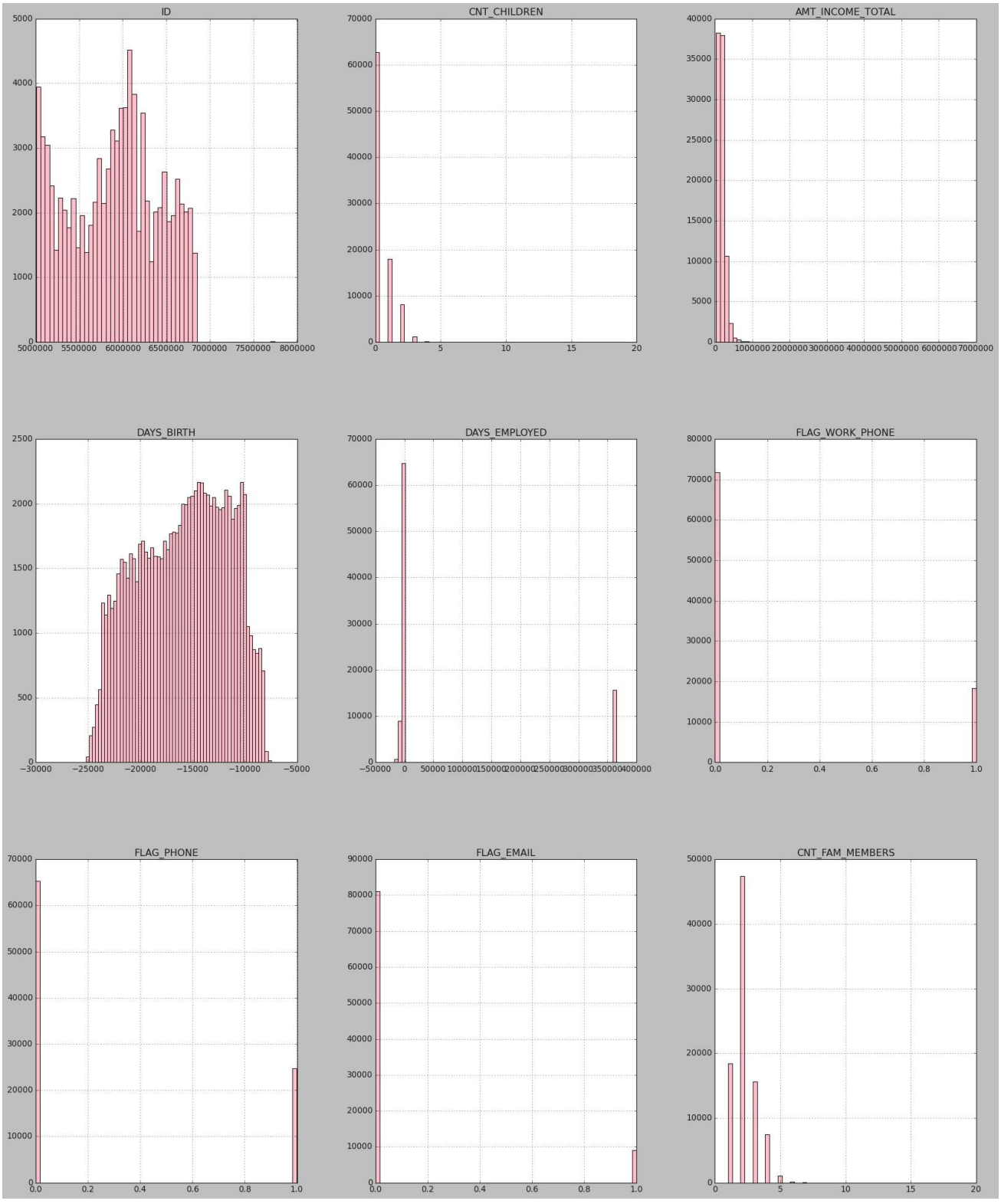


Correlation Matrics

In [53]:
```
plt.figure(figsize= (15,10))
sns.heatmap(data.corr(),annot=True)
plt.show()
```



In [54]:
```
data.hist(bins=60,figsize=(25,30),color='pink')
plt.show()
```



Converting data in a proper format

In [55]:
```
records['target'] = records['STATUS']
records['target'].replace('X',0, inplace=True)
records['target'].replace('C',0, inplace=True)
records['target'] = records['target'].astype(int)
records.loc[records['target']>=1, 'target']=1
```

In [56]:
```
records['STATUS'].value_counts()
```

Out[56]:
```
C    442031
0    383120
X    209230
1     11090
5      1693
2       868
3       320
4       223
Name: STATUS, dtype: int64
```

In [57]:
```
records
```

Out[57]:

	ID	MONTHS_BALANCE	STATUS	target
0	5001711	0	X	0
1	5001711	-1	0	0
2	5001711	-2	0	0
3	5001711	-3	0	0
4	5001712	0	C	0
...	...	...	...	...
1048570	5150487	-25	C	0
1048571	5150487	-26	C	0
1048572	5150487	-27	C	0
1048573	5150487	-28	C	0
1048574	5150487	-29	C	0

1048575 rows × 4 columns

In [58]:
```
records['target'].value_counts()
```

Out[58]:
```
0    1034381
1      14194
Name: target, dtype: int64
```

In [59]:
```
df = pd.DataFrame(records.groupby(['ID'])['target'].agg(max)).reset_index()
```

In [60]:
```
df
```

Out[60]:

	ID	target
0	5001711	0
1	5001712	0
2	5001713	0
3	5001714	0
4	5001715	0
...	...	...
45980	5150482	0
45981	5150483	0
45982	5150484	0
45983	5150485	0
45984	5150487	0

45985 rows × 2 columns

In [61]:
```
df['target'].value_counts()
```

Out[61]:
```
0    40635
1     5350
Name: target, dtype: int64
```

In [62]:
```
new_df = pd.merge(data, df, how='inner', on=['ID'])
```

In [63]:
```
new_df.head()
```

Out[63]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUC
0	5008804	M	Y	Y	0	427500.0	Working	Hig
1	5008806	M	Y	Y	0	112500.0	Working	Seconda
2	5008808	F	N	Y	0	270000.0	Commercial associate	Seconda
3	5008812	F	N	Y	0	283500.0	Pensioner	Hig
4	5008815	M	Y	Y	0	270000.0	Working	Hig

