Prepared by Asif Bhat

Exploratory Data Analysis ¶

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly as ply
import seaborn as sns
import warnings
import plotly.graph_objects as go
import plotly.offline as po
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, ipl
import plotly.express as px
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
warnings.filterwarnings('ignore')
import plotly.io as pio
pio.renderers.default = 'iframe'
pio.templates.default = "plotly_dark"
```

Explore Dataset

In [2]:
```
app_data = pd.read_csv('application_data.csv')
app_data.head()
```

Out[2]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_F
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	

5 rows × 122 columns

In [3]:
```
app_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

In [4]:
```
app_data.shape # 122 Columns
```
Out[4]:
```
(307511, 122)
```

In [5]:
```
# Summary of numeric columns
app_data.describe()
```
Out[5]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_AN
count	307511.000000	307511.000000	307511.000000	3.075110e+05	3.075110e+05	307499.(
mean	278180.518577	0.080729	0.417052	1.687979e+05	5.990260e+05	27108.!
std	102790.175348	0.272419	0.722121	2.371231e+05	4.024908e+05	14493.7
min	100002.000000	0.000000	0.000000	2.565000e+04	4.500000e+04	1615.!
25%	189145.500000	0.000000	0.000000	1.125000e+05	2.700000e+05	16524.(
50%	278202.000000	0.000000	0.000000	1.471500e+05	5.135310e+05	24903.(
75%	367142.500000	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.(
max	456255.000000	1.000000	19.000000	1.170000e+08	4.050000e+06	258025.!

8 rows × 106 columns

In [6]:
```
# Most of the columns are of type integer or float.
app_data.dtypes.value_counts()
```
Out[6]:
```
float64    65
int64      41
object     16
dtype: int64
```

Data Cleaning

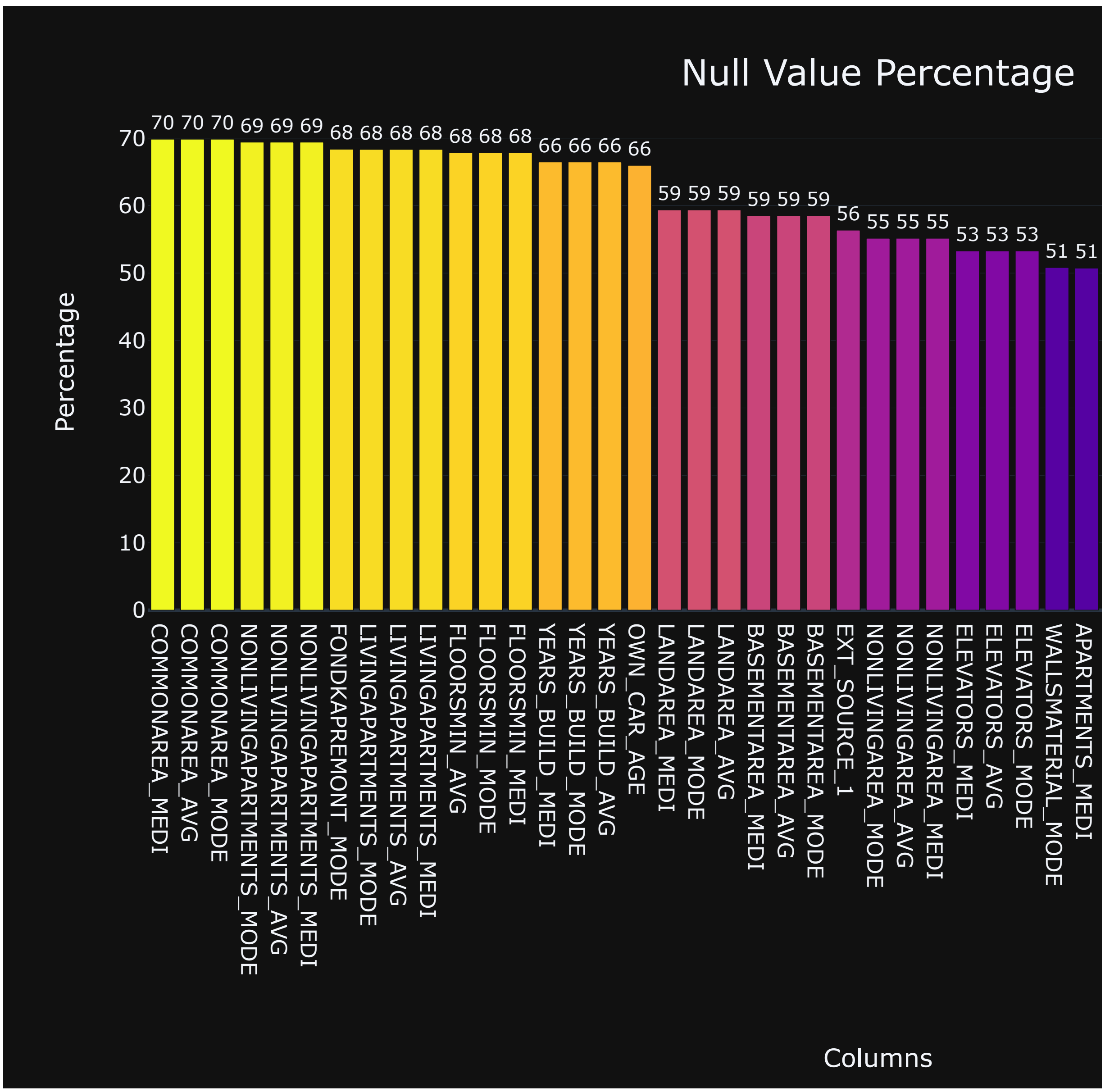
Dropping Columns with high percentage of NULL values

In [7]:
```
# Percentage of NULL Values in descending order
(app_data.isnull().mean()*100).sort_values(ascending=False)
```
Out[7]:
```
COMMONAREA_MEDI             69.872297
COMMONAREA_AVG              69.872297
COMMONAREA_MODE             69.872297
NONLIVINGAPARTMENTS_MODE    69.432963
NONLIVINGAPARTMENTS_AVG     69.432963
                              ...
NAME_HOUSING_TYPE            0.000000
NAME_FAMILY_STATUS           0.000000
NAME_EDUCATION_TYPE          0.000000
NAME_INCOME_TYPE             0.000000
SK_ID_CURR                   0.000000
Length: 122, dtype: float64
```

In [8]:
```
# Columns with NULL Values greater than 40%
s1= (app_data.isnull().mean()*100).sort_values(ascending=False)[app_data.is
s1
```

Out[8]:
```
COMMONAREA_MEDI                 69.872297
COMMONAREA_AVG                  69.872297
COMMONAREA_MODE                 69.872297
NONLIVINGAPARTMENTS_MODE        69.432963
NONLIVINGAPARTMENTS_AVG         69.432963
NONLIVINGAPARTMENTS_MEDI        69.432963
FONDKAPREMONT_MODE              68.386172
LIVINGAPARTMENTS_MODE           68.354953
LIVINGAPARTMENTS_AVG            68.354953
LIVINGAPARTMENTS_MEDI           68.354953
FLOORSMIN_AVG                   67.848630
FLOORSMIN_MODE                  67.848630
FLOORSMIN_MEDI                  67.848630
YEARS_BUILD_MEDI                66.497784
YEARS_BUILD_MODE                66.497784
YEARS_BUILD_AVG                 66.497784
OWN_CAR_AGE                     65.990810
LANDAREA_MEDI                   59.376738
LANDAREA_MODE                   59.376738
LANDAREA_AVG                    59.376738
BASEMENTAREA_MEDI               58.515956
BASEMENTAREA_AVG                58.515956
BASEMENTAREA_MODE               58.515956
EXT_SOURCE_1                    56.381073
NONLIVINGAREA_MODE              55.179164
NONLIVINGAREA_AVG               55.179164
NONLIVINGAREA_MEDI              55.179164
ELEVATORS_MEDI                  53.295980
ELEVATORS_AVG                   53.295980
ELEVATORS_MODE                  53.295980
WALLSMATERIAL_MODE              50.840783
APARTMENTS_MEDI                 50.749729
APARTMENTS_AVG                  50.749729
APARTMENTS_MODE                 50.749729
ENTRANCES_MEDI                  50.348768
ENTRANCES_AVG                   50.348768
ENTRANCES_MODE                  50.348768
LIVINGAREA_AVG                  50.193326
LIVINGAREA_MODE                 50.193326
LIVINGAREA_MEDI                 50.193326
HOUSETYPE_MODE                  50.176091
FLOORSMAX_MODE                  49.760822
FLOORSMAX_MEDI                  49.760822
FLOORSMAX_AVG                   49.760822
YEARS_BEGINEXPLUATATION_MODE    48.781019
YEARS_BEGINEXPLUATATION_MEDI    48.781019
YEARS_BEGINEXPLUATATION_AVG     48.781019
TOTALAREA_MODE                  48.268517
EMERGENCYSTATE_MODE             47.398304
dtype: float64
```

In [9]:
```
fig= px.bar(data_frame=s1,
       x=s1.index.tolist(),
       y=s1.values,
       color=s1.values,
       text=s1.values.round()
      )
fig.update_traces(textposition='outside',marker_coloraxis=None)
fig.update_xaxes(title='Columns')
fig.update_yaxes(title='Percentage')
fig.update_layout(
                  title=dict(text = "Null Value Percentage",x=0.5,y=0.95),
                  title_font_size=20,
                  showlegend=False,
                  height =600,
                 )
fig.show()
```



In [10]:
```
# Get Column names with NULL percentage greater than 40%
cols = (app_data.isnull().mean()*100 > 40)[app_data.isnull().mean()*100 > 4
cols
```

Out[10]:
```
['OWN_CAR_AGE',
 'EXT_SOURCE_1',
 'APARTMENTS_AVG',
 'BASEMENTAREA_AVG',
 'YEARS_BEGINEXPLUATATION_AVG',
 'YEARS_BUILD_AVG',
 'COMMONAREA_AVG',
 'ELEVATORS_AVG',
 'ENTRANCES_AVG',
 'FLOORSMAX_AVG',
 'FLOORSMIN_AVG',
 'LANDAREA_AVG',
 'LIVINGAPARTMENTS_AVG',
 'LIVINGAREA_AVG',
 'NONLIVINGAPARTMENTS_AVG',
 'NONLIVINGAREA_AVG',
 'APARTMENTS_MODE',
 'BASEMENTAREA_MODE',
 'YEARS_BEGINEXPLUATATION_MODE',
 'YEARS_BUILD_MODE',
 'COMMONAREA_MODE',
 'ELEVATORS_MODE',
 'ENTRANCES_MODE',
 'FLOORSMAX_MODE',
 'FLOORSMIN_MODE',
 'LANDAREA_MODE',
 'LIVINGAPARTMENTS_MODE',
 'LIVINGAREA_MODE',
 'NONLIVINGAPARTMENTS_MODE',
 'NONLIVINGAREA_MODE',
 'APARTMENTS_MEDI',
 'BASEMENTAREA_MEDI',
 'YEARS_BEGINEXPLUATATION_MEDI',
 'YEARS_BUILD_MEDI',
 'COMMONAREA_MEDI',
 'ELEVATORS_MEDI',
 'ENTRANCES_MEDI',
 'FLOORSMAX_MEDI',
 'FLOORSMIN_MEDI',
 'LANDAREA_MEDI',
 'LIVINGAPARTMENTS_MEDI',
 'LIVINGAREA_MEDI',
 'NONLIVINGAPARTMENTS_MEDI',
 'NONLIVINGAREA_MEDI',
 'FONDKAPREMONT_MODE',
 'HOUSETYPE_MODE',
 'TOTALAREA_MODE',
 'WALLSMATERIAL_MODE',
 'EMERGENCYSTATE_MODE']
```

In [11]:
```
# We are good to delete 49 columns because NULL percentage for these column
len(cols)
```

Out[11]: 49

In [12]:
```
# Drop 49 columns
app_data.drop(columns=cols,inplace=True)
```

In [13]:
```
app_data.shape # 307511 rows & 73 Columns
```

Out[13]: (307511, 73)

In [14]:
```
# NULL Values percentage in new dataset
s2= (app_data.isnull().mean()*100).sort_values(ascending=False)
s2
```

Out[14]:
```
OCCUPATION_TYPE                31.345545
EXT_SOURCE_3                   19.825307
AMT_REQ_CREDIT_BUREAU_YEAR     13.501631
AMT_REQ_CREDIT_BUREAU_QRT      13.501631
AMT_REQ_CREDIT_BUREAU_MON      13.501631
                                 ...
REG_REGION_NOT_LIVE_REGION      0.000000
REG_REGION_NOT_WORK_REGION      0.000000
LIVE_REGION_NOT_WORK_REGION     0.000000
TARGET                          0.000000
REG_CITY_NOT_LIVE_CITY          0.000000
Length: 73, dtype: float64
```

In [15]:
```
s2.head(10)
```

Out[15]:
```
OCCUPATION_TYPE               31.345545
EXT_SOURCE_3                  19.825307
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_HOUR    13.501631
NAME_TYPE_SUITE                0.420148
OBS_30_CNT_SOCIAL_CIRCLE       0.332021
dtype: float64
```

Imputation of Missing Values

In [16]:
```
app_data.head()
```

Out[16]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_I
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	

5 rows × 73 columns

In [17]:
```
'''
Impute the missing values of below columns with mode
- AMT_REQ_CREDIT_BUREAU_MONTH
- AMT_REQ_CREDIT_BUREAU_WEEK
- AMT_REQ_CREDIT_BUREAU_DAY
- AMT_REQ_CREDIT_BUREAU_HOUR
- AMT_REQ_CREDIT_BUREAU_QRT
'''
for i in s2.head(10).index.to_list():
    if 'AMT_REQ_CREDIT' in i:
        print('Most frequent value in {0} is : {1}'.format(i,app_data[i].mo
        print('Imputing the missing value with : {0}'.format(app_data[i].mo
        app_data[i].fillna(app_data[i].mode()[0],inplace=True)
        print('NULL Values in {0} after imputation : {1}'.format(i,app_data
        print()
```

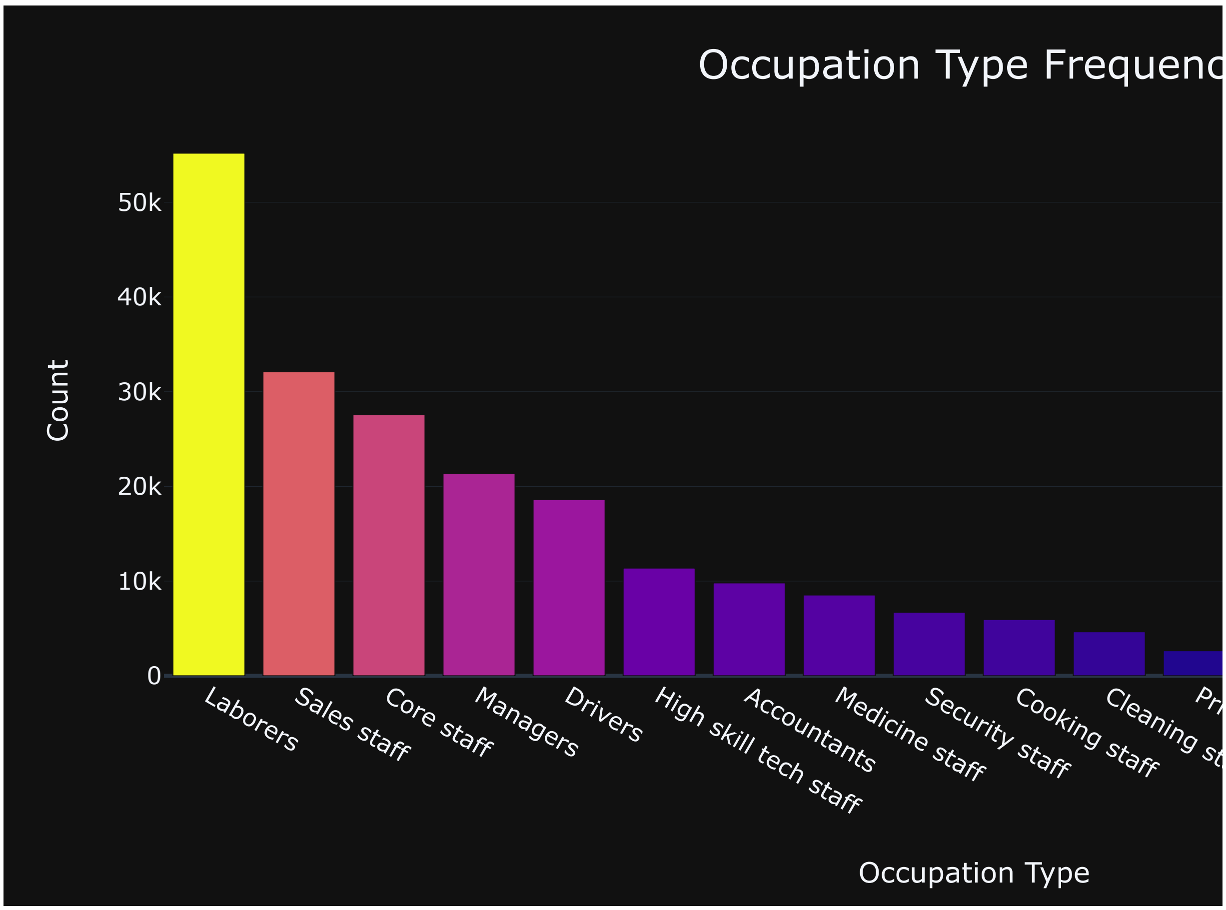
```
Most frequent value in AMT_REQ_CREDIT_BUREAU_YEAR is : 0.0
Imputing the missing value with : 0.0
NULL Values in AMT_REQ_CREDIT_BUREAU_YEAR after imputation : 0

Most frequent value in AMT_REQ_CREDIT_BUREAU_QRT is : 0.0
Imputing the missing value with : 0.0
NULL Values in AMT_REQ_CREDIT_BUREAU_QRT after imputation : 0

Most frequent value in AMT_REQ_CREDIT_BUREAU_MON is : 0.0
Imputing the missing value with : 0.0
NULL Values in AMT_REQ_CREDIT_BUREAU_MON after imputation : 0

Most frequent value in AMT_REQ_CREDIT_BUREAU_WEEK is : 0.0
Imputing the missing value with : 0.0
NULL Values in AMT_REQ_CREDIT_BUREAU_WEEK after imputation : 0

Most frequent value in AMT_REQ_CREDIT_BUREAU_DAY is : 0.0
Imputing the missing value with : 0.0
NULL Values in AMT_REQ_CREDIT_BUREAU_DAY after imputation : 0

Most frequent value in AMT_REQ_CREDIT_BUREAU_HOUR is : 0.0
Imputing the missing value with : 0.0
NULL Values in AMT_REQ_CREDIT_BUREAU_HOUR after imputation : 0
```

In [18]:
```
# Missing value percentage of remaining columns
(app_data.isnull().mean()*100).sort_values(ascending=False)
```

Out[18]:
```
OCCUPATION_TYPE                31.345545
EXT_SOURCE_3                   19.825307
NAME_TYPE_SUITE                 0.420148
OBS_30_CNT_SOCIAL_CIRCLE        0.332021
DEF_30_CNT_SOCIAL_CIRCLE        0.332021
                                  ...
REG_REGION_NOT_LIVE_REGION      0.000000
REG_REGION_NOT_WORK_REGION      0.000000
LIVE_REGION_NOT_WORK_REGION     0.000000
TARGET                          0.000000
AMT_REQ_CREDIT_BUREAU_YEAR      0.000000
Length: 73, dtype: float64
```

Impute missing values for OCCUPATION_TYPE

In [19]:
```
# We can impute missing values in 'OCCUPATION_TYPE' column with 'Laborers'
fig=px.bar(app_data.OCCUPATION_TYPE.value_counts(),color=app_data.OCCUPATIO
fig.update_traces(textposition='outside',marker_coloraxis=None)
fig.update_xaxes(title='Occupation Type')
fig.update_yaxes(title='Count')
fig.update_layout(
                    title=dict(text = "Occupation Type Frequency",x=0.5,y=0
                    title_font_size=20,
                    showlegend=False,
                    height =450,
                  )
fig.show()
```



In [20]:
```
app_data.OCCUPATION_TYPE.fillna('Laborers',inplace=True)
```

Impute Missing values (XNA) in CODE_GENDER with mode

In [21]:
```
app_data['CODE_GENDER'].value_counts()
```

Out[21]:
```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

In [22]:
```
app_data['CODE_GENDER'].replace(to_replace='XNA',value=app_data['CODE_GENDE
```

In [23]:
```
app_data['CODE_GENDER'].value_counts()
```

Out[23]:
```
F    202452
M    105059
Name: CODE_GENDER, dtype: int64
```

Impute missing values for EXT_SOURCE_3

In [24]:
```
app_data.EXT_SOURCE_3.dtype
```

Out[24]:
```
dtype('float64')
```

In [25]:
```
app_data.EXT_SOURCE_3.fillna(app_data.EXT_SOURCE_3.median(),inplace=True)
```

In [26]:
```
# Percentage of missing values after imputation
(app_data.isnull().mean()*100).sort_values(ascending=False)
```

Out[26]:
```
NAME_TYPE_SUITE                0.420148
DEF_60_CNT_SOCIAL_CIRCLE       0.332021
OBS_30_CNT_SOCIAL_CIRCLE       0.332021
DEF_30_CNT_SOCIAL_CIRCLE       0.332021
OBS_60_CNT_SOCIAL_CIRCLE       0.332021
                                 ...
REG_REGION_NOT_LIVE_REGION     0.000000
REG_REGION_NOT_WORK_REGION     0.000000
LIVE_REGION_NOT_WORK_REGION    0.000000
TARGET                         0.000000
AMT_REQ_CREDIT_BUREAU_YEAR     0.000000
Length: 73, dtype: float64
```

In [27]:
```
# Replace 'XNA' with NaN
app_data = app_data.replace('XNA',np.NaN)
```

DELETE all flag columns

In [28]:
```
app_data.columns
```

Out[28]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOT
AL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOB
IL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHO
NE',
       'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS',
       'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY',
       'ORGANIZATION_TYPE', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', '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', 'FLAG_DOCUMENT_12',
       'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
       'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18',
       'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')
```