In [64]:
```
new_df.shape
```

Out[64]:
```
(9709, 18)
```

In [65]:
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9709 entries, 0 to 9708
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   9709 non-null   int64  
 1   CODE_GENDER          9709 non-null   object 
 2   FLAG_OWN_CAR         9709 non-null   object 
 3   FLAG_OWN_REALTY      9709 non-null   object 
 4   CNT_CHILDREN         9709 non-null   int64  
 5   AMT_INCOME_TOTAL     9709 non-null   float64
 6   NAME_INCOME_TYPE     9709 non-null   object 
 7   NAME_EDUCATION_TYPE  9709 non-null   object 
 8   NAME_FAMILY_STATUS   9709 non-null   object 
 9   NAME_HOUSING_TYPE    9709 non-null   object 
 10  DAYS_BIRTH           9709 non-null   int64  
 11  DAYS_EMPLOYED        9709 non-null   int64  
 12  FLAG_WORK_PHONE      9709 non-null   int64  
 13  FLAG_PHONE           9709 non-null   int64  
 14  FLAG_EMAIL           9709 non-null   int64  
 15  OCCUPATION_TYPE      9709 non-null   object 
 16  CNT_FAM_MEMBERS      9709 non-null   float64
 17  target               9709 non-null   int32  
dtypes: float64(2), int32(1), int64(7), object(8)
memory usage: 1.4+ MB
```

In [66]:
```
new_df['target'].value_counts()
```

Out[66]:
```
0    8426
1    1283
Name: target, dtype: int64
```

In [67]:
```
new_df['DAYS_BIRTH']
```

Out[67]:
```
0      -12005
1      -21474
2      -19110
3      -22464
4      -16872
        ...
9704   -20600
9705   -15837
9706   -19101
9707   -12387
9708    -9188
Name: DAYS_BIRTH, Length: 9709, dtype: int64
```

In [68]:
```
new df.columns
```

Out[68]:
```
Index(['ID', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN',
       'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'FLAG_WORK_PHONE', 'FLAG_PHONE', 'FLAG_EMAIL',
       'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'target'],
      dtype='object')
```

In [69]:
```
new_df["Age_Years"] = round(-new_df['DAYS_BIRTH']/365,0)
```

In [70]:
```
new_df["Age_Years"]
```

Out[70]:
```
0       33.0
1       59.0
2       52.0
3       62.0
4       46.0
        ...
9704    56.0
9705    43.0
9706    52.0
9707    34.0
9708    25.0
Name: Age_Years, Length: 9709, dtype: float64
```

In [71]:
```
new_df.drop(['DAYS_BIRTH'], axis=1, inplace=True)
```

In [72]:
```
new_df
```

Out[72]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_ED
0	5008804	M	Y	Y	0	427500.0	Working	
1	5008806	M	Y	Y	0	112500.0	Working	Seco
2	5008808	F	N	Y	0	270000.0	Commercial associate	Seco
3	5008812	F	N	Y	0	283500.0	Pensioner	
4	5008815	M	Y	Y	0	270000.0	Working	
...	...	...	...	...	...	...	...	...
9704	5148694	F	N	N	0	180000.0	Pensioner	Seco
9705	5149055	F	N	Y	0	112500.0	Commercial associate	Seco
9706	5149729	M	Y	Y	0	90000.0	Working	Seco
9707	5149838	F	N	Y	0	157500.0	Pensioner	
9708	5150337	M	N	Y	0	112500.0	Working	Seco

9709 rows × 18 columns

In [73]:
```
new_df['Unemployed'] = 0
new_df.loc[-new_df['DAYS_EMPLOYED']<0,'Unemployed']=1
```

In [74]:
```
new_df
```

Out[74]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_ED
0	5008804	M	Y	Y	0	427500.0	Working	
1	5008806	M	Y	Y	0	112500.0	Working	Seco
2	5008808	F	N	Y	0	270000.0	Commercial associate	Seco
3	5008812	F	N	Y	0	283500.0	Pensioner	
4	5008815	M	Y	Y	0	270000.0	Working	
...	...	...	...	...	...	...	...	...
9704	5148694	F	N	N	0	180000.0	Pensioner	Seco
9705	5149055	F	N	Y	0	112500.0	Commercial associate	Seco
9706	5149729	M	Y	Y	0	90000.0	Working	Seco
9707	5149838	F	N	Y	0	157500.0	Pensioner	
9708	5150337	M	N	Y	0	112500.0	Working	Seco

9709 rows × 19 columns

In [75]:
```
new_df["Years_Employed"] = round(-new_df['DAYS_EMPLOYED']/365,0)
```

In [76]:
```
new_df.head()
```

Out[76]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUC
0	5008804	M	Y	Y	0	427500.0	Working	Hig
1	5008806	M	Y	Y	0	112500.0	Working	Seconda
2	5008808	F	N	Y	0	270000.0	Commercial associate	Seconda
3	5008812	F	N	Y	0	283500.0	Pensioner	Hig
4	5008815	M	Y	Y	0	270000.0	Working	Hig

In [77]:
```
new_df.loc[new_df['Years_Employed']<0,'Years_Employed'] = 0
```

In [78]:
```
new_df
```

Out[78]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_ED