In [29]:
```
# Flag Columns
col =[]
for i in app_data.columns:
    if 'FLAG' in i:
        col.append(i)
col
```

Out[29]:
```
['FLAG_OWN_CAR',
 'FLAG_OWN_REALTY',
 'FLAG_MOBIL',
 'FLAG_EMP_PHONE',
 'FLAG_WORK_PHONE',
 'FLAG_CONT_MOBILE',
 'FLAG_PHONE',
 'FLAG_EMAIL',
 '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',
 'FLAG_DOCUMENT_12',
 'FLAG_DOCUMENT_13',
 'FLAG_DOCUMENT_14',
 'FLAG_DOCUMENT_15',
 'FLAG_DOCUMENT_16',
 'FLAG_DOCUMENT_17',
 'FLAG_DOCUMENT_18',
 'FLAG_DOCUMENT_19',
 'FLAG_DOCUMENT_20',
 'FLAG_DOCUMENT_21']
```

In [30]:
```
# DELETE all flag columns as they won't be much useful in our analysis
app_data.drop(columns=col,inplace=True)
app_data.head()

#OR

#app_data= app_data[[i for i in  app_data.columns if 'FLAG' not in i]]
```

Out[30]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	CNT_CHILDREN	AMT_INCOME
0	100002	1	Cash loans	M	0	2
1	100003	0	Cash loans	F	0	2
2	100004	0	Revolving loans	M	0	
3	100006	0	Cash loans	F	0	1
4	100007	0	Cash loans	M	0	1

5 rows × 45 columns

Impute Missing values for AMT_ANNUITY & AMT_GOODS_PRICE

In [31]:
```
col=['AMT_INCOME_TOTAL','AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE']
for i in col:
    print('Null Values in {0} : {1}'.format(i,app_data[i].isnull().sum()))
```

```
Null Values in AMT_INCOME_TOTAL : 0
Null Values in AMT_CREDIT : 0
Null Values in AMT_ANNUITY : 12
Null Values in AMT_GOODS_PRICE : 278
```

In [32]:
```
app_data['AMT_ANNUITY'].fillna(app_data['AMT_ANNUITY'].median(),inplace=Tru
app_data['AMT_GOODS_PRICE'].fillna(app_data['AMT_GOODS_PRICE'].median(),inp
app_data['AMT_ANNUITY'].isnull().sum()
app_data['AMT_GOODS_PRICE'].isnull().sum()
```

Out[32]: 0

Correcting Data

In [33]:
```
days = []
for i in app_data.columns:
    if 'DAYS' in i:
        days.append(i)
        print('Unique Values in {0} column : {1}'.format(i,app_data[i].uniq
        print('NULL Values in {0} column : {1}'.format(i,app_data[i].isnull
        print()
```

```
Unique Values in DAYS_BIRTH column : [ -9461 -16765 -19046 ...  -7951  -7
857 -25061]
NULL Values in DAYS_BIRTH column : 0

Unique Values in DAYS_EMPLOYED column : [  -637  -1188   -225 ... -12971
-11084  -8694]
NULL Values in DAYS_EMPLOYED column : 0

Unique Values in DAYS_REGISTRATION column : [ -3648.  -1186.  -4260. ...
-16396. -14558. -14798.]
NULL Values in DAYS_REGISTRATION column : 0

Unique Values in DAYS_ID_PUBLISH column : [-2120  -291 -2531 ... -6194 -5
854 -6211]
NULL Values in DAYS_ID_PUBLISH column : 0

Unique Values in DAYS_LAST_PHONE_CHANGE column : [-1134.  -828.  -815.
... -3988. -3899. -3538.]
NULL Values in DAYS_LAST_PHONE_CHANGE column : 1
```

In [34]:
```
app_data[days]
```

Out[34]:

	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH	DAYS_LAST_PHO
0	-9461	-637	-3648.0	-2120	
1	-16765	-1188	-1186.0	-291	
2	-19046	-225	-4260.0	-2531	
3	-19005	-3039	-9833.0	-2437	
4	-19932	-3038	-4311.0	-3458	
...	...	...	...	...	
307506	-9327	-236	-8456.0	-1982	
307507	-20775	365243	-4388.0	-4090	
307508	-14966	-7921	-6737.0	-5150	
307509	-11961	-4786	-2562.0	-931	
307510	-16856	-1262	-5128.0	-410	

307511 rows × 5 columns

In [35]:
```
# Use absolute values in DAYS columns
app_data[days] = abs(app_data[days])
app_data[days]
```

Out[35]:

	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH	DAYS_LAST_PHO
0	9461	637	3648.0	2120	
1	16765	1188	1186.0	291	
2	19046	225	4260.0	2531	
3	19005	3039	9833.0	2437	
4	19932	3038	4311.0	3458	
...	...	...	...	...	
307506	9327	236	8456.0	1982	
307507	20775	365243	4388.0	4090	
307508	14966	7921	6737.0	5150	
307509	11961	4786	2562.0	931	
307510	16856	1262	5128.0	410	

307511 rows × 5 columns

Binning

In [36]:
```
# Lets do binning of these variables
for i in col:
    app_data[i+'_Range']=pd.qcut(app_data[i],q=5,labels=['Very Low' , 'Low'
    print(app_data[i+'_Range'].value_counts())
    print()
```

```
Low          85756
High         75513
Very Low     63671
Very High    47118
Medium       35453
Name: AMT_INCOME_TOTAL_Range, dtype: int64

Very Low     64925
High         64024
Medium       61552
Very High    58912
Low          58098
Name: AMT_CREDIT_Range, dtype: int64

Medium       61569
Very Low     61507
Low          61499
Very High    61484
High         61452
Name: AMT_ANNUITY_Range, dtype: int64

Very Low     71454
Medium       61533
Very High    61430
High         61349
Low          51745
Name: AMT_GOODS_PRICE_Range, dtype: int64
```

In [37]:
```
app_data['YEARS_EMPLOYED']= app_data['DAYS_EMPLOYED']/365
app_data['Client_Age']= app_data['DAYS_BIRTH']/365
```

In [38]:
```
# Drop 'DAYS_EMPLOYED'& 'DAYS_BIRTH' column as we will be performing analys
app_data.drop(columns=['DAYS_EMPLOYED','DAYS_BIRTH'],inplace=True)
```

In [39]:
```
app_data['Age Group']=pd.cut(
                             x=app_data['Client_Age'],
                             bins=[0,20,30,40,50,60,100],
                             labels=['0-20','20-30','30-40','40-50','50-60'
                             )
```

In [40]:
```
app_data[['SK_ID_CURR','Client_Age','Age Group']]
```

Out[40]:

	SK_ID_CURR	Client_Age	Age Group
0	100002	25.920548	20-30
1	100003	45.931507	40-50
2	100004	52.180822	50-60
3	100006	52.068493	50-60
4	100007	54.608219	50-60
...	...	...	...
307506	456251	25.553425	20-30
307507	456252	56.917808	50-60
307508	456253	41.002740	40-50
307509	456254	32.769863	30-40
307510	456255	46.180822	40-50

307511 rows × 3 columns

In [41]:
```
app_data['Work Experience']=pd.cut(
                              x=app_data['YEARS_EMPLOYED'],
                              bins=[0,5,10,15,20,25,30,100],
                              labels=['0-5','5-10','10-15','15-20','20-25','
                              )
```

In [42]:
```
app_data[['SK_ID_CURR','YEARS_EMPLOYED','Work Experience']]
```

Out[42]:

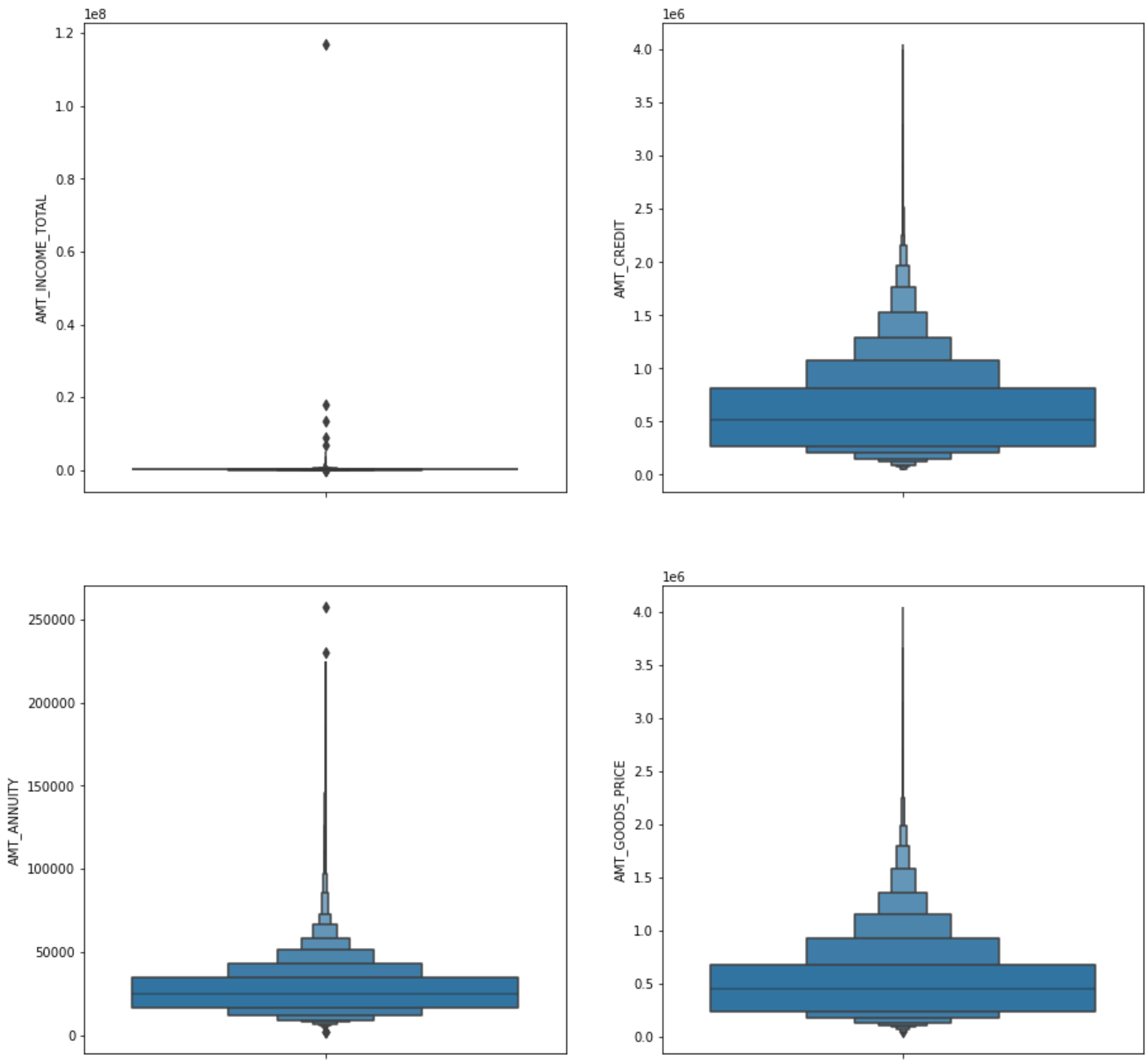
	SK_ID_CURR	YEARS_EMPLOYED	Work Experience
0	100002	1.745205	0-5
1	100003	3.254795	0-5
2	100004	0.616438	0-5
3	100006	8.326027	5-10
4	100007	8.323288	5-10
...	...	...	...
307506	456251	0.646575	0-5
307507	456252	1000.665753	NaN
307508	456253	21.701370	20-25
307509	456254	13.112329	10-15
307510	456255	3.457534	0-5

307511 rows × 3 columns

Outlier Detection

Analyzing AMT column for Outliers

In [43]:
```
cols= ['AMT_INCOME_TOTAL','AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE']
fig,axes = plt.subplots(ncols=2,nrows=2,figsize=(15,15))
count=0
for i in range(0,2):
    for j in range(0,2):
        sns.boxenplot(y=app_data[cols[count]],ax=axes[i,j])
        count+=1
plt.show()
```



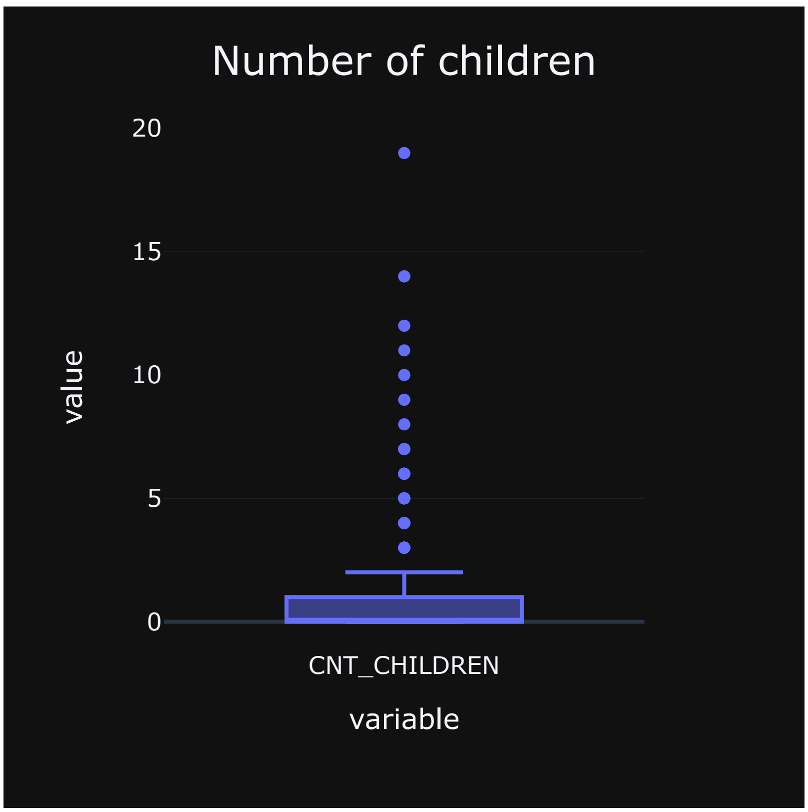
Below Columns have Outliers and those values can be dropped :-

- AMT_INCOME_TOTAL
- AMT_ANNUITY

In [44]:
```
#Remove Outlier for 'AMT_INCOME_TOTAL' column
app_data=app_data[app_data['AMT_INCOME_TOTAL']<app_data['AMT_INCOME_TOTAL']

#Remove Outlier for 'AMT_ANNUITY' column
app_data=app_data[app_data['AMT_ANNUITY']<app_data['AMT_ANNUITY'].max()]
```

Analysing CNT_CHILDREN column for Outliers

In [45]:
```
fig=px.box(app_data['CNT_CHILDREN'])
fig.update_layout(
                    title=dict(text = "Number of children",x=0.5,y=0.95),
                    title_font_size=20,
                    showlegend=False,
                    width =400,
                    height =400,
                  )
fig.show()
```



In [46]:
```
app_data['CNT_CHILDREN'].value_counts()
```

Out[46]:
```
0      215371
1       61118
2       26748
3        3717
4         429
5          84
6          21
7           7
14          3
8           2
9           2
12          2
10          2
19          2
11          1
Name: CNT_CHILDREN, dtype: int64
```

In [47]:
```
app_data.shape[0]
```

Out[47]:
```
307509
```

In [48]:
```
# Remove all data points where CNT_CHILDREN is greater than 10
app_data= app_data[app_data['CNT_CHILDREN']<=10]
app_data.shape[0]
```

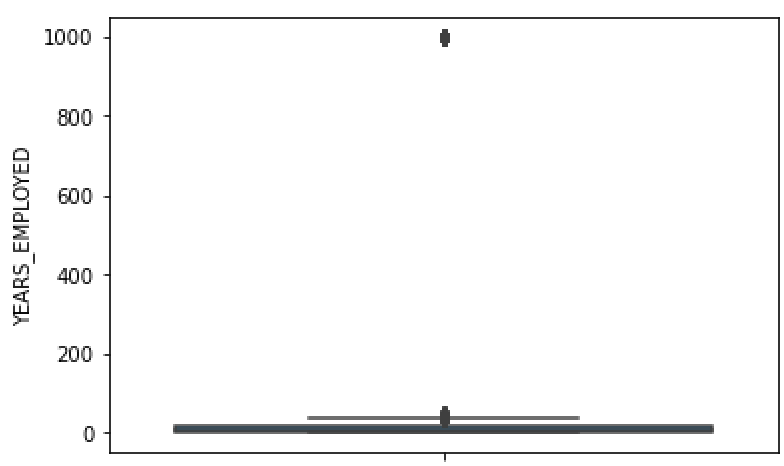
Out[48]:
```
307501
```

Eight values dropped where number of children are greater than 10

Analysing YEARS_EMPLOYED column for Outliers

In [49]:
```
sns.boxplot(y=app_data['YEARS_EMPLOYED'])
```

Out[49]:
```
<AxesSubplot:ylabel='YEARS_EMPLOYED'>
```



In [50]:
```
app_data['YEARS_EMPLOYED'].value_counts()
```

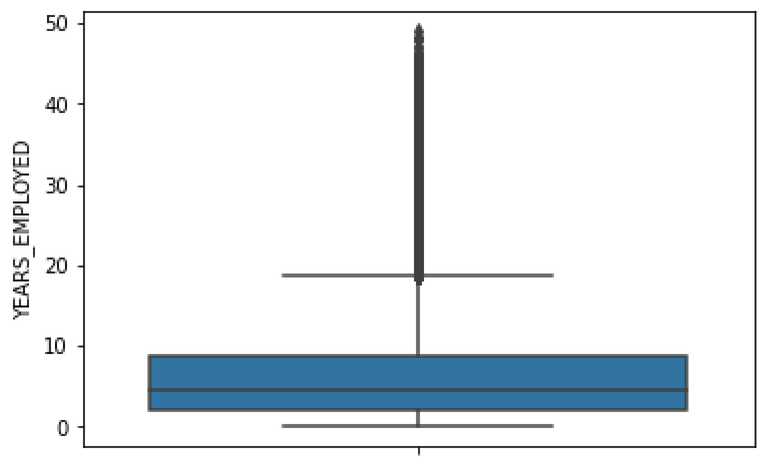
Out[50]:
```
1000.665753    55373
0.547945         156
0.613699         152
0.630137         151
0.545205         151
               ...  
38.249315          1
32.402740          1
27.879452          1
25.915068          1
23.819178          1
Name: YEARS_EMPLOYED, Length: 12574, dtype: int64
```

In [51]:
```
app_data.shape[0]
```

Out[51]:
```
307501
```

In [52]:
```
app_data['YEARS_EMPLOYED'][app_data['YEARS_EMPLOYED']>1000]=np.NaN
```

In [53]:
```
sns.boxplot(y=app_data['YEARS_EMPLOYED'])
plt.show()
```



In [54]:
```
app_data.isnull().sum().sort_values(ascending=False).head(10)
```

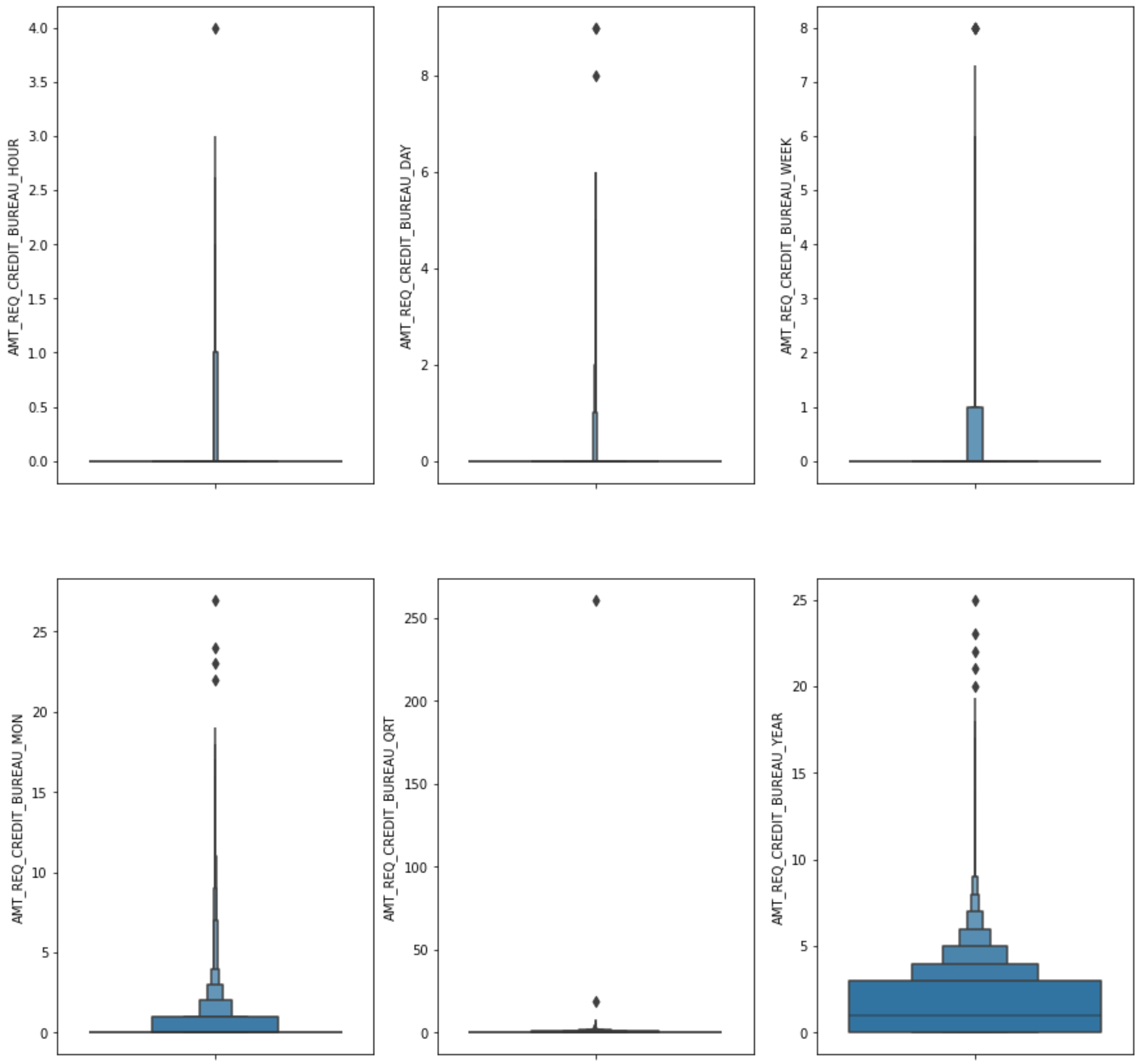
Out[54]:
```
Work Experience             55375
ORGANIZATION_TYPE           55373
YEARS_EMPLOYED              55373
NAME_TYPE_SUITE              1292
OBS_30_CNT_SOCIAL_CIRCLE     1021
DEF_30_CNT_SOCIAL_CIRCLE     1021
OBS_60_CNT_SOCIAL_CIRCLE     1021
DEF_60_CNT_SOCIAL_CIRCLE     1021
EXT_SOURCE_2                  660
CNT_FAM_MEMBERS                 2
dtype: int64
```