0	5008804	M	Y	Y	0	427500.0	Working	
1	5008806	M	Y	Y	0	112500.0	Working	Seco
2	5008808	F	N	Y	0	270000.0	Commercial associate	Seco
3	5008812	F	N	Y	0	283500.0	Pensioner	
4	5008815	M	Y	Y	0	270000.0	Working	
...	...	...	...	...	...	...	...	...
9704	5148694	F	N	N	0	180000.0	Pensioner	Seco
9705	5149055	F	N	Y	0	112500.0	Commercial associate	Seco
9706	5149729	M	Y	Y	0	90000.0	Working	Seco
9707	5149838	F	N	Y	0	157500.0	Pensioner	
9708	5150337	M	N	Y	0	112500.0	Working	Seco

9709 rows × 20 columns

In [79]:
```
new_df.drop("DAYS_EMPLOYED", axis=1, inplace=True)
```

In [80]:
```
new_df
```

Out[80]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_ED
0	5008804	M	Y	Y	0	427500.0	Working	
1	5008806	M	Y	Y	0	112500.0	Working	Seco
2	5008808	F	N	Y	0	270000.0	Commercial associate	Seco
3	5008812	F	N	Y	0	283500.0	Pensioner	
4	5008815	M	Y	Y	0	270000.0	Working	
...	...	...	...	...	...	...	...	...
9704	5148694	F	N	N	0	180000.0	Pensioner	Seco
9705	5149055	F	N	Y	0	112500.0	Commercial associate	Seco
9706	5149729	M	Y	Y	0	90000.0	Working	Seco
9707	5149838	F	N	Y	0	157500.0	Pensioner	
9708	5150337	M	N	Y	0	112500.0	Working	Seco

9709 rows × 19 columns

In [81]:
```
new_df.columns
```

Out[81]:
```
Index(['ID', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN',
       'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'FLAG_WORK_PHONE',
       'FLAG_PHONE', 'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS',
       'target', 'Age_Years', 'Unemployed', 'Years_Employed'],
      dtype='object')
```

Renaming the column names

In [82]:
```
new_df = new_df.rename(columns={'CODE_GENDER':'Gender', 'FLAG_OWN_CAR':'Own_Car',
                                'FLAG_OWN_REALTY':'Own_Property', 'CNT_CHILDREN':'Num_Children',
       'AMT_INCOME_TOTAL':'Total_Income', 'NAME_INCOME_TYPE':'Income_Type',
                                'NAME_EDUCATION_TYPE':'Education_Type',
       'NAME_FAMILY_STATUS':'Family_Status', 'NAME_HOUSING_TYPE':'Housing_Type',
                                'FLAG_WORK_PHONE':"Work_Phone",
       'FLAG_PHONE':'Phone', 'FLAG_EMAIL':'Email', 'OCCUPATION_TYPE':'Occupation_Type',
                                'CNT_FAM_MEMBERS':'Num_Family_Members',
       'target':'Target', 'Age_Years':'Age', 'Unemployed':'Unemployed',
                                'Years_Employed':'Years_Experience'})
```

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

Out[83]:

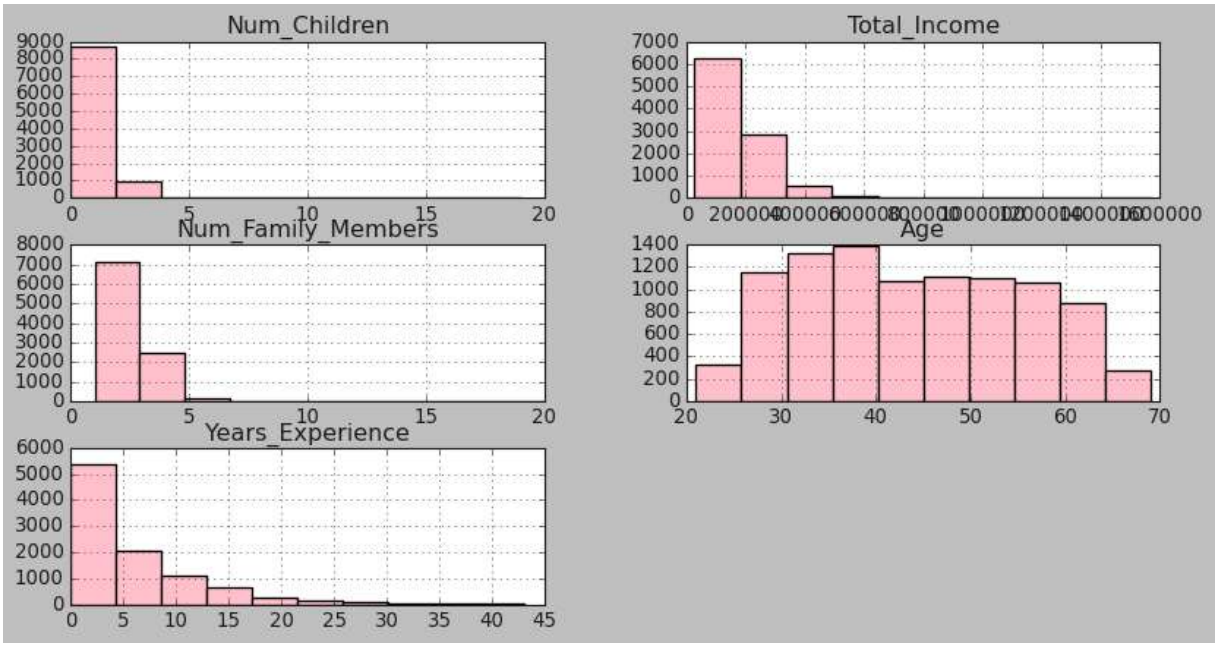
	ID	Gender	Own_Car	Own_Property	Num_Children	Total_Income	Income_Type	Education_Type	Family_Status	Housing_Type	Work_
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	
1	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	Married	House / apartment	
2	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	
3	5008812	F	N	Y	0	283500.0	Pensioner	Higher education	Separated	House / apartment	
4	5008815	M	Y	Y	0	270000.0	Working	Higher education	Married	House / apartment	

In [84]:
```
new_df.dtypes
```

Out[84]:
```
ID                    int64
Gender               object
Own_Car              object
Own_Property         object
Num_Children          int64
Total_Income        float64
Income_Type          object
Education_Type       object
Family_Status        object
Housing_Type         object
Work_Phone            int64
Phone                 int64
Email                 int64
Occupation_Type      object
Num_Family_Members  float64
Target                int32
Age                 float64
Unemployed            int64
Years_Experience    float64
dtype: object
```

In [85]:
```
plt.figure(figsize=(10,10))
cols_to_plot = ["Num_Children","Total_Income","Num_Family_Members","Age","Years_Experience"]
new_df[cols_to_plot].hist(edgecolor='black', linewidth=1.2,color = "pink")
fig=plt.gcf()
fig.set_size_inches(12,6)
```

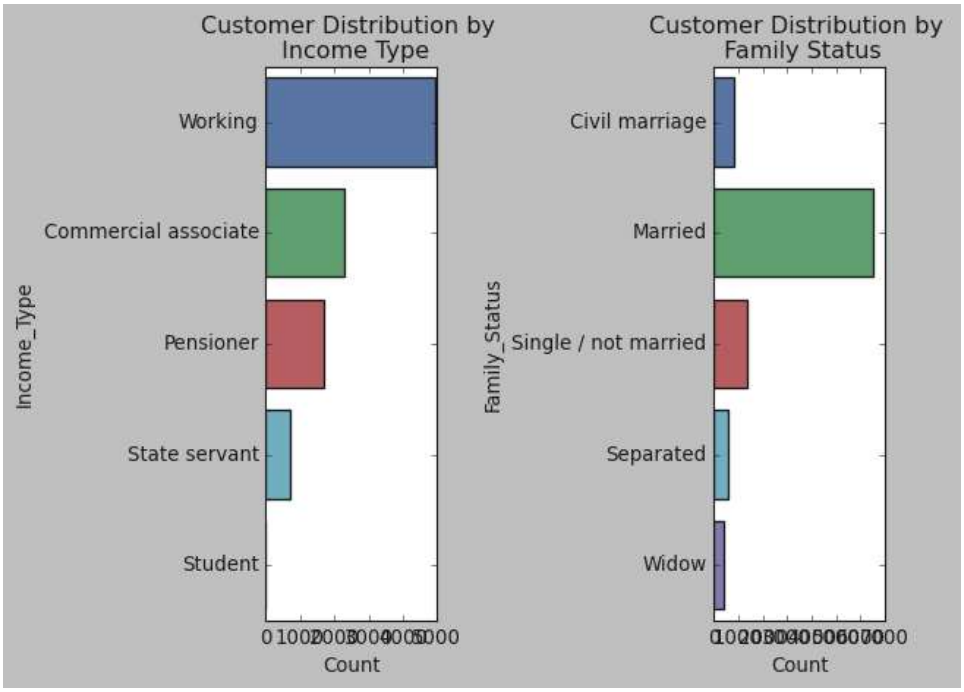
```
<Figure size 800x800 with 0 Axes>
```



In [86]:
```
ig, axes = plt.subplots(1,2)
g1 = sns.countplot(y=new_df.Income_Type, linewidth =1.2, ax=axes[0])
g1.set_title("Customer Distribution by \n Income Type")
g1.set_xlabel("Count")

g2 = sns.countplot(y=new_df.Family_Status, linewidth =1.2, ax=axes[1])
g2.set_title("Customer Distribution by \n Family Status")
g2.set_xlabel("Count")

fig.set_size_inches(14,5)
plt.tight_layout()
plt.show()
```

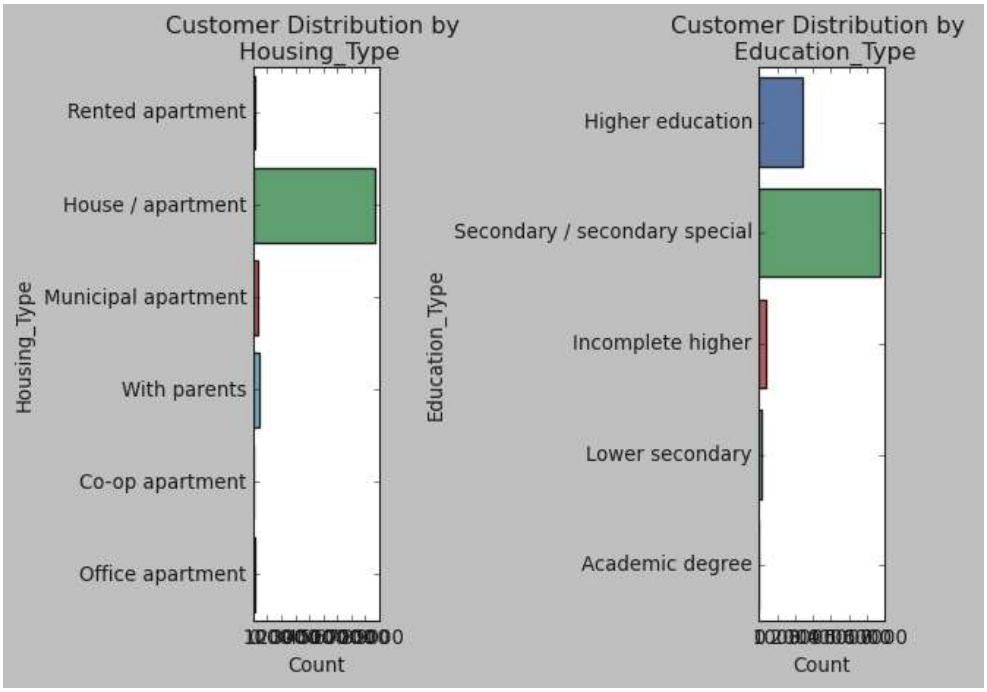


In [87]:
```
ig, axes = plt.subplots(1,2)

g1 = sns.countplot(y=new_df.Housing_Type, linewidth =1.2, ax=axes[0])
g1.set_title("Customer Distribution by \n Housing_Type")
g1.set_xlabel("Count")

g2 = sns.countplot(y=new_df.Education_Type, linewidth =1.2, ax=axes[1])
g2.set_title("Customer Distribution by \n Education_Type")
g2.set_xlabel("Count")

fig.set_size_inches(14,5)
plt.tight_layout()
plt.show()
```



In [88]:
```
object_columns = new_df.columns[new_df.dtypes=='object'].tolist()
```

In [89]:
```
object_columns
```

Out[89]:
```
['Gender',
 'Own_Car',
 'Own_Property',
 'Income_Type',
 'Education_Type',
 'Family_Status',
 'Housing_Type',
 'Occupation_Type']
```

In [90]:
```
unique_counts = pd.DataFrame.from_records([(col, new_df[object_columns][col].nunique()) for col in new_df[object_col
                                           columns=['Column_Name',"No_Unique"]).sort_values(by=['No_Unique'])
```

In [91]:
```
unique_counts
```

Out[91]:

	Column_Name	No_Unique
0	Gender	2
1	Own_Car	2
2	Own_Property	2
3	Income_Type	5
4	Education_Type	5
5	Family_Status	5
6	Housing_Type	6
7	Occupation_Type	19

In [92]:
```
for i in new_df.columns[(new_df.dtypes=='object').values].tolist():
    print(i, '\n')
    print(new_df[i].value_counts())
    print("***************"*5)
```