Analyzing AMT_REQ_CREDIT columns for Outliers

In [55]:
```
cols = [i for i in  app_data.columns if 'AMT_REQ' in i]
cols
```

Out[55]:
```
['AMT_REQ_CREDIT_BUREAU_HOUR',
 'AMT_REQ_CREDIT_BUREAU_DAY',
 'AMT_REQ_CREDIT_BUREAU_WEEK',
 'AMT_REQ_CREDIT_BUREAU_MON',
 'AMT_REQ_CREDIT_BUREAU_QRT',
 'AMT_REQ_CREDIT_BUREAU_YEAR']
```

In [56]:
```
fig,axes = plt.subplots(ncols=3,nrows=2,figsize=(15,15))
count=0
for i in range(0,2):
    for j in range(0,3):
        sns.boxenplot(y=app_data[cols[count]],ax=axes[i,j])
        count+=1
plt.show()
```



AMT_REQ_CREDIT_BUREAU_QRT contains an outlier

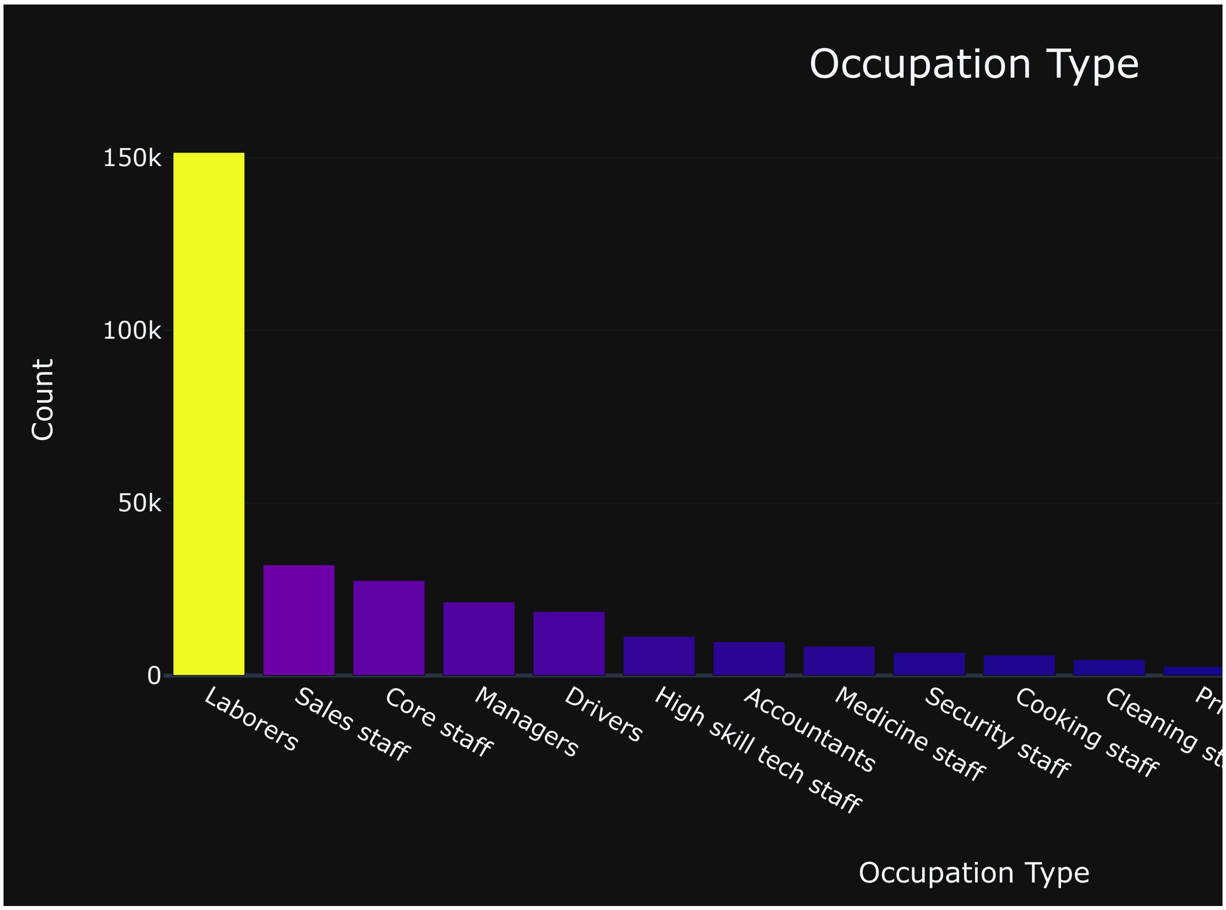
In [57]:
```
# Remove Outlier for AMT_REQ_CREDIT_BUREAU_QRT
app_data=app_data[app_data['AMT_REQ_CREDIT_BUREAU_QRT']<app_data['AMT_REQ_C
```

Univariate Analysis

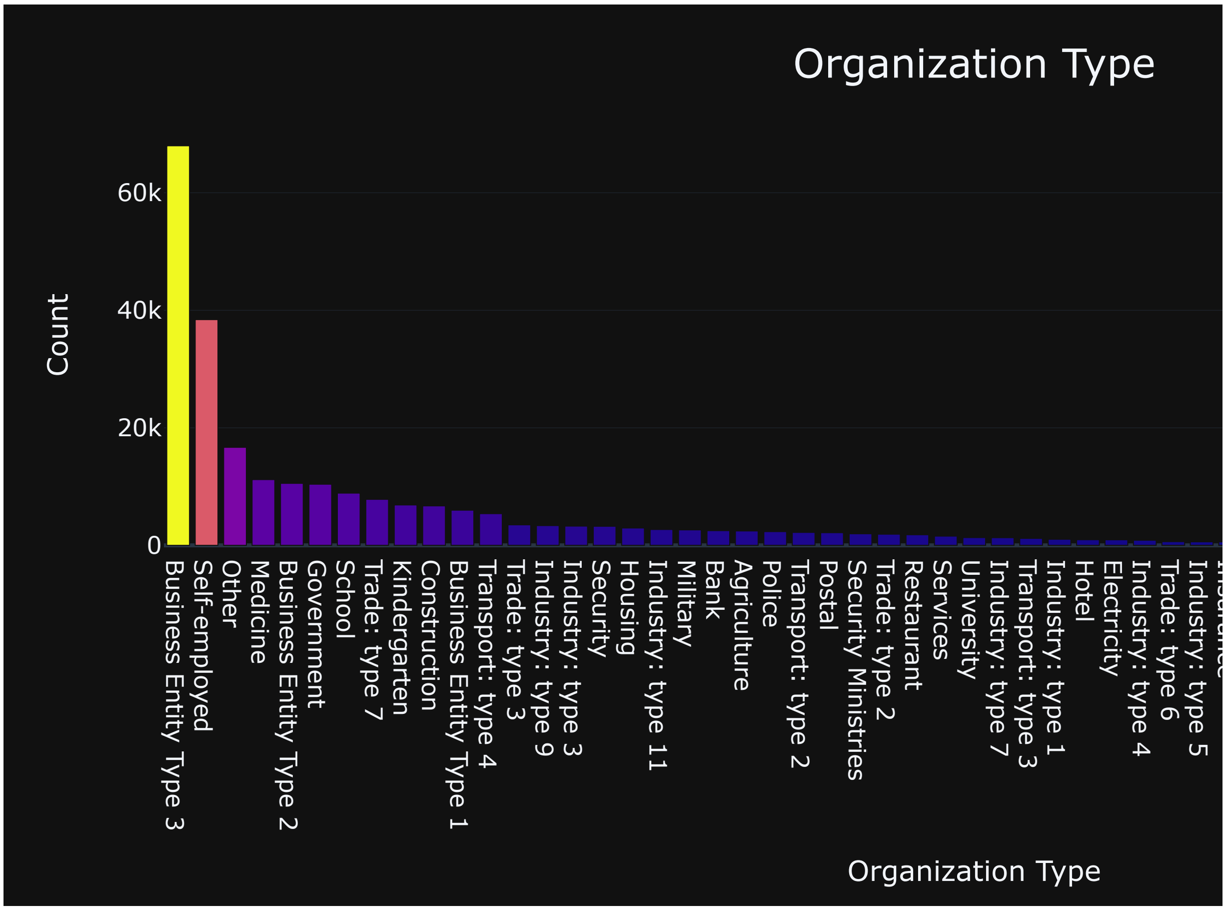
In [58]:
```
app_data.columns
```

Out[58]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY',
       'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE',
       'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE',
       'REGION_POPULATION_RELATIVE', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLIS
H',
       'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
       'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
       'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
       'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY',
       'LIVE_CITY_NOT_WORK_CITY', 'ORGANIZATION_TYPE', 'EXT_SOURCE_2',
       'EXT_SOURCE_3', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIR
CLE',
       'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
       'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_BUREAU_HOUR',
       'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
       'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
       'AMT_REQ_CREDIT_BUREAU_YEAR', 'AMT_INCOME_TOTAL_Range',
       'AMT_CREDIT_Range', 'AMT_ANNUITY_Range', 'AMT_GOODS_PRICE_Range',
       'YEARS_EMPLOYED', 'Client_Age', 'Age Group', 'Work Experience'],
      dtype='object')
```

In [59]:
```
fig1=px.bar(app_data['OCCUPATION_TYPE'].value_counts(),color=app_data['OCCU
fig1.update_traces(textposition='outside',marker_coloraxis=None)
fig1.update_xaxes(title='Occupation Type')
fig1.update_yaxes(title='Count')
fig1.update_layout(
                    title=dict(text = "Occupation Type",x=0.5,y=0.95),
                    title_font_size=20,
                    showlegend=False,
                    height =450,
                  )
fig1.show()
```



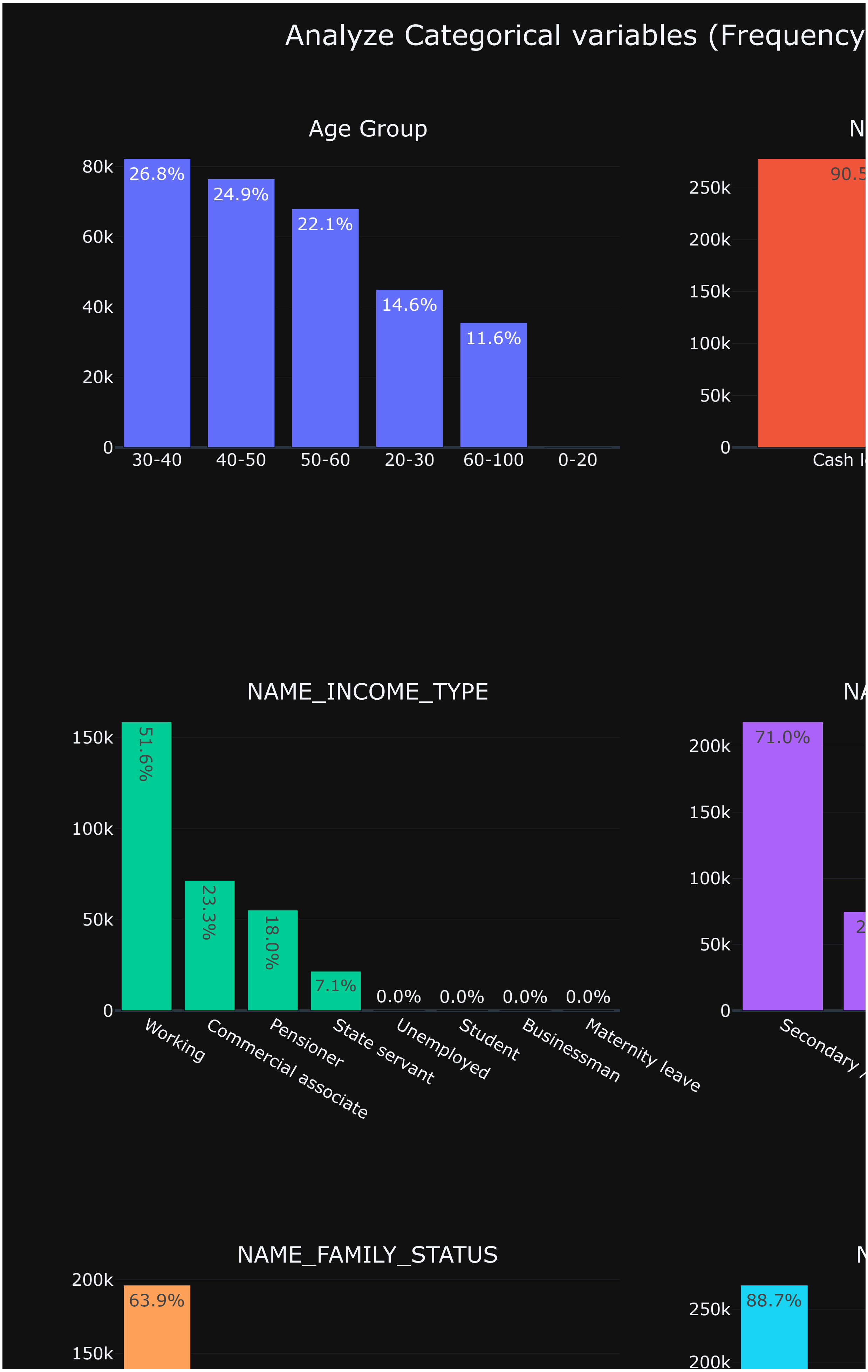
In [60]:
```
fig2=px.bar(app_data['ORGANIZATION_TYPE'].value_counts(),color=app_data['OR
fig2.update_traces(textposition='outside',marker_coloraxis=None)
fig2.update_xaxes(title='Organization Type')
fig2.update_yaxes(title='Count')
fig2.update_layout(
                        title=dict(text = "Organization Type",x=0.5,y=0.95)
                    title_font_size=20,
                    showlegend=False,
                    height =450,
                  )
fig2.show()
```



Insights

- Most People who applied for Loan application are Laborers
- Most People who applied for Loan application belong to either **Business Entity Type3** or **Self-Employed** Organization Type.

In [61]:
```python
cols = ['Age Group','NAME_CONTRACT_TYPE', 'NAME_INCOME_TYPE','NAME_EDUCATIO
        'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE','CODE_GENDER','Work Exper