```
Gender

F    6323
M    3386
Name: Gender, dtype: int64
**************************************************************************
Own_Car

N    6139
Y    3570
Name: Own_Car, dtype: int64
**************************************************************************
Own_Property

Y    6520
N    3189
Name: Own_Property, dtype: int64
**************************************************************************
Income_Type

Working                 4960
Commercial associate    2312
Pensioner               1712
State servant            722
Student                    3
Name: Income_Type, dtype: int64
**************************************************************************
Education_Type

Secondary / secondary special    6761
Higher education                 2457
Incomplete higher                 371
Lower secondary                   114
Academic degree                     6
Name: Education_Type, dtype: int64
**************************************************************************
Family_Status

Married                 6530
Single / not married    1359
Civil marriage           836
Separated                574
Widow                    410
Name: Family_Status, dtype: int64
**************************************************************************
Housing_Type

House / apartment      8684
With parents            448
Municipal apartment     323
Rented apartment        144
Office apartment         76
Co-op apartment          34
Name: Housing_Type, dtype: int64
**************************************************************************
Occupation_Type

Other                    2994
Laborers                 1724
Sales staff               959
Core staff                877
Managers                  782
Drivers                   623
High skill tech staff     357
Accountants               300
Medicine staff            291
Cooking staff             193
Security staff            182
Cleaning staff            146
Private service staff      86
Low-skill Laborers         53
Secretaries                46
Waiters/barmen staff       40
HR staff                   22
IT staff                   18
Realty agents              16
Name: Occupation_Type, dtype: int64
**************************************************************************
```

In [93]:
```
new_df.columns
```

Out[93]:
```
Index(['ID', 'Gender', 'Own_Car', 'Own_Property', 'Num_Children',
       'Total_Income', 'Income_Type', 'Education_Type', 'Family_Status',
       'Housing_Type', 'Work_Phone', 'Phone', 'Email', 'Occupation_Type',
       'Num_Family_Members', 'Target', 'Age', 'Unemployed',
       'Years_Experience'],
      dtype='object')
```

In [94]:
```
# Encode binary features
new_df['Gender'] = new_df['Gender'].replace(['F','M'], [0,1])
new_df['Own_Car'] = new_df['Own_Car'].replace(['Y','N'], [1,0])
new_df['Own_Property'] = new_df['Own_Property'].replace(['Y','N'], [1,0])
```

In [95]:
```
new_df.head()
```

Out[95]:

	ID	Gender	Own_Car	Own_Property	Num_Children	Total_Income	Income_Type	Education_Type	Family_Status	Housing_Type	Work_
0	5008804	1	1	1	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	
1	5008806	1	1	1	0	112500.0	Working	Secondary / secondary special	Married	House / apartment	
2	5008808	0	0	1	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	
3	5008812	0	0	1	0	283500.0	Pensioner	Higher education	Separated	House / apartment	
4	5008815	1	1	1	0	270000.0	Working	Higher education	Married	House / apartment	

In [96]:
```
new_df = pd.get_dummies(new_df, columns=['Income_Type','Education_Type',
                                         'Family_Status',
                                         'Housing_Type','Occupation_Type'], drop_first=True)
```

In [97]:
```
new_df
```

Out[97]:

	ID	Gender	Own_Car	Own_Property	Num_Children	Total_Income	Work_Phone	Phone	Email	Num_Family_Members	...	Occupati
0	5008804	1	1	1	0	427500.0	1	0	0	2.0	...	
1	5008806	1	1	1	0	112500.0	0	0	0	2.0	...	
2	5008808	0	0	1	0	270000.0	0	1	1	1.0	...	
3	5008812	0	0	1	0	283500.0	0	0	0	1.0	...	
4	5008815	1	1	1	0	270000.0	1	1	1	2.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
9704	5148694	0	0	0	0	180000.0	0	0	0	2.0	...	
9705	5149055	0	0	1	0	112500.0	1	1	0	2.0	...	
9706	5149729	1	1	1	0	90000.0	0	0	0	2.0	...	
9707	5149838	0	0	1	0	157500.0	0	1	1	2.0	...	
9708	5150337	1	0	1	0	112500.0	0	0	0	1.0	...	

9709 rows × 49 columns

In [98]:
```
new df.columns
```

Out[98]:
```
Index(['ID', 'Gender', 'Own_Car', 'Own_Property', 'Num_Children',
       'Total_Income', 'Work_Phone', 'Phone', 'Email', 'Num_Family_Members',
       'Target', 'Age', 'Unemployed', 'Years_Experience',
       'Income_Type_Pensioner', 'Income_Type_State servant',
       'Income_Type_Student', 'Income_Type_Working',
       'Education_Type_Higher education', 'Education_Type_Incomplete higher',
       'Education_Type_Lower secondary',
       'Education_Type_Secondary / secondary special', 'Family_Status_Married',
       'Family_Status_Separated', 'Family_Status_Single / not married',
       'Family_Status_Widow', 'Housing_Type_House / apartment',
       'Housing_Type_Municipal apartment', 'Housing_Type_Office apartment',
       'Housing_Type_Rented apartment', 'Housing_Type_With parents',
       '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_Other',
       'Occupation_Type_Private service staff',
       'Occupation_Type_Realty agents', 'Occupation_Type_Sales staff',
       'Occupation_Type_Secretaries', 'Occupation_Type_Security staff',
       'Occupation_Type_Waiters/barmen staff'],
      dtype='object')
```

In [99]:
```
new_df.describe().T
```

Out[99]:

	count	mean	std	min	25%	50%	75%	max
ID	9709.0	5.076105e+06	40802.696053	5008804.0	5036955.0	5069449.0	5112986.0	5150479.0
Gender	9709.0	3.487486e-01	0.476599	0.0	0.0	0.0	1.0	1.0
Own_Car	9709.0	3.677001e-01	0.482204	0.0	0.0	0.0	1.0	1.0
Own_Property	9709.0	6.715419e-01	0.469677	0.0	0.0	1.0	1.0	1.0
Num_Children	9709.0	4.228036e-01	0.767019	0.0	0.0	0.0	1.0	19.0
Total_Income	9709.0	1.812282e+05	99277.305097	27000.0	112500.0	157500.0	225000.0	1575000.0
Work_Phone	9709.0	2.174271e-01	0.412517	0.0	0.0	0.0	0.0	1.0
Phone	9709.0	2.876712e-01	0.452700	0.0	0.0	0.0	1.0	1.0
Email	9709.0	8.754764e-02	0.282650	0.0	0.0	0.0	0.0	1.0
Num_Family_Members	9709.0	2.182614e+00	0.932918	1.0	2.0	2.0	3.0	20.0
Target	9709.0	1.321454e-01	0.338666	0.0	0.0	0.0	0.0	1.0
Age	9709.0	4.381759e+01	11.643172	21.0	34.0	43.0	54.0	69.0
Unemployed	9709.0	1.746833e-01	0.379716	0.0	0.0	0.0	0.0	1.0
Years_Experience	9709.0	5.680091e+00	6.354310	0.0	1.0	4.0	8.0	43.0
Income_Type_Pensioner	9709.0	1.763312e-01	0.381121	0.0	0.0	0.0	0.0	1.0
Income_Type_State servant	9709.0	7.436399e-02	0.262376	0.0	0.0	0.0	0.0	1.0
Income_Type_Student	9709.0	3.089917e-04	0.017576	0.0	0.0	0.0	0.0	1.0
Income_Type_Working	9709.0	5.108662e-01	0.499908	0.0	0.0	1.0	1.0	1.0
Education_Type_Higher education	9709.0	2.530642e-01	0.434790	0.0	0.0	0.0	1.0	1.0
Education_Type_Incomplete higher	9709.0	3.821197e-02	0.191717	0.0	0.0	0.0	0.0	1.0
Education_Type_Lower secondary	9709.0	1.174168e-02	0.107727	0.0	0.0	0.0	0.0	1.0
Education_Type_Secondary / secondary special	9709.0	6.963642e-01	0.459851	0.0	0.0	1.0	1.0	1.0
Family_Status_Married	9709.0	6.725718e-01	0.469299	0.0	0.0	1.0	1.0	1.0
Family_Status_Separated	9709.0	5.912040e-02	0.235862	0.0	0.0	0.0	0.0	1.0
Family_Status_Single / not married	9709.0	1.399732e-01	0.346977	0.0	0.0	0.0	0.0	1.0
Family_Status_Widow	9709.0	4.222886e-02	0.201121	0.0	0.0	0.0	0.0	1.0
Housing_Type_House / apartment	9709.0	8.944279e-01	0.307305	0.0	1.0	1.0	1.0	1.0
Housing_Type_Municipal apartment	9709.0	3.326810e-02	0.179345	0.0	0.0	0.0	0.0	1.0
Housing_Type_Office apartment	9709.0	7.827789e-03	0.088132	0.0	0.0	0.0	0.0	1.0
Housing_Type_Rented apartment	9709.0	1.483160e-02	0.120885	0.0	0.0	0.0	0.0	1.0
Housing_Type_With parents	9709.0	4.614275e-02	0.209805	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Cleaning staff	9709.0	1.503759e-02	0.121709	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Cooking staff	9709.0	1.987846e-02	0.139590	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Core staff	9709.0	9.032856e-02	0.286667	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Drivers	9709.0	6.416727e-02	0.245063	0.0	0.0	0.0	0.0	1.0
Occupation_Type_HR staff	9709.0	2.265939e-03	0.047550	0.0	0.0	0.0	0.0	1.0
Occupation_Type_High skill tech staff	9709.0	3.677001e-02	0.188206	0.0	0.0	0.0	0.0	1.0
Occupation_Type_IT staff	9709.0	1.853950e-03	0.043020	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Laborers	9709.0	1.775672e-01	0.382168	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Low-skill Laborers	9709.0	5.458853e-03	0.073686	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Managers	9709.0	8.054383e-02	0.272147	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Medicine staff	9709.0	2.997219e-02	0.170519	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Other	9709.0	3.083737e-01	0.461846	0.0	0.0	0.0	1.0	1.0
Occupation_Type_Private service staff	9709.0	8.857761e-03	0.093703	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Realty agents	9709.0	1.647956e-03	0.040564	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Sales staff	9709.0	9.877433e-02	0.298374	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Secretaries	9709.0	4.737872e-03	0.068672	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Security staff	9709.0	1.874549e-02	0.135632	0.0	0.0	0.0	0.0	1.0
Occupation_Type_Waiters/barmen staff	9709.0	4.119889e-03	0.064057	0.0	0.0	0.0	0.0	1.0

In [100]:
```
X = new_df.drop(['ID','Target','Phone','Email'], axis=1)
Y = new_df['Target']
```

In [101]:
```
X
```

Out[101]:

	Gender	Own_Car	Own_Property	Num_Children	Total_Income	Work_Phone	Num_Family_Members	Age	Unemployed	Years_Experience
0	1	1	1	0	427500.0	1	2.0	33.0	0	12.0
1	1	1	1	0	112500.0	0	2.0	59.0	0	3.0
2	0	0	1	0	270000.0	0	1.0	52.0	0	8.0
3	0	0	1	0	283500.0	0	1.0	62.0	1	0.0
4	1	1	1	0	270000.0	1	2.0	46.0	0	2.0
...	...	...	...	...	...	...	...	...	...	...
9704	0	0	0	0	180000.0	0	2.0	56.0	0	1.0
9705	0	0	1	0	112500.0	1	2.0	43.0	0	7.0
9706	1	1	1	0	90000.0	0	2.0	52.0	0	5.0
9707	0	0	1	0	157500.0	0	2.0	34.0	0	4.0
9708	1	0	1	0	112500.0	0	1.0	25.0	0	3.0

9709 rows × 45 columns

In [102]:
```
# Feature scaling
from sklearn.preprocessing import Normalizer
scaler = Normalizer()
X = scaler.fit_transform(X)
pd.DataFrame(X)
```

Out[102]:

	0	1	2	3	4	5	6	7	8	9	...	35	36	37	38	39	40	
0	0.000002	0.000002	0.000002	0.0	1.0	0.000002	0.000005	0.000077	0.000000	0.000028	...	0.0	0.0	0.000000	0.000002	0.0	0.0	0.0000
1	0.000009	0.000009	0.000009	0.0	1.0	0.000000	0.000018	0.000524	0.000000	0.000027	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0000
2	0.000000	0.000000	0.000004	0.0	1.0	0.000000	0.000004	0.000193	0.000000	0.000030	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0000
3	0.000000	0.000000	0.000004	0.0	1.0	0.000000	0.000004	0.000219	0.000004	0.000000	...	0.0	0.0	0.000000	0.000004	0.0	0.0	0.0000
4	0.000004	0.000004	0.000004	0.0	1.0	0.000004	0.000007	0.000170	0.000000	0.000007	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9704	0.000000	0.000000	0.000000	0.0	1.0	0.000000	0.000011	0.000311	0.000000	0.000006	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0000
9705	0.000000	0.000000	0.000009	0.0	1.0	0.000009	0.000018	0.000382	0.000000	0.000062	...	0.0	0.0	0.000000	0.000009	0.0	0.0	0.0000
9706	0.000011	0.000011	0.000011	0.0	1.0	0.000000	0.000022	0.000578	0.000000	0.000056	...	0.0	0.0	0.000000	0.000011	0.0	0.0	0.0000
9707	0.000000	0.000000	0.000006	0.0	1.0	0.000000	0.000013	0.000216	0.000000	0.000025	...	0.0	0.0	0.000006	0.000000	0.0	0.0	0.0000
9708	0.000009	0.000000	0.000009	0.0	1.0	0.000000	0.000009	0.000222	0.000000	0.000027	...	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0000

9709 rows × 45 columns

SMOTE

In [103]:
```
# balance the data
```

In [104]:
```
from imblearn.over_sampling import SMOTE
sm = SMOTE()
X_sm, Y_sm = sm.fit_resample(X, Y)
print(Y.value_counts())
print(Y_sm.value_counts())
```