#Subplot initialization
fig = make_subplots(
                     rows=4,
                     cols=2,
                     subplot_titles=cols,
                     horizontal_spacing=0.1,
                     vertical_spacing=0.13
                   )
# Adding subplots
count=0
for i in range(1,5):
    for j in range(1,3):
        fig.add_trace(go.Bar(x=app_data[cols[count]].value_counts().index,
                             y=app_data[cols[count]].value_counts(),
                             name=cols[count],
                             textposition='auto',
                             text= [str(i) + '%' for i in (app_data[cols[co
                            ),
                      row=i,col=j)
        count+=1
fig.update_layout(
                    title=dict(text = "Analyze Categorical variables (Frequ
                    title_font_size=20,
                    showlegend=False,
                    width = 960,
                    height = 1600,
                  )
fig.show()
```


Analyze Categorical variables (Frequency
Age Group
26.8%
24.9%
22.1%
14.6%
11.6%
80k
60k
40k
20k
0
30-40
40-50
50-60
20-30
60-100
0-20
250k
200k
150k
100k
50k
0
Cash l
NAME_INCOME_TYPE
51.6%
23.3%
18.0%
7.1%
0.0%
0.0%
0.0%
0.0%
150k
100k
50k
0
Working
Commercial associate
Pensioner
State servant
Unemployed
Student
Businessman
Maternity leave
71.0%
200k
150k
100k
50k
0
Secondary
NAME_FAMILY_STATUS
63.9%
200k
150k
88.7%
250k
200k


100k
50k
0
14.8%
9.7%
6.4%
5.2%
Married
Single / not married
Civil marriage
Separated
Widow
Unknown
150k
100k
50k
0
House / apar
CODE_GENDER
200k
150k
100k
50k
0
65.8%
34.2%
F
M
100k
50k
0
54.1%
25.7%
0-5
5-10

In [62]:
```
app_data.nunique().sort_values()
```

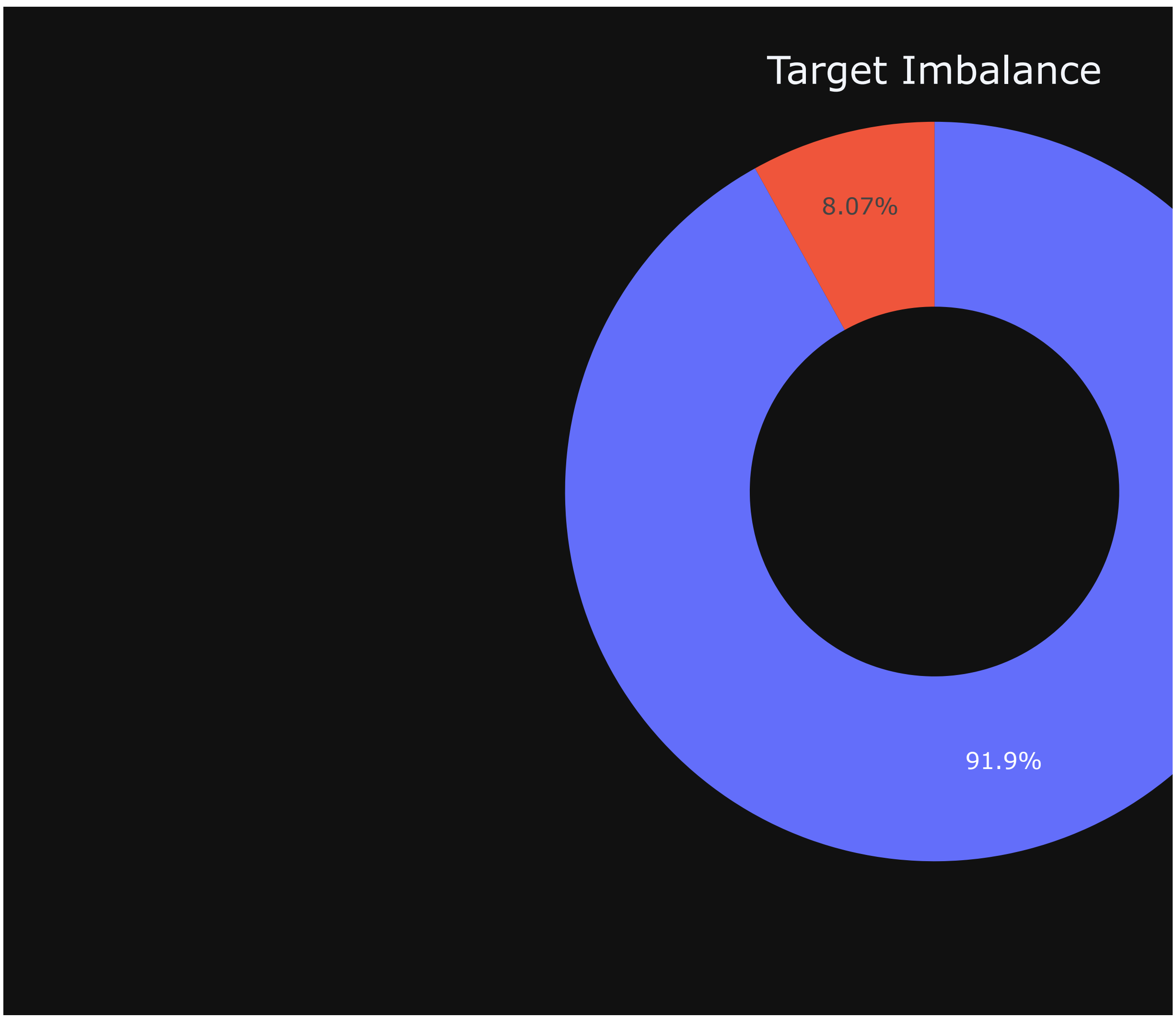
Out[62]:
```
LIVE_REGION_NOT_WORK_REGION          2
REG_REGION_NOT_LIVE_REGION           2
REG_REGION_NOT_WORK_REGION           2
REG_CITY_NOT_LIVE_CITY               2
LIVE_CITY_NOT_WORK_CITY              2
REG_CITY_NOT_WORK_CITY               2
CODE_GENDER                          2
NAME_CONTRACT_TYPE                   2
TARGET                               2
REGION_RATING_CLIENT                 3
REGION_RATING_CLIENT_W_CITY          3
AMT_CREDIT_Range                     5
AMT_INCOME_TOTAL_Range               5
AMT_REQ_CREDIT_BUREAU_HOUR           5
AMT_GOODS_PRICE_Range                5
NAME_EDUCATION_TYPE                  5
Age Group                            5
AMT_ANNUITY_Range                    5
NAME_FAMILY_STATUS                   6
NAME_HOUSING_TYPE                    6
WEEKDAY_APPR_PROCESS_START           7
Work Experience                      7
NAME_TYPE_SUITE                      7
NAME_INCOME_TYPE                     8
AMT_REQ_CREDIT_BUREAU_WEEK           9
AMT_REQ_CREDIT_BUREAU_DAY            9
DEF_60_CNT_SOCIAL_CIRCLE             9
AMT_REQ_CREDIT_BUREAU_QRT           10
DEF_30_CNT_SOCIAL_CIRCLE            10
CNT_CHILDREN                        11
CNT_FAM_MEMBERS                     12
OCCUPATION_TYPE                     18
HOUR_APPR_PROCESS_START             24
AMT_REQ_CREDIT_BUREAU_MON           24
AMT_REQ_CREDIT_BUREAU_YEAR          25
OBS_60_CNT_SOCIAL_CIRCLE            33
OBS_30_CNT_SOCIAL_CIRCLE            33
ORGANIZATION_TYPE                   57
REGION_POPULATION_RELATIVE          81
EXT_SOURCE_3                       814
AMT_GOODS_PRICE                   1002
AMT_INCOME_TOTAL                  2547
DAYS_LAST_PHONE_CHANGE            3773
AMT_CREDIT                        5603
DAYS_ID_PUBLISH                   6168
YEARS_EMPLOYED                   12573
AMT_ANNUITY                      13671
DAYS_REGISTRATION                15688
Client_Age                       17460
EXT_SOURCE_2                    119830
SK_ID_CURR                      307500
dtype: int64
```

Checking Imbalance

In [63]:
```
app_data['TARGET'].value_counts(normalize=True)
```

Out[63]:
```
0    0.919275
1    0.080725
Name: TARGET, dtype: float64
```

In [64]:
```
fig=px.pie(values=app_data['TARGET'].value_counts(normalize=True),
           names=app_data['TARGET'].value_counts(normalize=True).index,
           hole = 0.5
          )
fig.update_layout(
                  title=dict(text = "Target Imbalance",x=0.5,y=0.95),
                  title_font_size=20,
                  showlegend=False
                 )
fig.show()
```



In [65]:
```
app_target0 = app_data.loc[app_data.TARGET == 0]
app_target1 = app_data.loc[app_data.TARGET == 1]
```

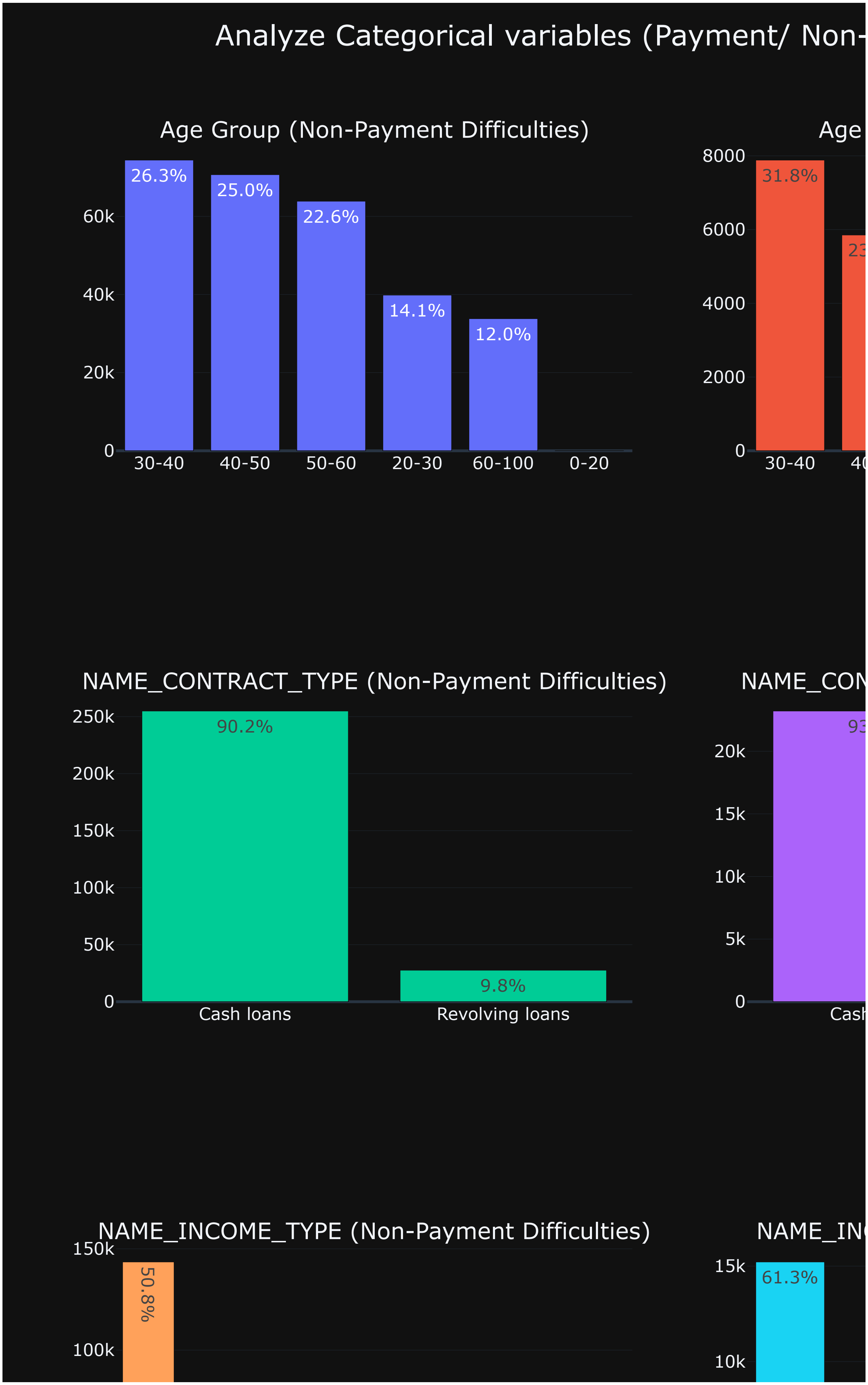
In [66]:
```
app_target0.shape
```

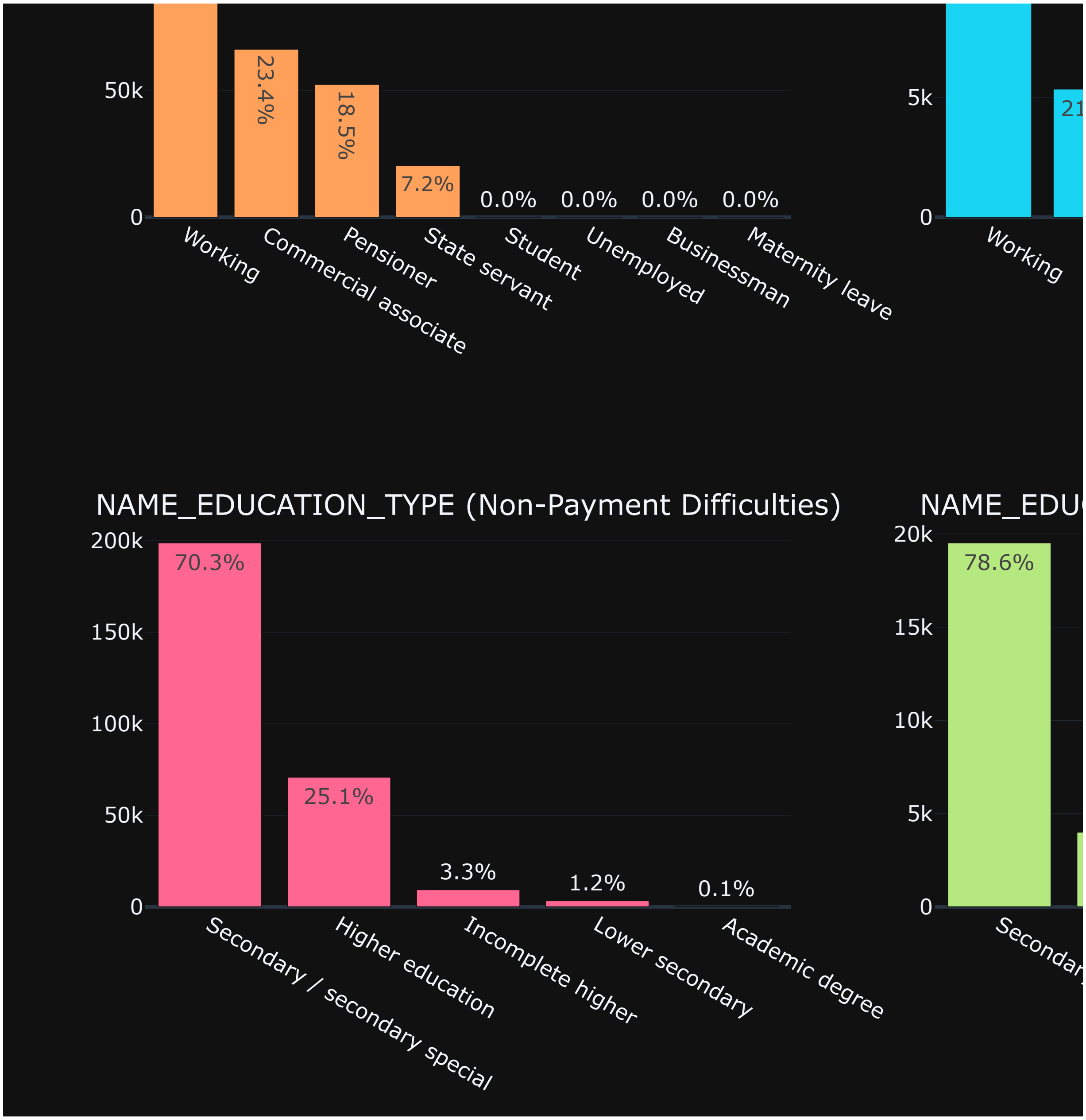
Out[66]:
```
(282677, 51)
```

In [67]:
```
app_target1.shape
```

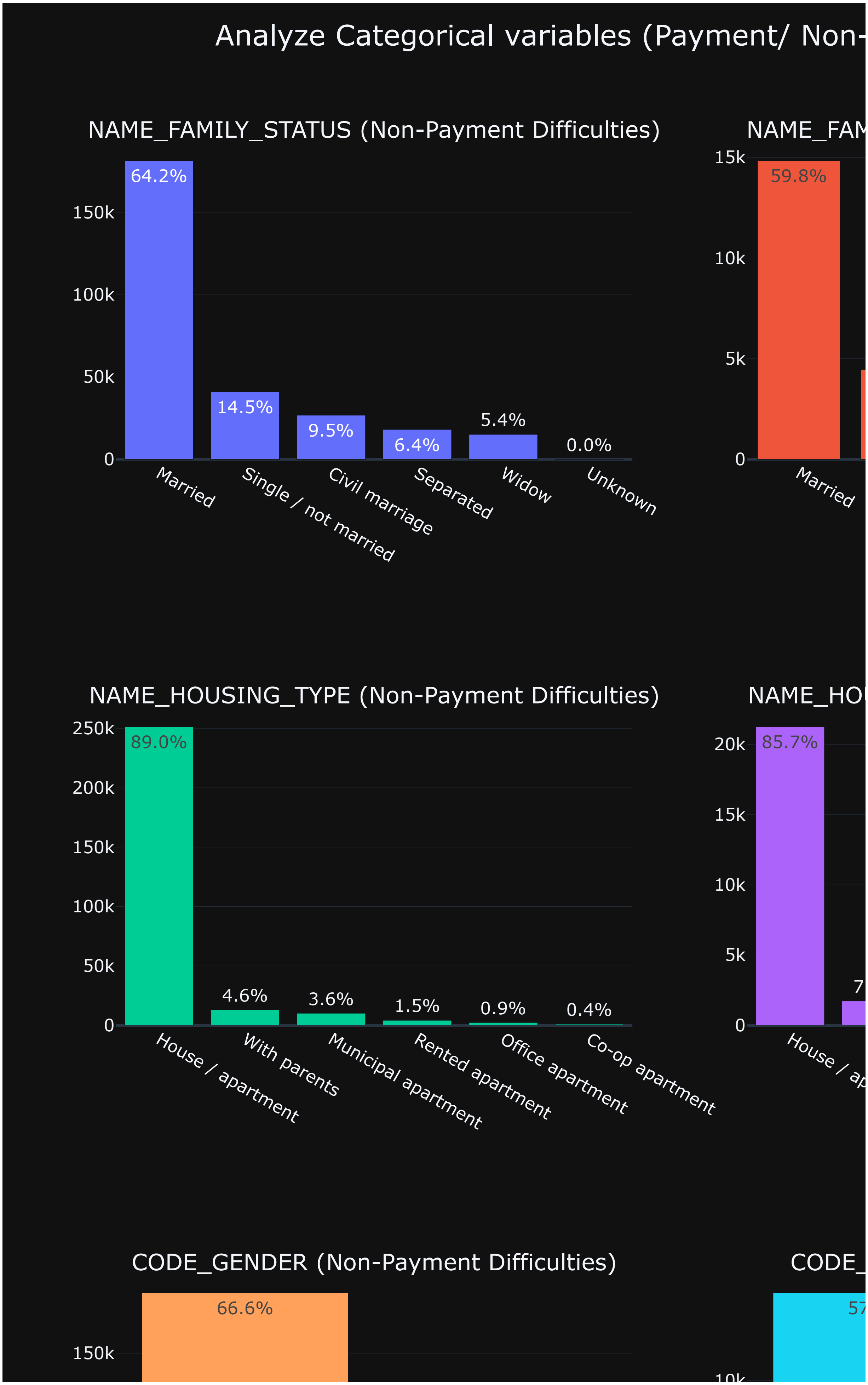
Out[67]:
```
(24823, 51)
```

In [68]:
```
ls = ['Age Group','NAME_CONTRACT_TYPE', 'NAME_INCOME_TYPE','NAME_EDUCATION_'

tle = [None]*(2*len(cols))
tle[::2]=[i+' (Non-Payment Difficulties)' for i in cols]
tle[1::2]=[i+' (Payment Difficulties)' for i in cols]

'ubplot initialization
g = make_subplots(
                    rows=4,
                    cols=2,
                    subplot_titles=title,
                  )
Adding subplots
unt=0
r i in range(1,5):
  for j in range(1,3):
      if j==1:
          fig.add_trace(go.Bar(x=app_target0[cols[count]].value_counts().in
                               y=app_target0[cols[count]].value_counts(),
                               name=cols[count],
                               textposition='auto',
                               text= [str(i) + '%' for i in (app_target0[cols[c
                              ),
                     row=i,col=j)
      else:
          fig.add_trace(go.Bar(x=app_target1[cols[count]].value_counts().in
                               y=app_target1[cols[count]].value_counts(),
                               name=cols[count],
                               textposition='auto',
                               text= [str(i) + '%' for i in (app_target1[cols[c
                              ),
                     row=i,col=j)
          count+=1
g.update_layout(
                  title=dict(text = "Analyze Categorical variables (Payment/
                  title_font_size=20,
                  showlegend=False,
                  height = 1600,
                )
g.show()
```

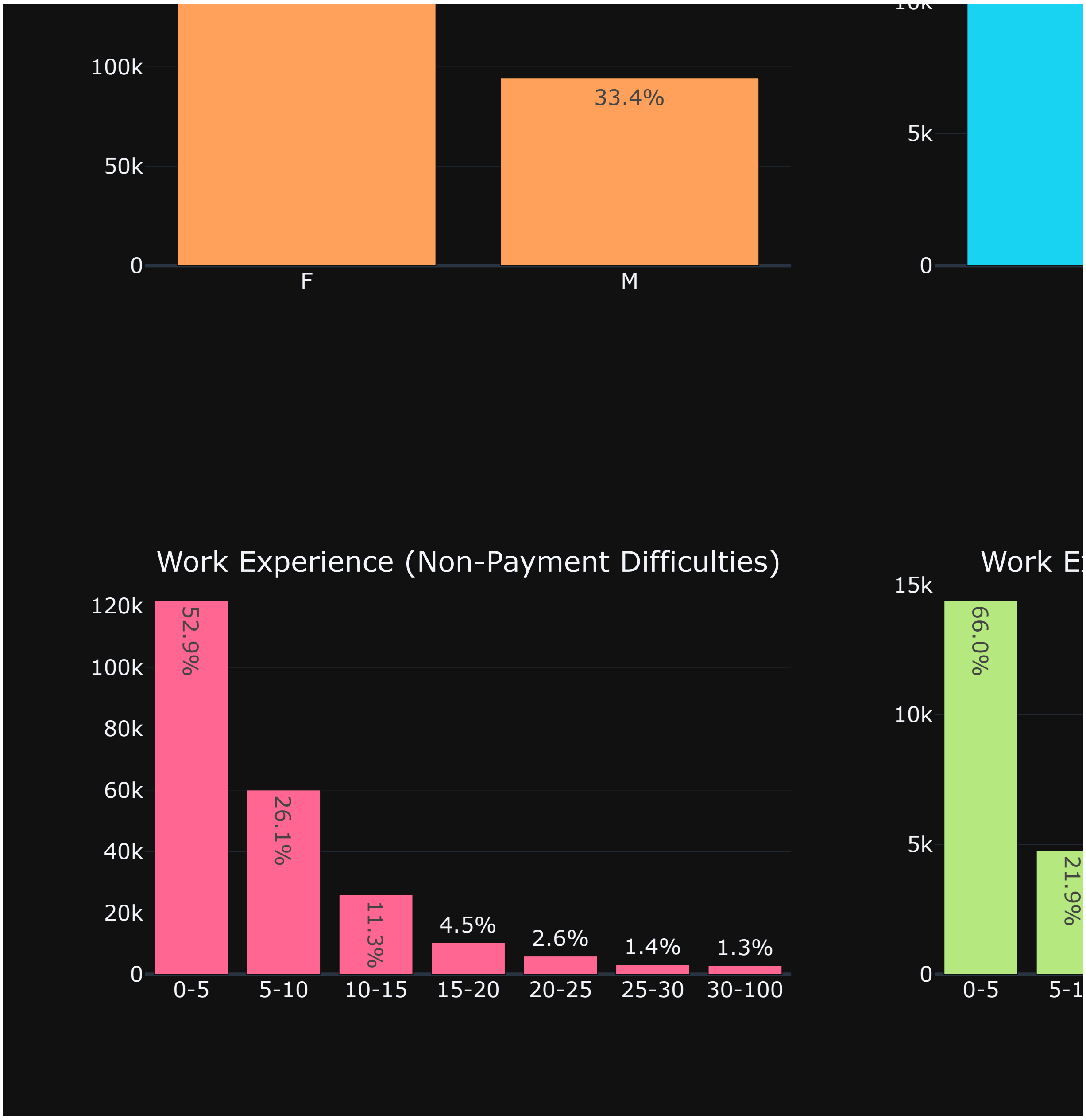

Analyze Categorical variables (Payment/ Non-
Age Group (Non-Payment Difficulties)
26.3%
25.0%
22.6%
14.1%
12.0%
60k
40k
20k
0
30-40
40-50
50-60
20-30
60-100
0-20
Age
8000
6000
4000
2000
0
31.8%
30-40
NAME_CONTRACT_TYPE (Non-Payment Difficulties)
250k
200k
150k
100k
50k
0
90.2%
9.8%
Cash loans
Revolving loans
20k
15k
10k
5k
0
NAME_INCOME_TYPE (Non-Payment Difficulties)
150k
100k
50.8%
15k
10k
61.3%


50k
0
23.4%
18.5%
7.2%
0.0%
0.0%
0.0%
0.0%
Working
Commercial associate
Pensioner
State servant
Student
Unemployed
Businessman
Maternity leave
5k
0
21
Working
NAME_EDUCATION_TYPE (Non-Payment Difficulties)
200k
150k
100k
50k
0
70.3%
25.1%
3.3%
1.2%
0.1%
Secondary / secondary special
Higher education
Incomplete higher
Lower secondary
Academic degree
NAME_EDU
20k
15k
10k
5k
0
78.6%

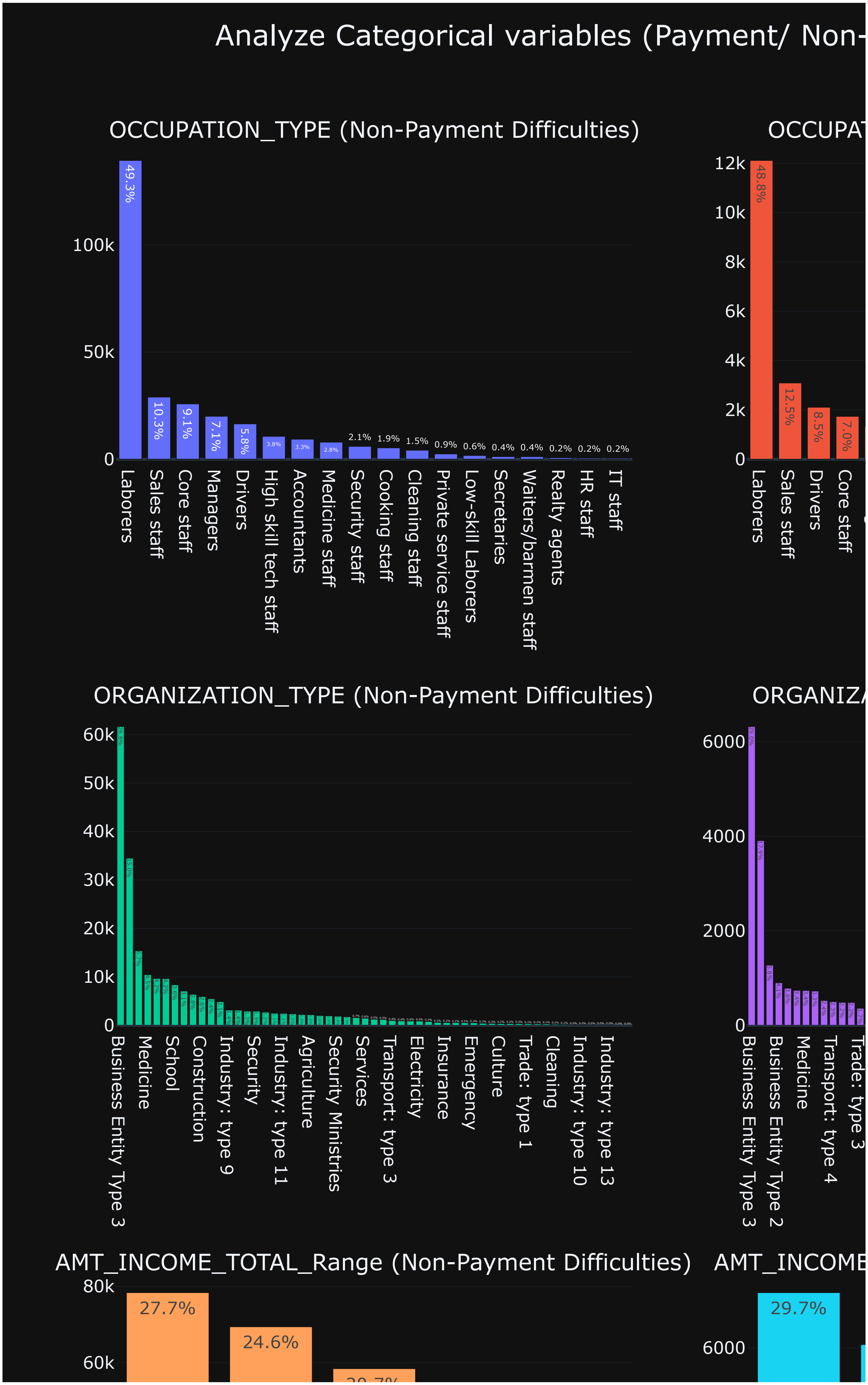
In [69]:
```
cols = ['NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE','CODE_GENDER','Work Exper

title = [None]*(2*len(cols))
title[::2]=[i+' (Non-Payment Difficulties)' for i in cols]
title[1::2]=[i+' (Payment Difficulties)' for i in cols]

#Subplot initialization
fig = make_subplots(
                    rows=4,
                    cols=2,
                    subplot_titles=title,
                   )
# Adding subplots
count=0
for i in range(1,5):
    for j in range(1,3):
        if j==1:
            fig.add_trace(go.Bar(x=app_target0[cols[count]].value_counts().
                                 y=app_target0[cols[count]].value_counts(),
                                 name=cols[count],
                                 textposition='auto',
                                 text= [str(i) + '%' for i in (app_target0[cols
                                ),
                      row=i,col=j)
        else:
            fig.add_trace(go.Bar(x=app_target1[cols[count]].value_counts().
                                 y=app_target1[cols[count]].value_counts(),
                                 name=cols[count],
                                 textposition='auto',
                                 text= [str(i) + '%' for i in (app_target1[cols
                                ),
                      row=i,col=j)
            count+=1
fig.update_layout(
                    title=dict(text = "Analyze Categorical variables (Payme
                    title_font_size=20,
                    showlegend=False,
                    height = 1600,
                  )
fig.show()
```

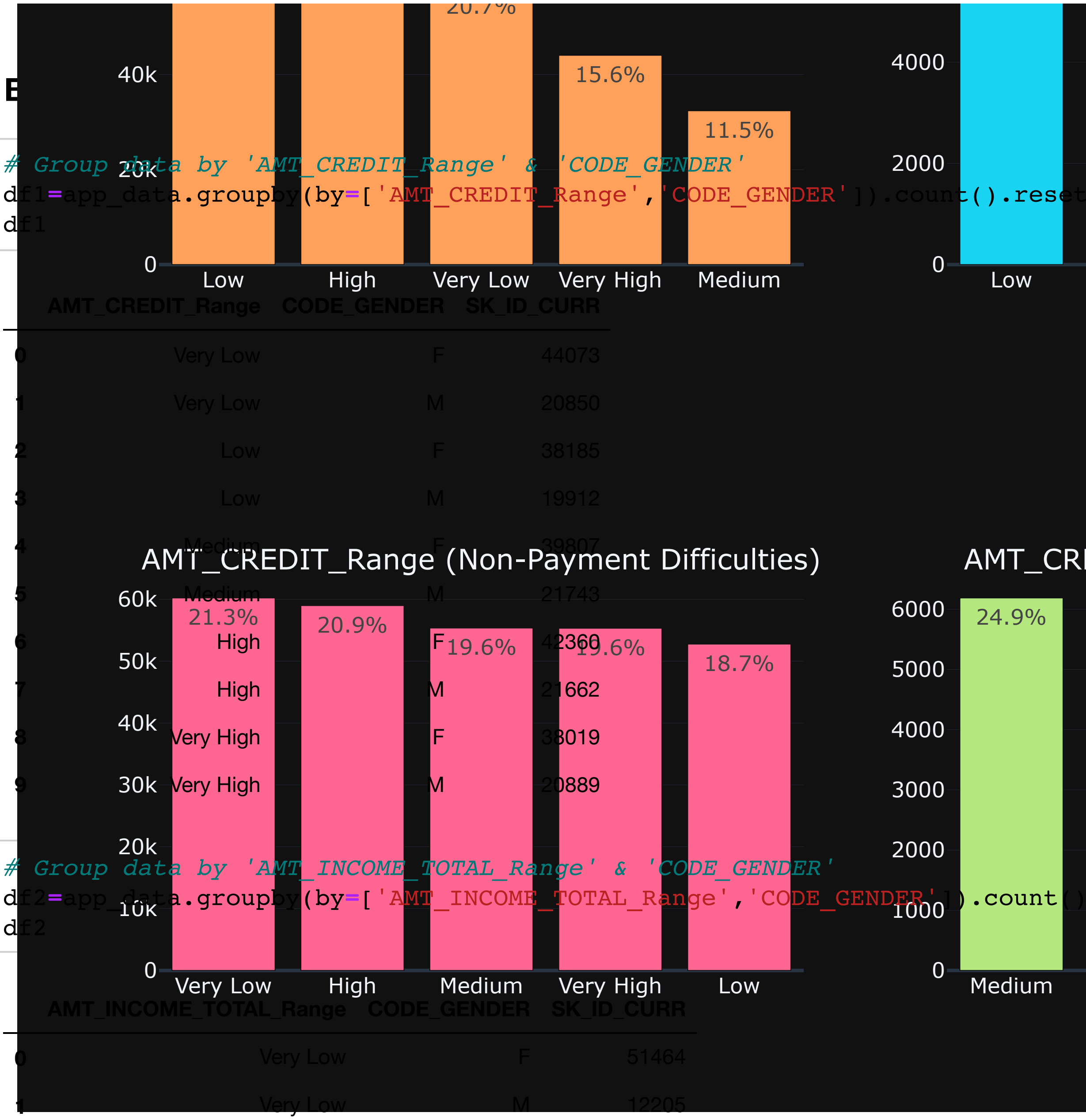


Analyze Categorical variables (Payment/ Non-
NAME_FAMILY_STATUS (Non-Payment Difficulties)
64.2%
14.5%
9.5%
6.4%
5.4%
0.0%
0
50k
100k
150k
Married
Single / not married
Civil marriage
Separated
Widow
Unknown
NAME_FAM
15k
10k
5k
0
59.8%
Married
NAME_HOUSING_TYPE (Non-Payment Difficulties)
250k
200k
150k
100k
50k
0
89.0%
4.6%
3.6%
1.5%
0.9%
0.4%
House / apartment
With parents
Municipal apartment
Rented apartment
Office apartment
Co-op apartment
NAME_HOU
20k
15k
10k
5k
0
85.7%
House / ap
CODE_GENDER (Non-Payment Difficulties)
66.6%
150k
CODE_


100k
50k
0
33.4%
F
M
10k
5k
0
Work Experience (Non-Payment Difficulties)
120k
100k
80k
60k
40k
20k
0
52.9%
26.1%
11.3%
4.5%
2.6%
1.4%
1.3%
0-5
5-10
10-15
15-20
20-25
25-30
30-100
Work Ex
15k
10k
5k
0
66.0%
21.9%
0-5
5-1

In [70]:
```
= ['OCCUPATION_TYPE', 'ORGANIZATION_TYPE' ,'AMT_INCOME_TOTAL_Range','AMT_CRE

 = [None]*(2*len(cols))
[::2]=[i+' (Non-Payment Difficulties)' for i in cols]
[1::2]=[i+' (Payment Difficulties)' for i in cols]

lot initialization
 make_subplots(
                rows=4,
                cols=2,
                subplot_titles=title,
               )
ing subplots
=0
 in range(1,5):
or j in range(1,3):
   if j==1:
       fig.add_trace(go.Bar(x=app_target0[cols[count]].value_counts().index,
                        y=app_target0[cols[count]].value_counts(),
                        name=cols[count],
                        textposition='auto',
                        text= [str(i) + '%' for i in (app_target0[cols[count
                       ),
                 row=i,col=j)
   else:
       fig.add_trace(go.Bar(x=app_target1[cols[count]].value_counts().index,
                        y=app_target1[cols[count]].value_counts(),
                        name=cols[count],
                        textposition='auto',
                        text= [str(i) + '%' for i in (app_target1[cols[count
                       ),
                 row=i,col=j)
       count+=1
pdate_layout(
               title=dict(text = "Analyze Categorical variables (Payment/ No
               title_font_size=20,
               showlegend=False,
               height = 1600,
              )
how()
```


Analyze Categorical variables (Payment/ Non-
OCCUPATION_TYPE (Non-Payment Difficulties)
100k
50k
0
49.3%
10.3%
9.1%
7.1%
5.8%
3.8%
3.3%
2.8%
2.1%
1.9%
1.5%
0.9%
0.6%
0.4%
0.4%
0.2%
0.2%
0.2%
Laborers
Sales staff
Core staff
Managers
Drivers
High skill tech staff
Accountants
Medicine staff
Security staff
Cooking staff
Cleaning staff
Private service staff
Low-skill Laborers
Secretaries
Waiters/barmen staff
Realty agents
HR staff
IT staff
OCCUPAT
12k
10k
8k
6k
4k
2k
0
48.8%
12.5%
8.5%
7.0%
Laborers
Sales staff
Drivers
Core staff
ORGANIZATION_TYPE (Non-Payment Difficulties)
60k
50k
40k
30k
20k
10k
0
Business Entity Type 3
Medicine
School
Construction
Industry: type 9
Security
Industry: type 11
Agriculture
Security Ministries
Services
Transport: type 3
Electricity
Insurance
Emergency
Culture
Trade: type 1
Cleaning
Industry: type 10
Industry: type 13
ORGANIZ
6000
4000
2000
0
Business Entity Type 3
Business Entity Type 2
Medicine
Transport: type 4
Trade: type 3
AMT_INCOME_TOTAL_Range (Non-Payment Difficulties)
80k
60k
27.7%
24.6%
AMT_INCOME
6000
29.7%



In [71]:
```python
# Group data by 'AMT_CREDIT_Range' & 'CODE_GENDER'
df1=app_data.groupby(by=['AMT_CREDIT_Range','CODE_GENDER']).count().reset_i
df1
```

Out[71]:

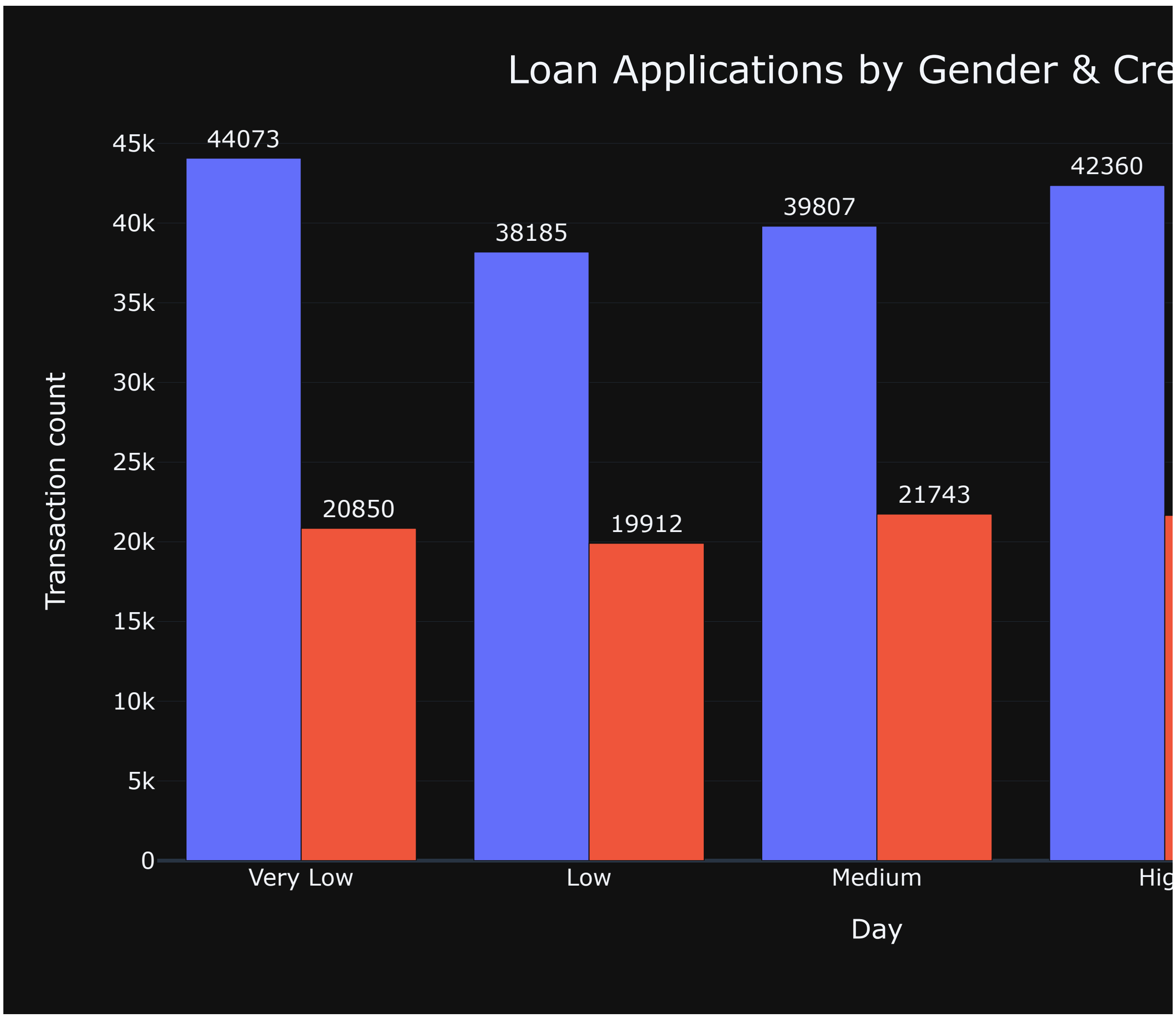
	AMT_CREDIT_Range	CODE_GENDER	SK_ID_CURR
0	Very Low	F	44073
1	Very Low	M	20850
2	Low	F	38185
3	Low	M	19912
4	Medium	F	39807
5	Medium	M	21743
6	High	F	42360
7	High	M	21662
8	Very High	F	38019
9	Very High	M	20889

In [72]:
```python
# Group data by 'AMT_INCOME_TOTAL_Range' & 'CODE_GENDER'
df2=app_data.groupby(by=['AMT_INCOME_TOTAL_Range','CODE_GENDER']).count().r
df2
```

Out[72]:

	AMT_INCOME_TOTAL_Range	CODE_GENDER	SK_ID_CURR
0	Very Low	F	51464
1	Very Low	M	12205
2	Low	F	59967
3	Low	M	25787
4	Medium	F	23184
5	Medium	M	12269
6	High	F	43441
7	High	M	32070
8	Very High	F	24388
9	Very High	M	22725

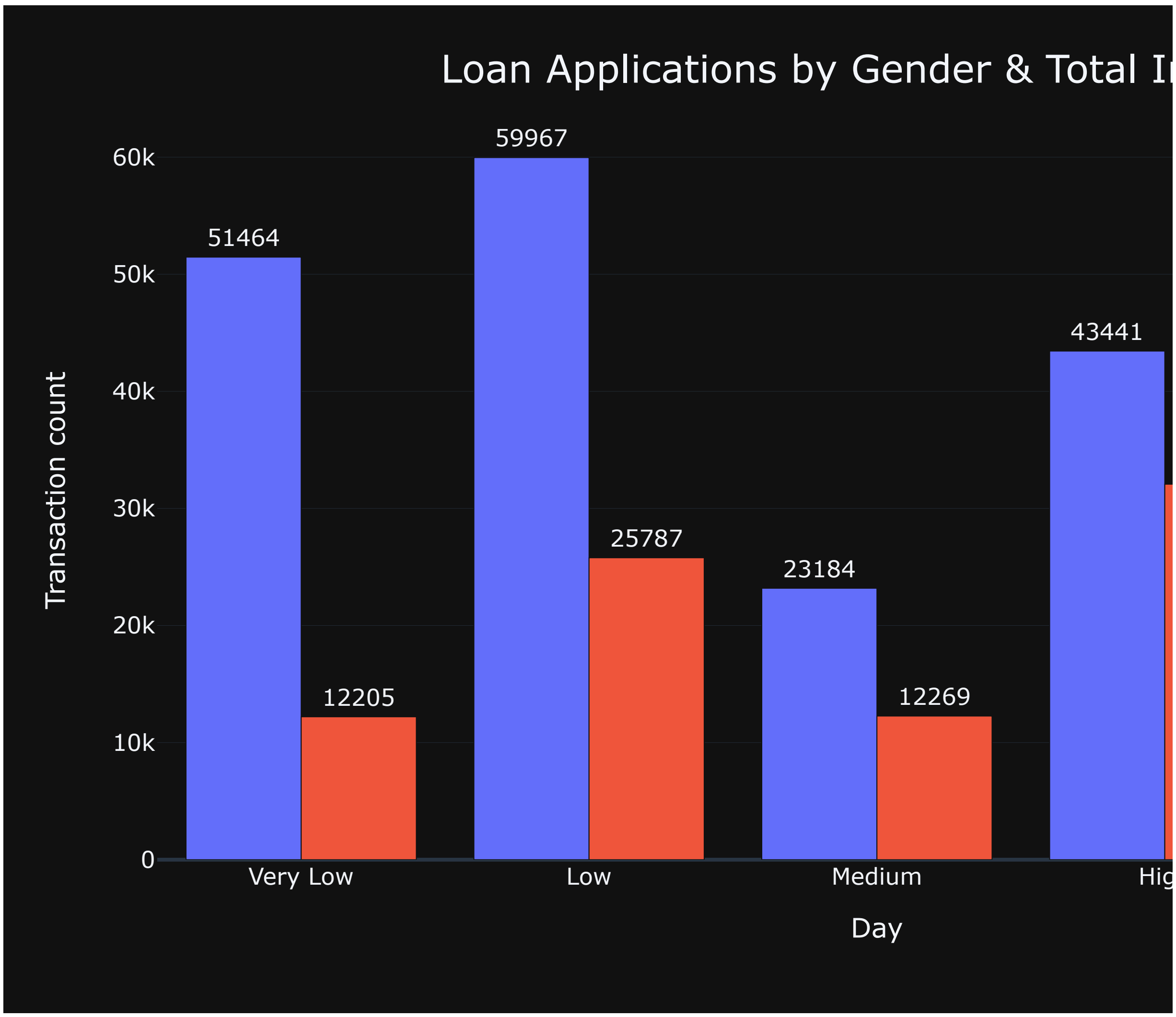
In [73]:
```
fig1=px.bar(data_frame=df1,
        x='AMT_CREDIT_Range',
        y='SK_ID_CURR',color='CODE_GENDER',
        barmode='group',
        text='SK_ID_CURR'
       )
fig1.update_traces(textposition='outside')
fig1.update_xaxes(title='Day')
fig1.update_yaxes(title='Transaction count')
fig1.update_layout(
                    title=dict(text = "Loan Applications by Gender & Credit
                    title_font_size=20,
                  )
fig1.show()
```



Insights

- Females are mostly applying for **Very Low** credit loans.
- Males are applying for **Medium** & **High** credit loans.

In [74]:
```
fig2=px.bar(data_frame=df2,
       x='AMT_INCOME_TOTAL_Range',
       y='SK_ID_CURR',color='CODE_GENDER',
       barmode='group',
       text='SK_ID_CURR'
      )
fig2.update_traces(textposition='outside')
fig2.update_xaxes(title='Day')
fig2.update_yaxes(title='Transaction count')
fig2.update_layout(
                   title=dict(text = "Loan Applications by Gender & Total
                   title_font_size=20,
                  )
fig2.show()
```

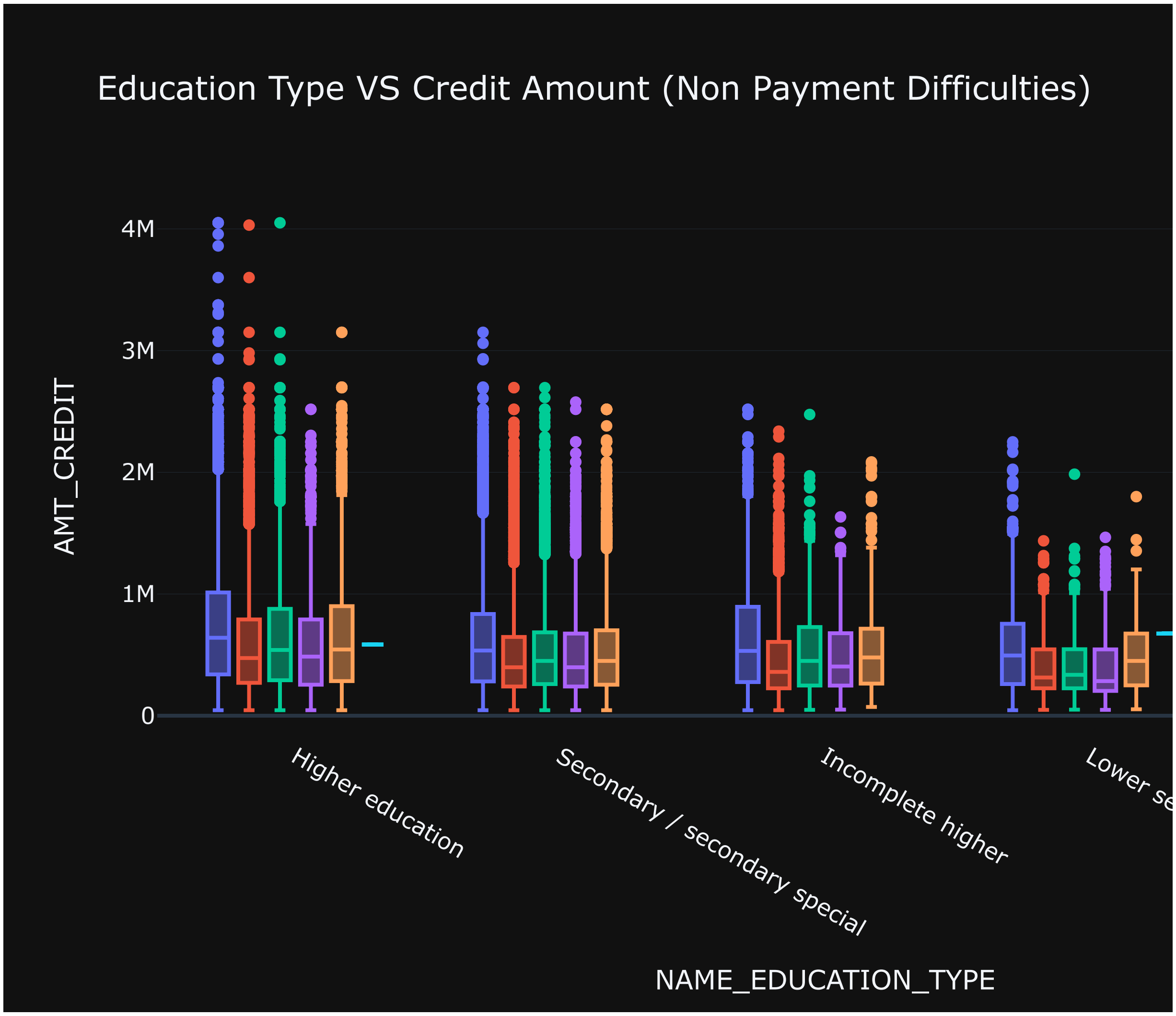


Insights

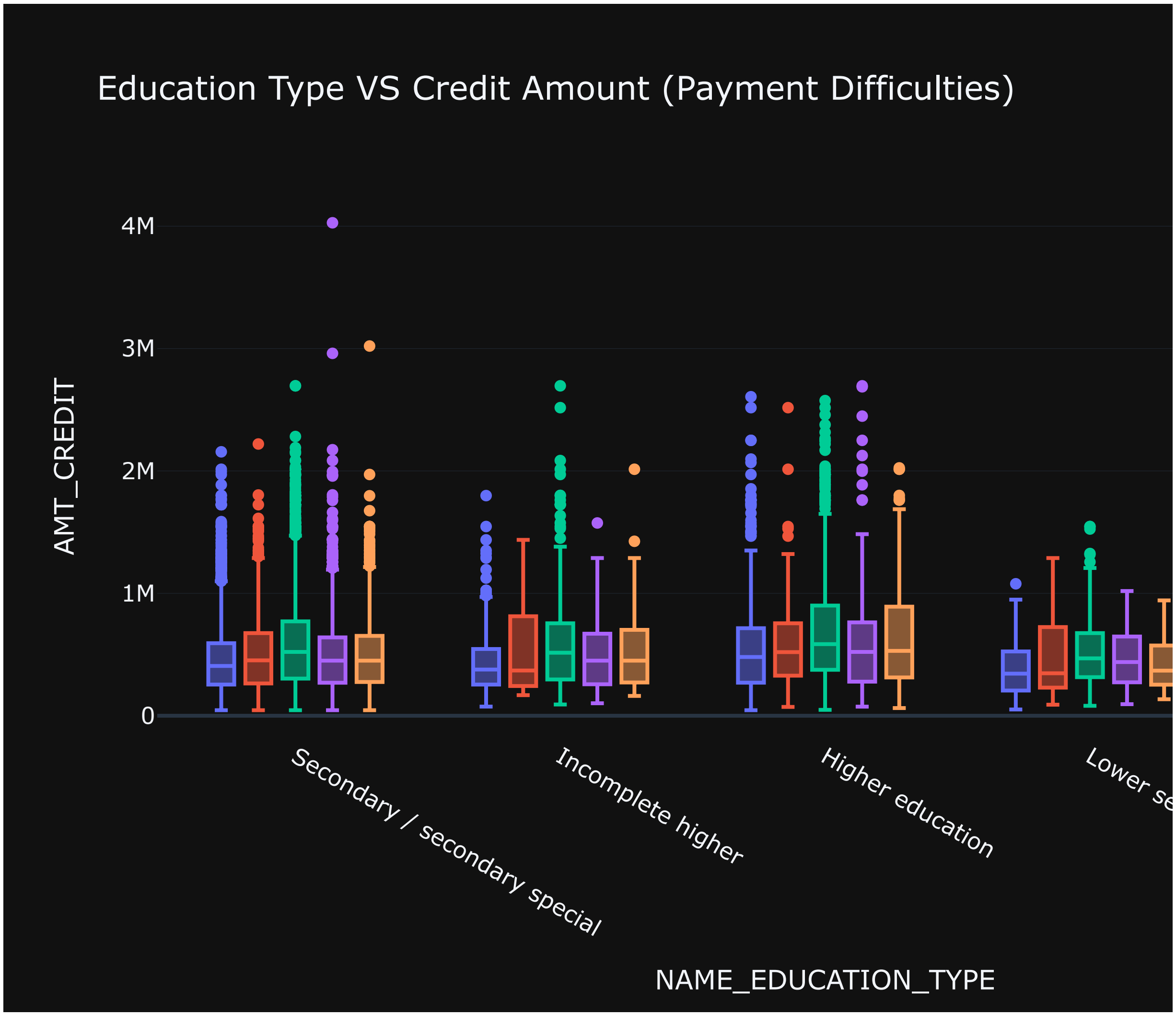
- Females with **Low** & **Very Low** total income have applied the most for the loan.

Education Type VS Credit Amount (Payment / Non Payment Difficulties)

In [75]:
```
fig = px.box(app_target0, x="NAME_EDUCATION_TYPE", y="AMT_CREDIT", color='N
             title="Education Type VS Credit Amount (Non Payment Difficulti
fig.show()
```

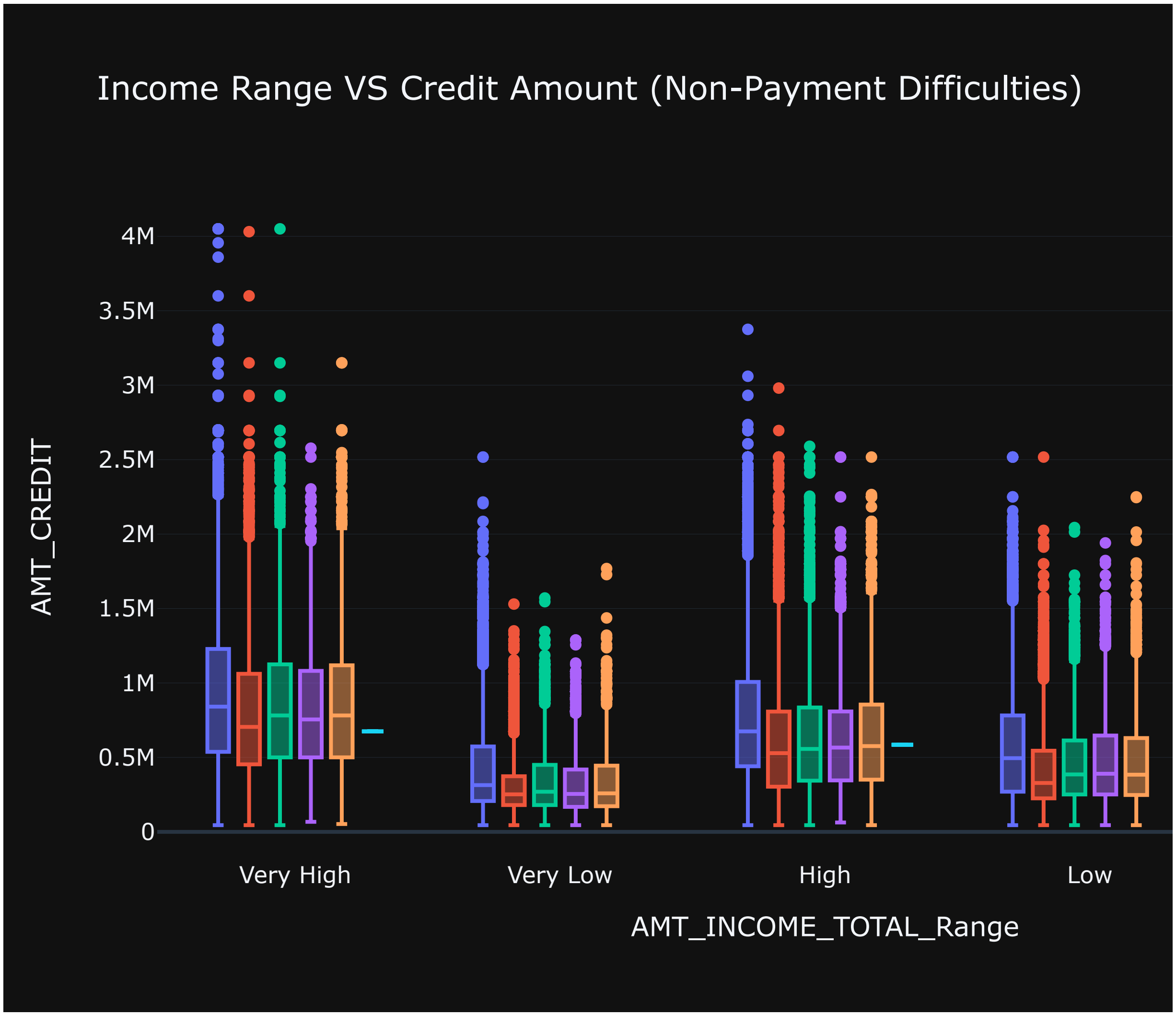


In [76]:
```
fig = px.box(app_target1, x="NAME_EDUCATION_TYPE", y="AMT_CREDIT", color='N
             title="Education Type VS Credit Amount (Payment Difficulties)"
fig.show()
```

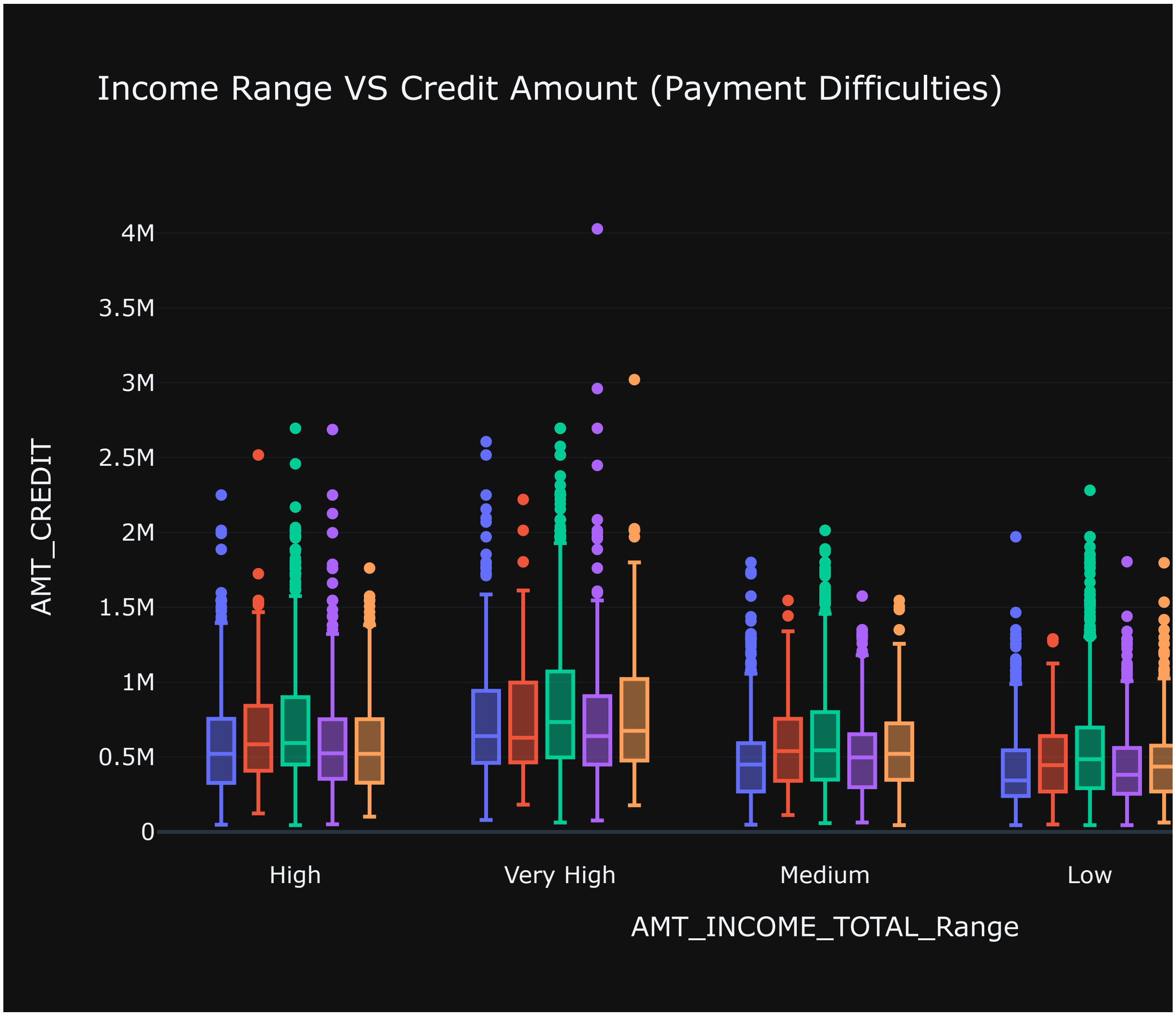


Income VS Credit Amount (Payment / Non Payment Difficulties)

In [77]:
```
fig = px.box(app_target0, x="AMT_INCOME_TOTAL_Range", y="AMT_CREDIT", color
             title="Income Range VS Credit Amount (Non-Payment Difficulties
fig.show()
```

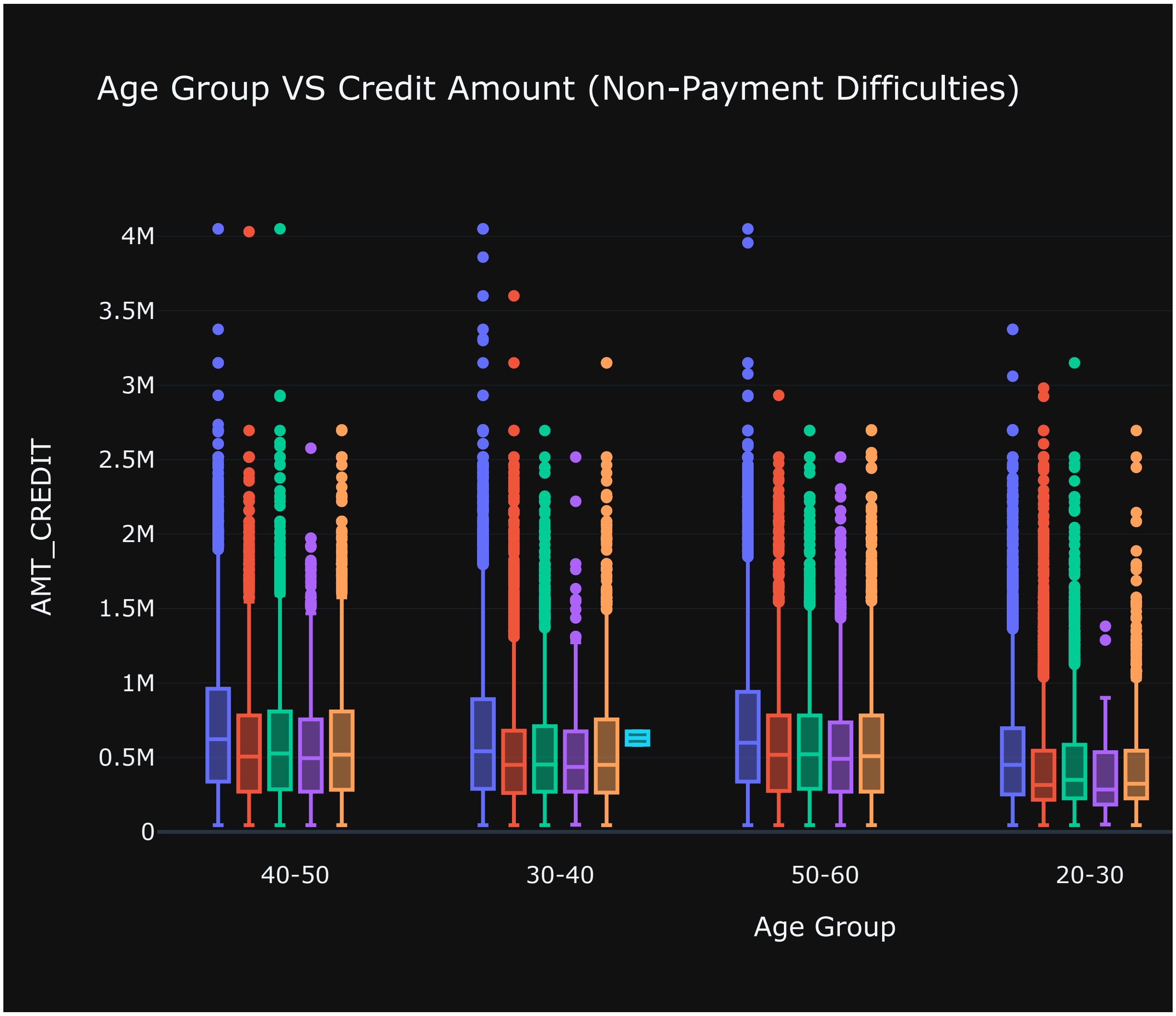


In [78]:
```
fig = px.box(app_target1, x="AMT_INCOME_TOTAL_Range", y="AMT_CREDIT", color
             title="Income Range VS Credit Amount (Payment Difficulties)")
fig.show()
```

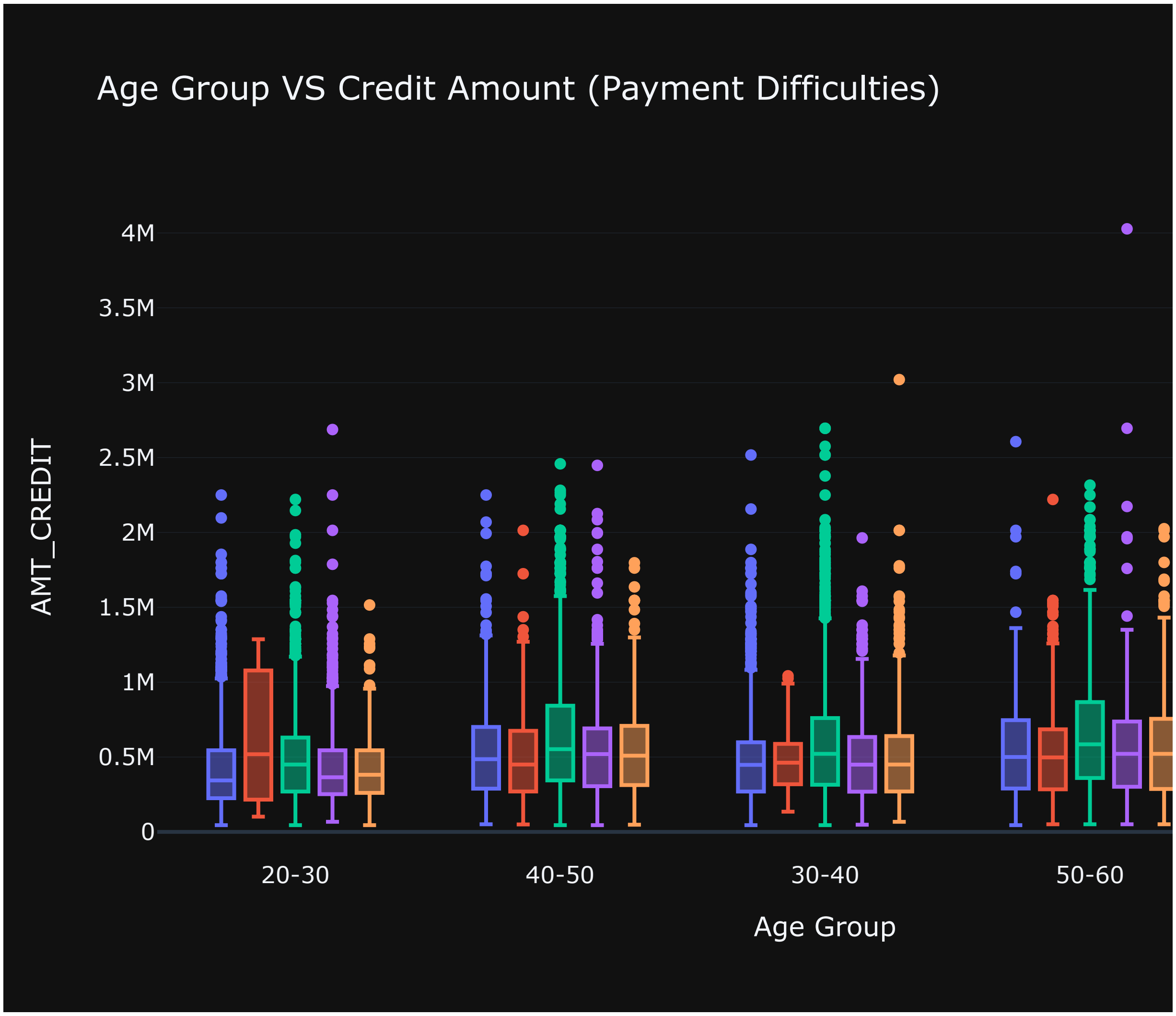


Age Group VS Credit Amount (Payment / Non Payment Difficulties)

In [79]:
```
fig = px.box(app_target0, x="Age Group", y="AMT_CREDIT", color='NAME_FAMILY
             title="Age Group VS Credit Amount (Non-Payment Difficulties)")
fig.show()
```

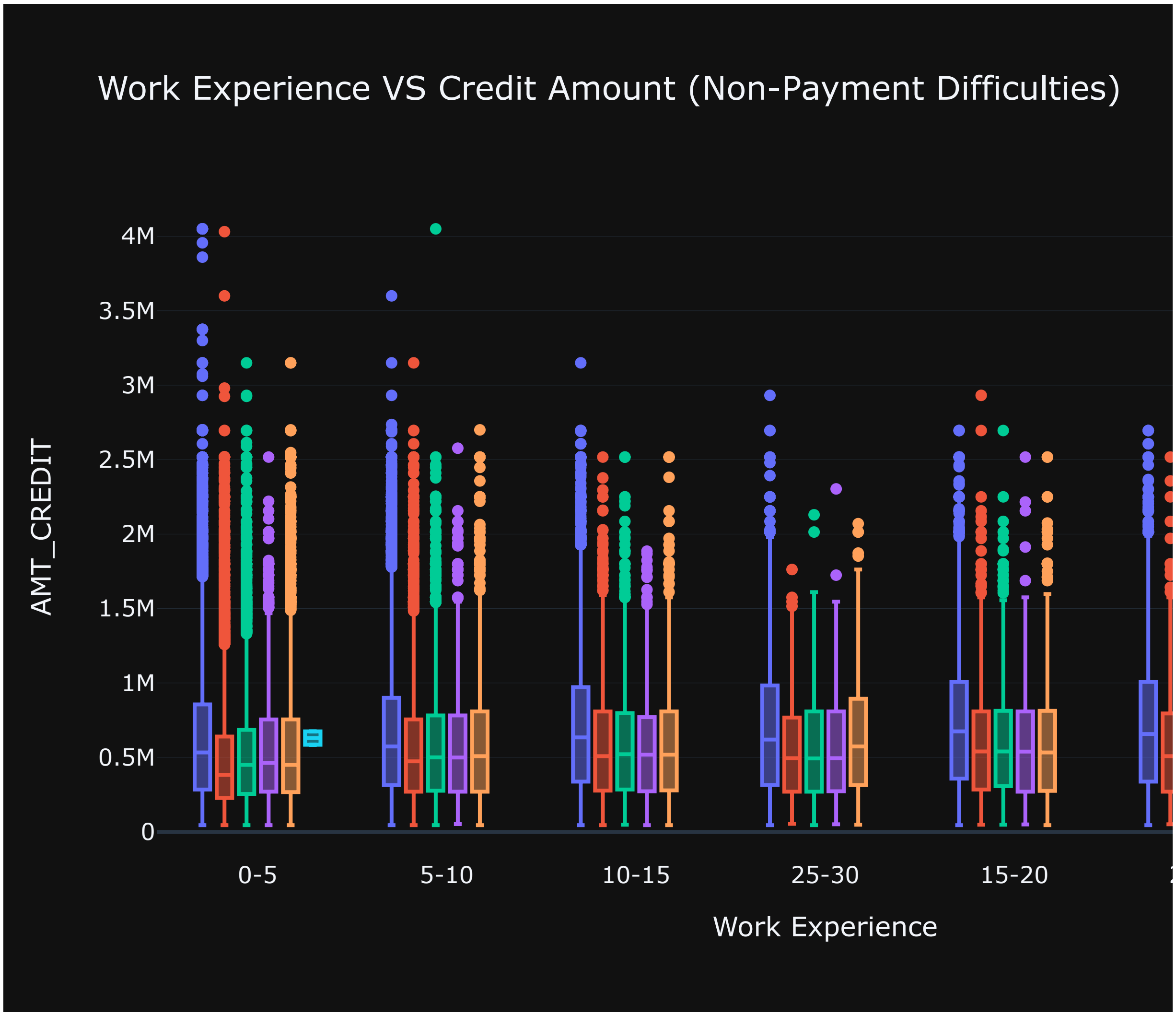


In [80]:
```
fig = px.box(app_target1, x="Age Group", y="AMT_CREDIT", color='NAME_FAMILY
             title="Age Group VS Credit Amount (Payment Difficulties)")
fig.show()
```

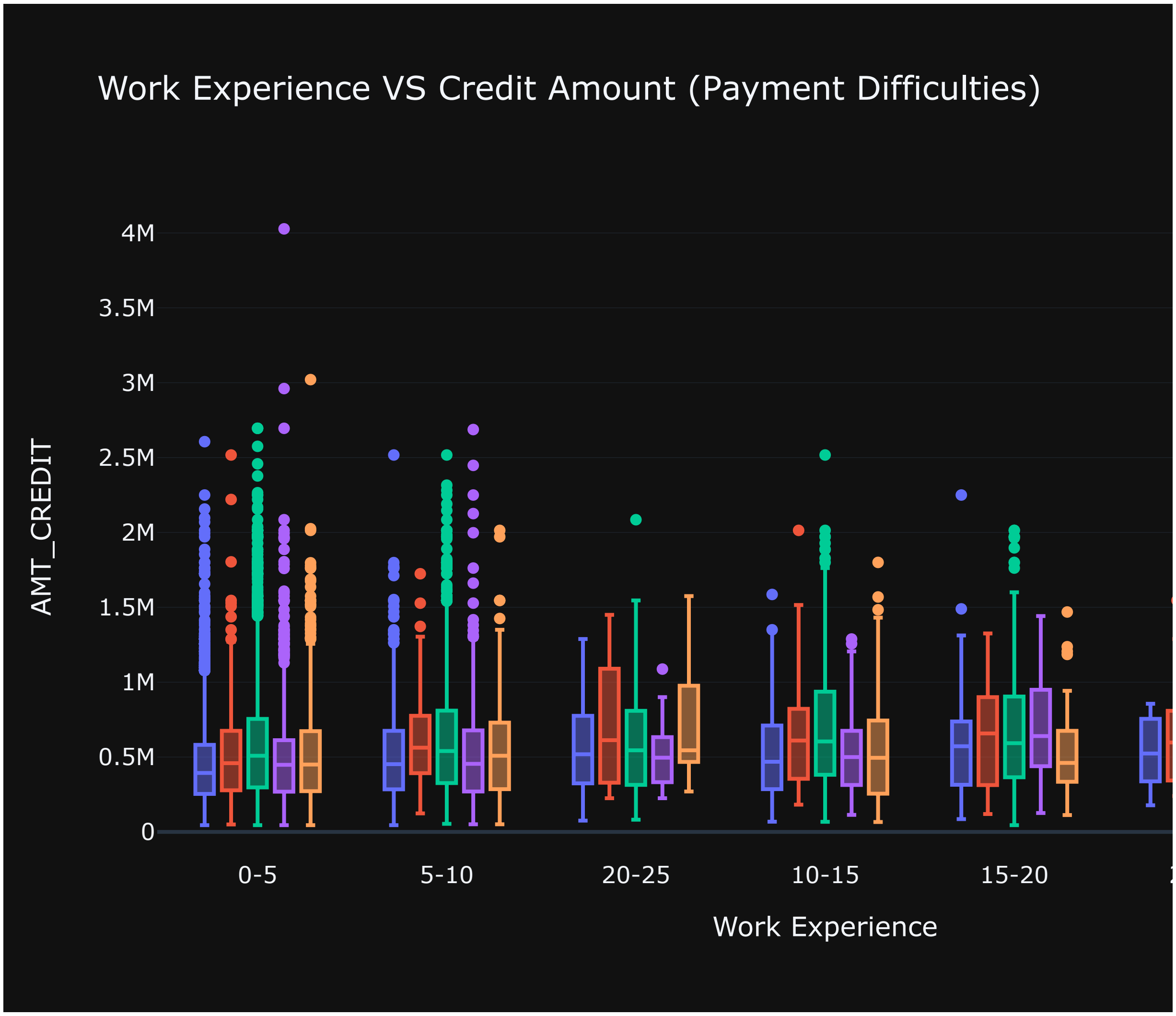


Work Experience VS Credit Amount (Payment / Non Payment Difficulties)

In [81]:
```
fig = px.box(app_target0, x="Work Experience", y="AMT_CREDIT", color='NAME_
             title="Work Experience VS Credit Amount (Non-Payment Difficult
fig.show()
```

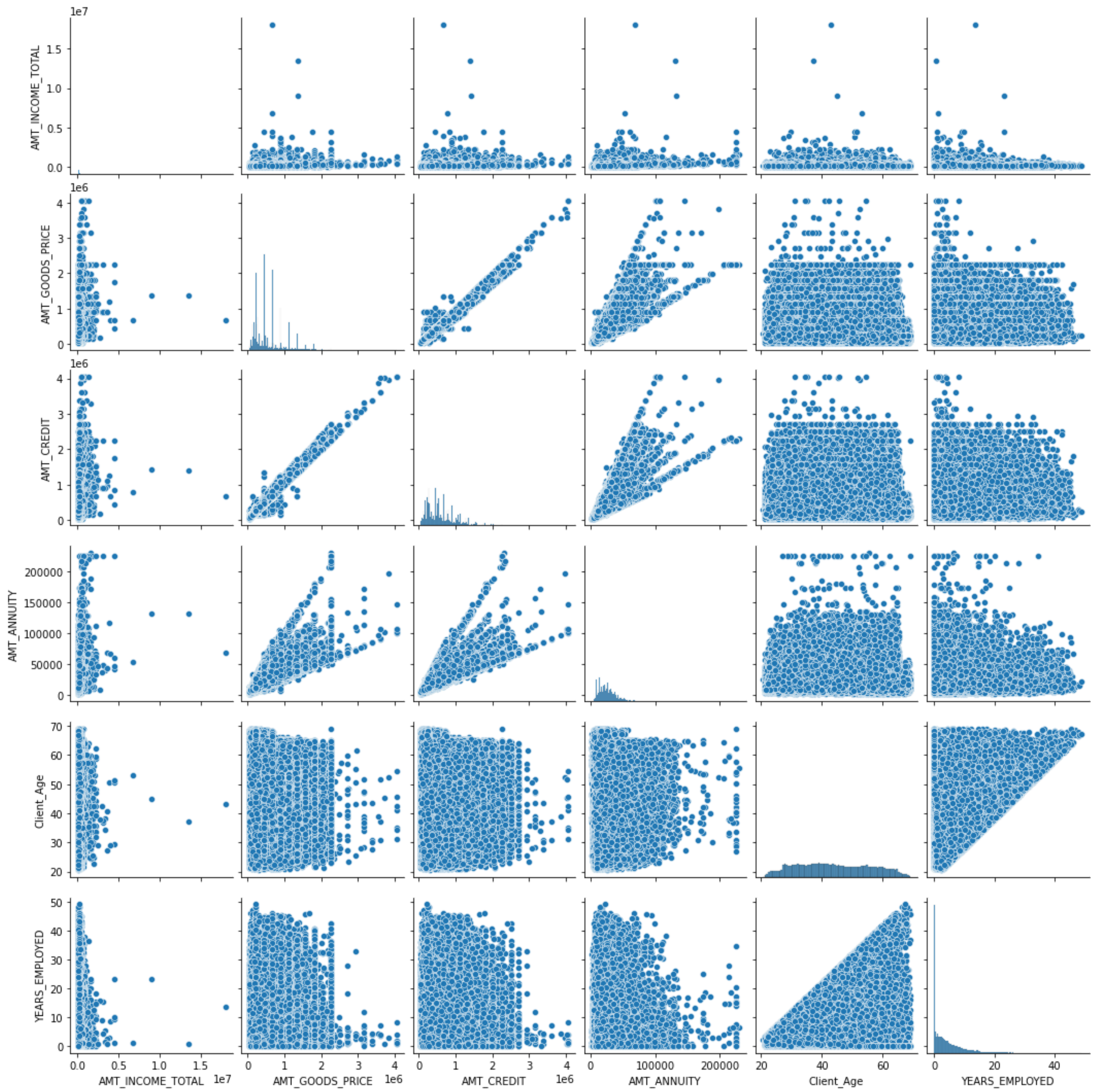


In [82]:
```
fig = px.box(app_target1, x="Work Experience", y="AMT_CREDIT", color='NAME_
             title="Work Experience VS Credit Amount (Payment Difficulties)
fig.show()
```



Numerical vs Numerical Variables

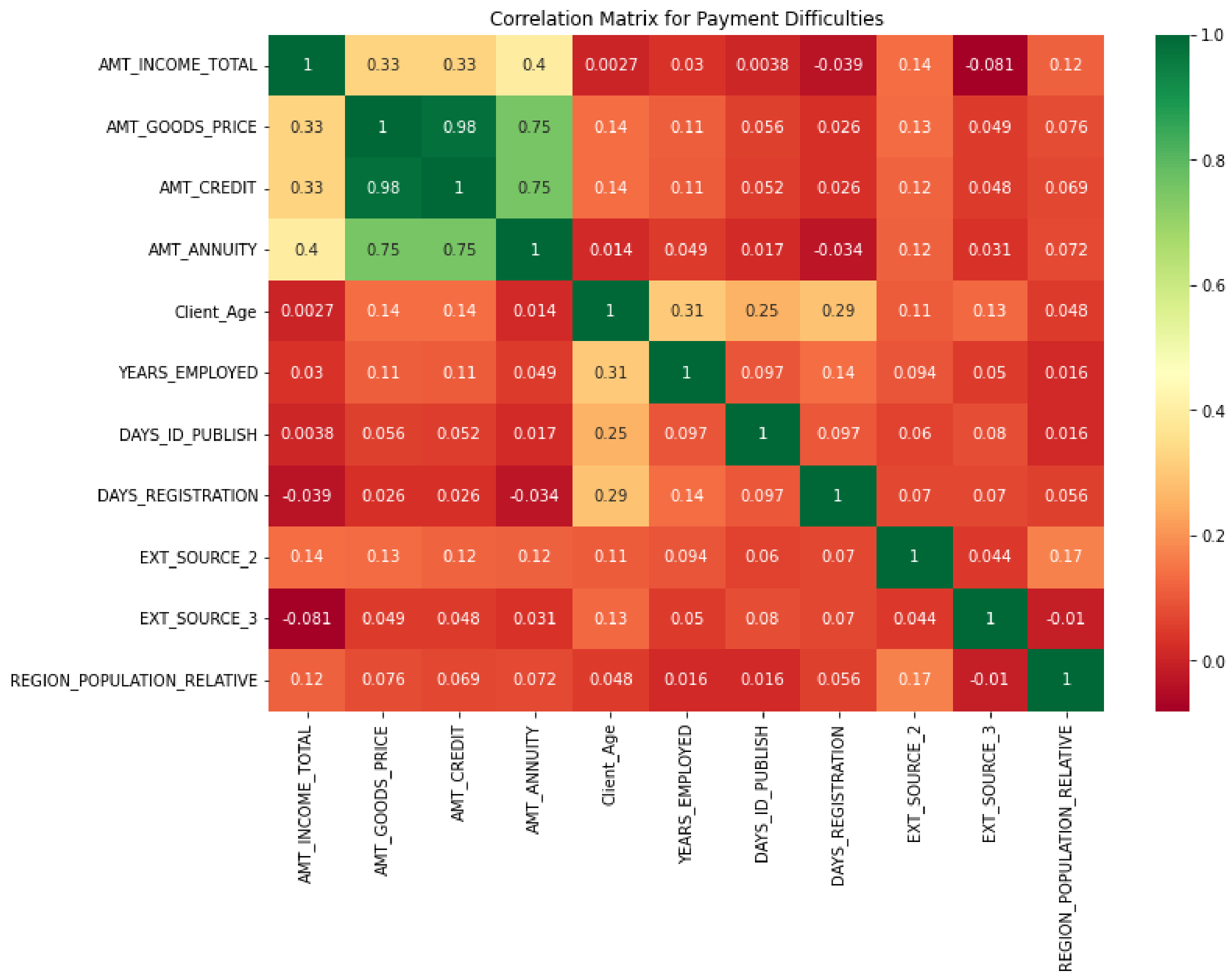
In [83]:
```
sns.pairplot(app_data[['AMT_INCOME_TOTAL', 'AMT_GOODS_PRICE',
                       'AMT_CREDIT', 'AMT_ANNUITY',
                       'Client_Age','YEARS_EMPLOYED' ]].fillna(0))
plt.show()
```



Correlation in target0 & target1

In [84]:
```
plt.figure(figsize=(12,8))
sns.heatmap(app_target0[['AMT_INCOME_TOTAL', 'AMT_GOODS_PRICE',
                         'AMT_CREDIT', 'AMT_ANNUITY',
                         'Client_Age','YEARS_EMPLOYED' ,
                         'DAYS_ID_PUBLISH', 'DAYS_REGISTRATION',
                         'EXT_SOURCE_2','EXT_SOURCE_3','REGION_POPULATION_REL
plt.title('Correlation matrix for Non-Payment Difficulties')
plt.show()
```

In [85]:
```
plt.figure(figsize=(12,8))
sns.heatmap(app_target1[['AMT_INCOME_TOTAL', 'AMT_GOODS_PRICE',
                        'AMT_CREDIT', 'AMT_ANNUITY',
                        'Client_Age','YEARS_EMPLOYED' ,
                        'DAYS_ID_PUBLISH', 'DAYS_REGISTRATION',
                        'EXT_SOURCE_2','EXT_SOURCE_3','REGION_POPULATION_REL
plt.title('Correlation Matrix for Payment Difficulties')
plt.show()
```



Data Analysis on Previous Application dataset

In [86]:
```
appdata_previous = pd.read_csv('previous_application.csv')
appdata_previous.head()
```

Out[86]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CI
0	2030495	271877	Consumer loans	1730.430	17145.0	17
1	2802425	108129	Cash loans	25188.615	607500.0	679
2	2523466	122040	Cash loans	15060.735	112500.0	136
3	2819243	176158	Cash loans	47041.335	450000.0	470
4	1784265	202054	Cash loans	31924.395	337500.0	404

5 rows × 37 columns

Drop Columns with NULL Values greater than 40%

In [87]:
```
s1= (appdata_previous.isnull().mean()*100).sort_values(ascending=False)[app
s1
```

Out[87]:
```
RATE_INTEREST_PRIVILEGED      99.643698
RATE_INTEREST_PRIMARY         99.643698
AMT_DOWN_PAYMENT              53.636480
RATE_DOWN_PAYMENT             53.636480
NAME_TYPE_SUITE               49.119754
NFLAG_INSURED_ON_APPROVAL     40.298129
DAYS_TERMINATION              40.298129
DAYS_LAST_DUE                 40.298129
DAYS_LAST_DUE_1ST_VERSION     40.298129
DAYS_FIRST_DUE                40.298129
DAYS_FIRST_DRAWING            40.298129
dtype: float64
```

In [88]:
```
appdata_previous.shape
```

Out[88]:
```
(1670214, 37)
```

In [89]:
```
appdata_previous.drop(columns = s1.index,inplace=True)
```

In [90]:
```
appdata_previous.shape
```

Out[90]:
```
(1670214, 26)
```

Changing negative values in the DAYS columns to positive values

In [91]:
```
days = []
for i in appdata_previous.columns:
    if 'DAYS' in i:
        days.append(i)
        print('Unique Values in {0} column : {1}'.format(i,appdata_previous
        print()
```

```
Unique Values in DAYS_DECISION column : [  -73  -164  -301 ... -1967 -238
9    -1]
```

In [92]:
```
appdata_previous[days]= abs(appdata_previous[days])
```

In [93]:
```
appdata_previous[days]
```

0	73
1	164
2	301
3	512
4	781
...	...
1670209	544
1670210	1694
1670211	1488
1670212	1185
1670213	1193

1670214 rows × 1 columns

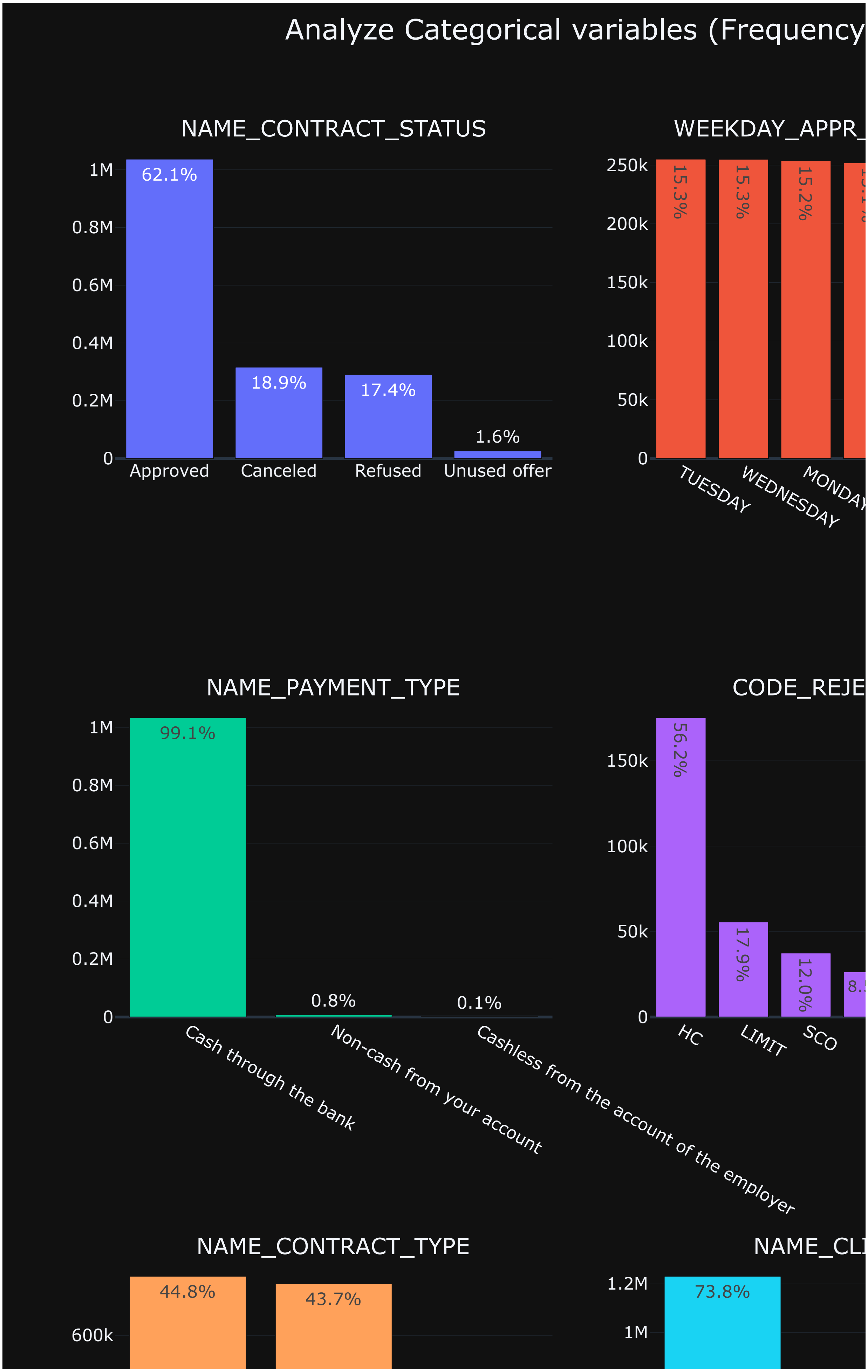
In [94]:
```
# Replcae XNA and XAP are replaced by NaN
appdata_previous=appdata_previous.replace('XNA', np.NaN)
appdata_previous=appdata_previous.replace('XAP', np.NaN)
```

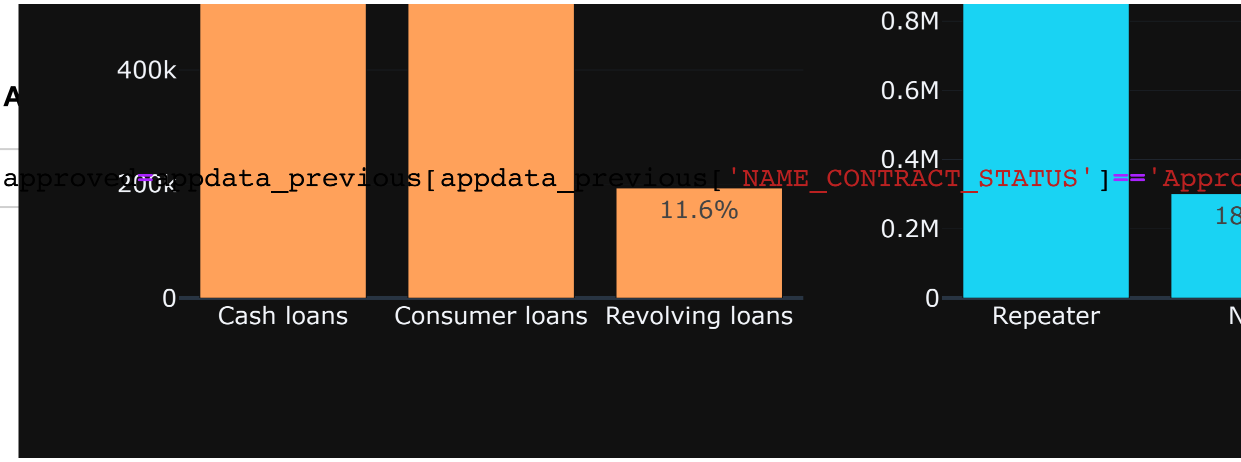
Univariate Analysis on Previous Application Data

In [95]:
```
appdata_previous.columns
```

Out[95]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY',
       'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_GOODS_PRICE',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY',
       'NAME_CASH_LOAN_PURPOSE', 'NAME_CONTRACT_STATUS', 'DAYS_DECISION',
       'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON', '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'],
      dtype='object')
```

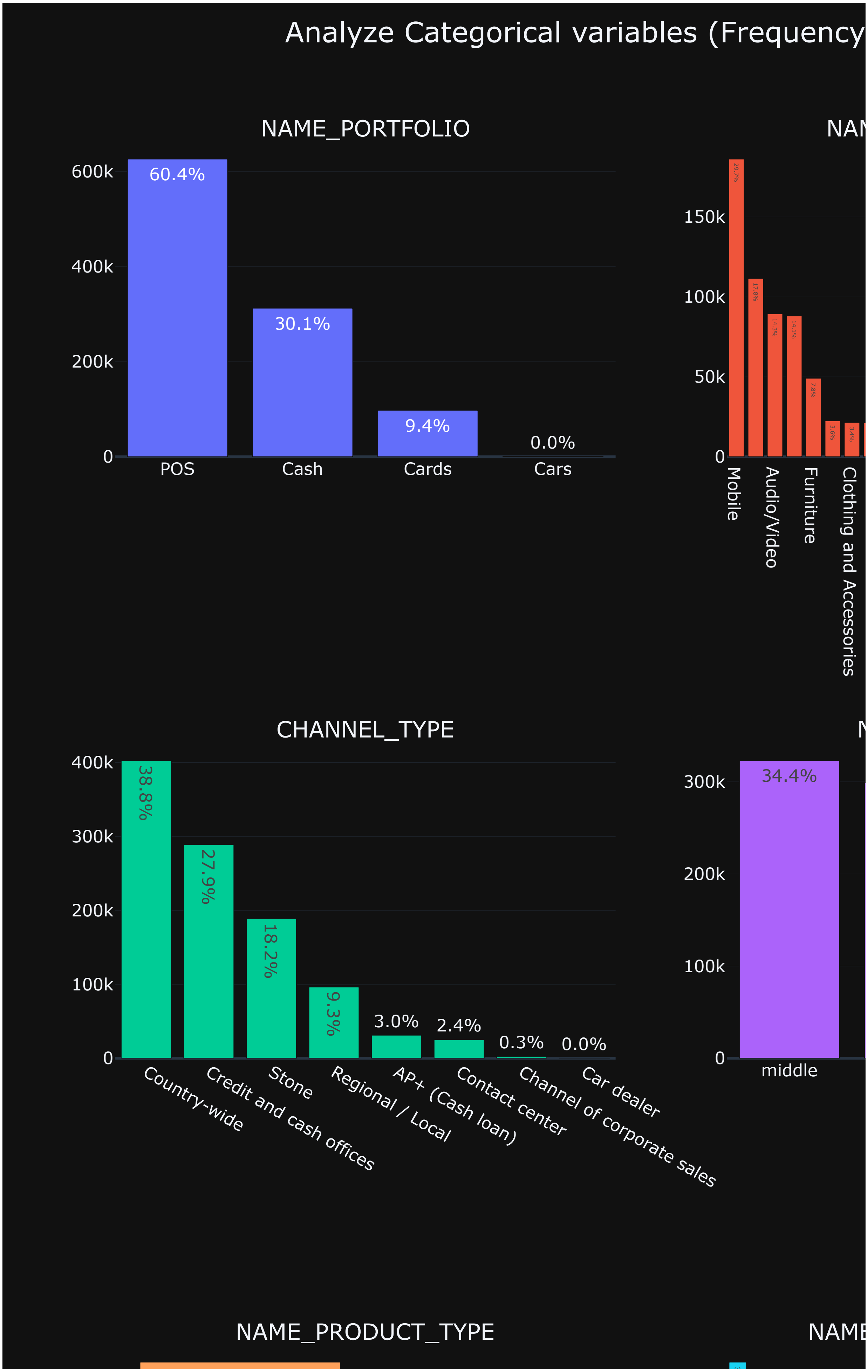
In [96]:
```
cols = ['NAME_CONTRACT_STATUS','WEEKDAY_APPR_PROCESS_START',
        'NAME_PAYMENT_TYPE','CODE_REJECT_REASON',
        'NAME_CONTRACT_TYPE','NAME_CLIENT_TYPE']

#Subplot initialization
fig = make_subplots(
                     rows=3,
                     cols=2,
                     subplot_titles=cols,
                     horizontal_spacing=0.1,
                     vertical_spacing=0.17
                   )
# Adding subplots
count=0
for i in range(1,4):
    for j in range(1,3):
        fig.add_trace(go.Bar(x=appdata_previous[cols[count]].value_counts()
                             y=appdata_previous[cols[count]].value_counts()
                             name=cols[count],
                             textposition='auto',
                             text= [str(i) + '%' for i in (appdata_previous
                            ),
                      row=i,col=j)
        count+=1
fig.update_layout(
                    title=dict(text = "Analyze Categorical variables (Frequ
                    title_font_size=20,
                    showlegend=False,
                    width = 960,
                    height = 1200,
                  )
fig.show()
```

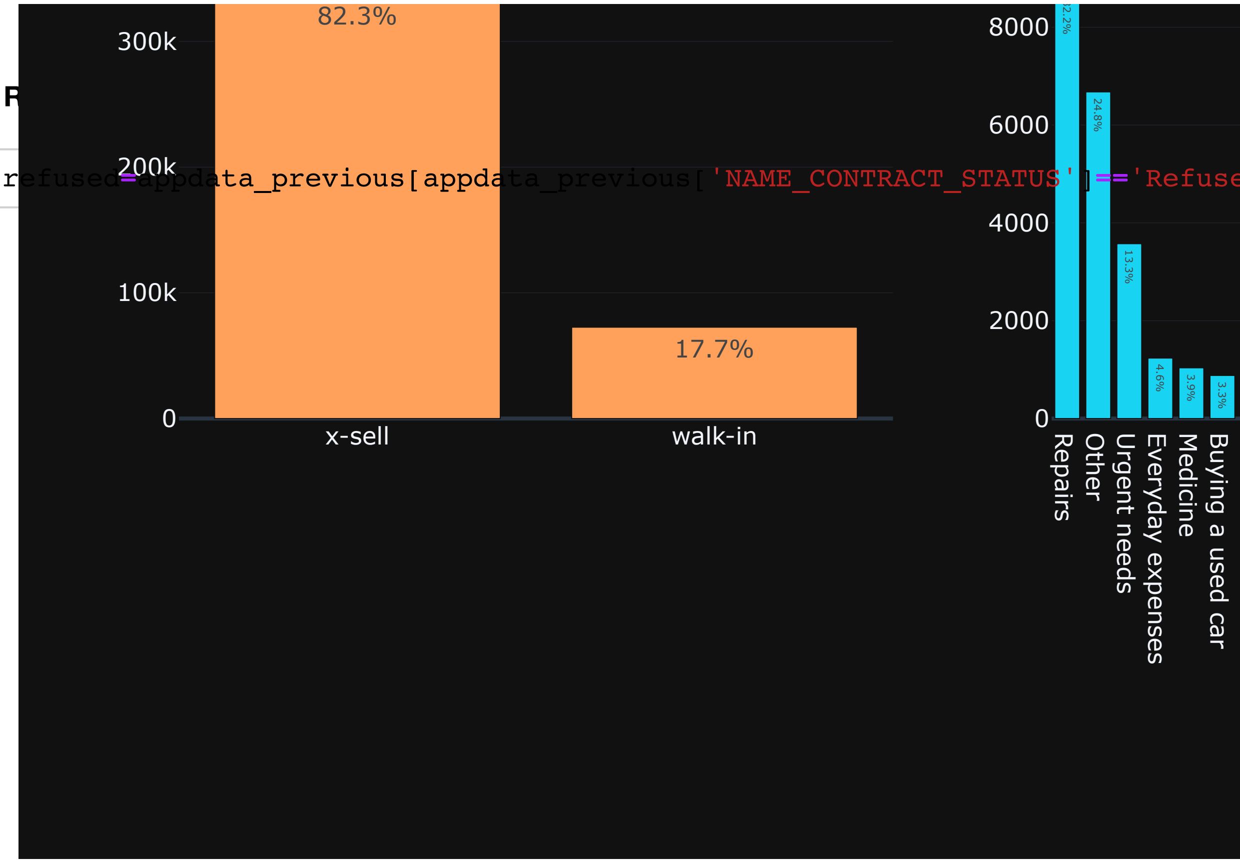

Analyze Categorical variables (Frequency
NAME_CONTRACT_STATUS
1M
0.8M
0.6M
0.4M
0.2M
0
62.1%
18.9%
17.4%
1.6%
Approved
Canceled
Refused
Unused offer
WEEKDAY_APPR_
250k
200k
150k
100k
50k
0
15.3%
15.3%
15.2%
TUESDAY
WEDNESDAY
MONDAY
NAME_PAYMENT_TYPE
1M
0.8M
0.6M
0.4M
0.2M
0
99.1%
0.8%
0.1%
Cash through the bank
Non-cash from your account
Cashless from the account of the employer
CODE_REJE
150k
100k
50k
0
56.2%
17.9%
12.0%
HC
LIMIT
SCO
NAME_CONTRACT_TYPE
600k
44.8%
43.7%
NAME_CLI
1.2M
1M
73.8%



In [97]:
```
approved=appdata_previous[appdata_previous['NAME_CONTRACT_STATUS']=='Approv
```

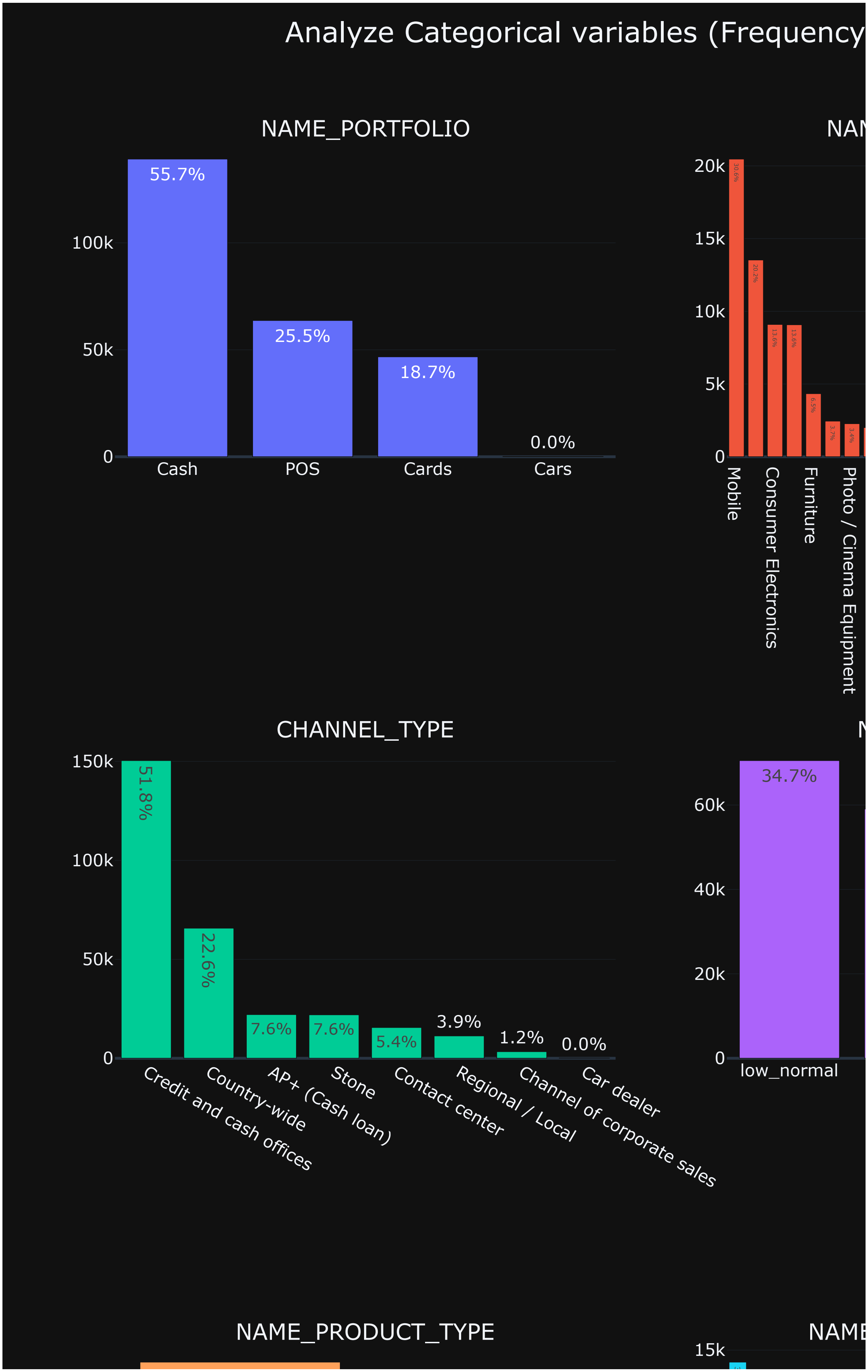
In [98]:
```
; = ['NAME_PORTFOLIO','NAME_GOODS_CATEGORY',
     'CHANNEL_TYPE','NAME_YIELD_GROUP' , 'NAME_PRODUCT_TYPE','NAME_CASH_LOAN

›plot initialization
 = make_subplots(
                rows=3,
                cols=2,
                subplot_titles=cols,
                horizontal_spacing=0.1,
                vertical_spacing=0.19
               )
ıding subplots
ıt=0
 i in range(1,4):
 for j in range(1,3):
     fig.add_trace(go.Bar(x=approved[cols[count]].value_counts().index,
                         y=approved[cols[count]].value_counts(),
                         name=cols[count],
                         textposition='auto',
                         text= [str(i) + '%' for i in (approved[cols[count]
                        ),
                  row=i,col=j)
     count+=1
 update_layout(
                title=dict(text = "Analyze Categorical variables (Frequency
                title_font_size=20,
                showlegend=False,
                width = 960,
                height = 1400,
               )
 show()
```


Analyze Categorical variables (Frequency
NAME_PORTFOLIO
600k
400k
200k
0
60.4%
30.1%
9.4%
0.0%
POS
Cash
Cards
Cars
NAN
150k
100k
50k
0
Mobile
Audio/Video
Furniture
Clothing and Accessories
CHANNEL_TYPE
400k
300k
200k
100k
0
38.8%
27.9%
18.2%
9.3%
3.0%
2.4%
0.3%
0.0%
Country-wide
Credit and cash offices
Stone
Regional / Local
AP+ (Cash loan)
Contact center
Channel of corporate sales
Car dealer
300k
200k
100k
0
34.4%
middle
NAME_PRODUCT_TYPE
NAME



In [99]:
```
refused=appdata_previous[appdata_previous['NAME_CONTRACT_STATUS']=='Refused
```

In [100]:
```
cols = ['NAME_PORTFOLIO','NAME_GOODS_CATEGORY',
        'CHANNEL_TYPE','NAME_YIELD_GROUP' , 'NAME_PRODUCT_TYPE','NAME_CASH_

#Subplot initialization
fig = make_subplots(
                     rows=3,
                     cols=2,
                     