```
0    8426
1    1283
Name: Target, dtype: int64
1    8426
0    8426
Name: Target, dtype: int64
```

In [105]:
```
# Split the data into training and test
from sklearn.model_selection import train_test_split
x train, x test, y train, y test = train test split(X sm, Y sm, test size=0.25, random state=101)
```

In [106]:
```
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(x train, y train)
```

Out[106]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
```

In [107]:
```
y_pred_train_xgb = xgb.predict(x_train)
y_pred_test_xgb = xgb.predict(x_test)
```

In [108]:
```
from sklearn.metrics import classification_report, accuracy_score
```

In [109]:
```
print(classification_report(y_train, y_pred_train_xgb))
print("**********"*10)
print(classification_report(y_test, y_pred_test_xgb))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95      6250
           1       1.00      0.90      0.95      6389

    accuracy                           0.95     12639
   macro avg       0.95      0.95      0.95     12639
weighted avg       0.95      0.95      0.95     12639

**********************************************************************************************************************
*************************
              precision    recall  f1-score   support

           0       0.86      0.98      0.92      2176
           1       0.97      0.84      0.90      2037

    accuracy                           0.91      4213
   macro avg       0.92      0.91      0.91      4213
weighted avg       0.92      0.91      0.91      4213
```

In [110]:
```
print("Training Accuracy", accuracy_score(y_train, y_pred_train_xgb))
print("**********"*5)
print("Test Accuracy", accuracy_score(y_test, y_pred_test_xgb))
```

```
Training Accuracy 0.9478597990347337
**************************************************
Test Accuracy 0.910040351293615
```

K-Fold method

In [111]:
```
from sklearn.model_selection import cross_val_score
training = cross_val_score(xgb, x_train, y_train, cv=10)
test = cross_val_score(xgb, x_test, y_test, cv=10)
```

In [112]:
```
print("Training Accuracy", training.mean())
print("**********"*5)
print("Test Accuracy", test.mean())
```

```
Training Accuracy 0.9054510934383675
**************************************************
Test Accuracy 0.8853542119305198
```

In [113]:
```
training
```

Out[113]:
```
array([0.89952532, 0.91139241, 0.90901899, 0.91297468, 0.90664557,
       0.89873418, 0.90348101, 0.90585443, 0.90506329, 0.90182106])
```

In [114]:
```
test
```

Out[114]:
```
array([0.88388626, 0.88388626, 0.89099526, 0.87885986, 0.89073634,
       0.88598575, 0.89786223, 0.89073634, 0.86698337, 0.88361045])
```

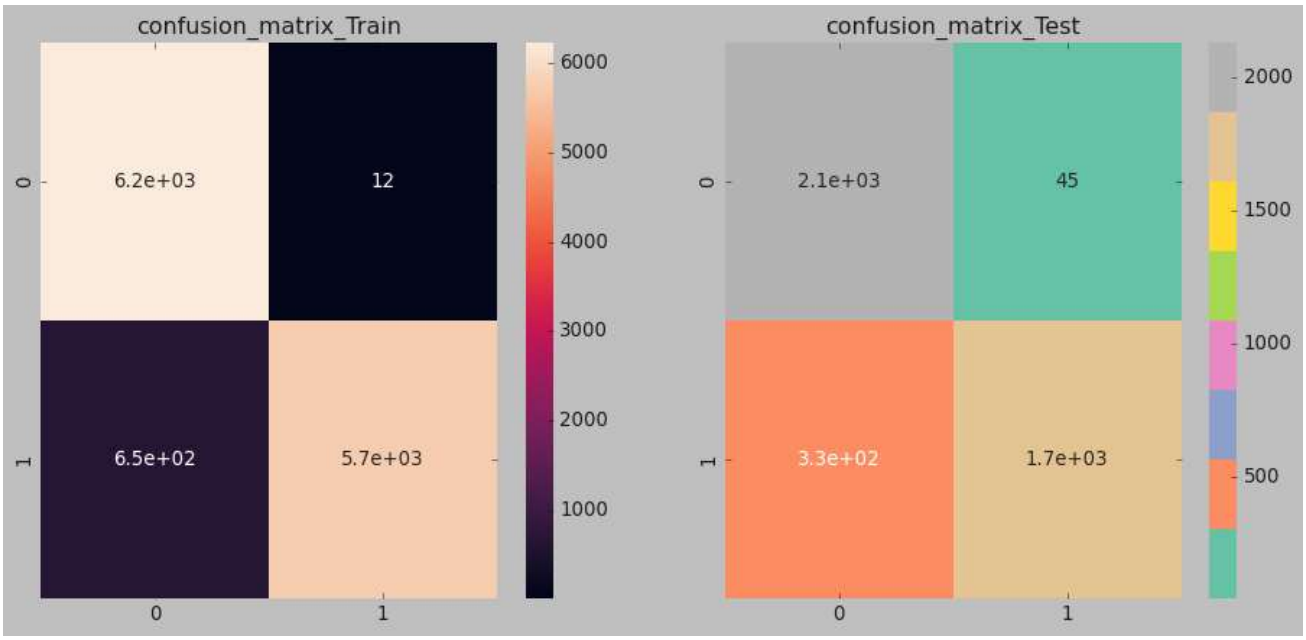
In [115]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
cm_train_sc= confusion_matrix(y_train,  y_pred_train_xgb)
cm_test_sc = confusion_matrix(y_test, y_pred_test_xgb)
print('===='*10)
print( 'confusion_matrix_Test :','\n','\n',cm_test_sc)
print('===='*10)
print( 'confusion_matrix_Train :','\n','\n',cm_train_sc)
print('===='*10)
plt.figure(figsize = (14,6))
plt.subplot(1,2,1)
sns.heatmap(cm_train_sc, annot = True, )
plt.title('confusion_matrix_Train')
plt.subplot(1,2,2)
sns.heatmap(cm_test_sc, annot = True, cmap= 'Set2')
plt.title('confusion_matrix_Test')
```

```
========================================
confusion_matrix_Test :

 [[2131   45]
 [ 334 1703]]
========================================
confusion_matrix_Train :

 [[6238   12]
 [ 647 5742]]
========================================
```

Out[115]:
```
Text(0.5, 1.0, 'confusion_matrix_Test')
```



In [116]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay,roc_auc_
xgb_roc_auc = roc_auc_score(y_test, y_pred_test_xgb)
print('Area under curve :',xgb_roc_auc)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_test_xgb)
display(fpr[:10])
display(tpr[:10])
display(thresholds[:10])
```

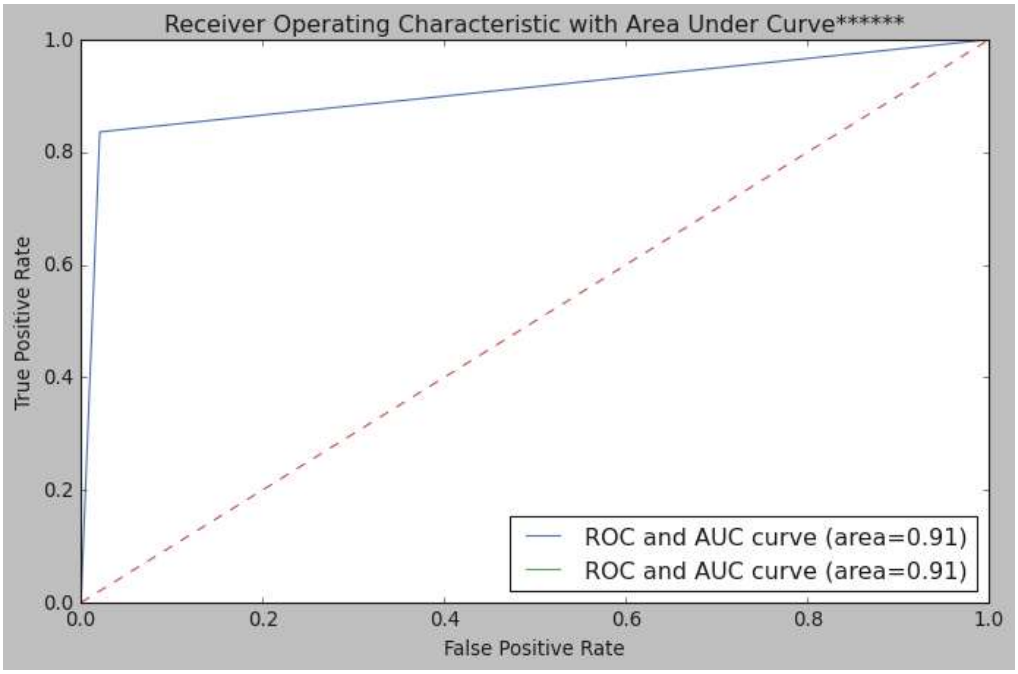
```
Area under curve : 0.9076766176831558

array([0.        , 0.02068015, 1.        ])

array([0.        , 0.83603338, 1.        ])

array([2, 1, 0])
```

In [117]:
```
plt.figure(figsize=(10,6))
plt.plot(fpr, tpr,thresholds, label="ROC and AUC curve (area=%0.2f)" % xgb_roc_auc)
plt.plot([0,1],[0,1], 'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("Receiver Operating Characteristic with Area Under Curve")
plt.legend(loc='lower right')
plt.show()
```



In [ ]: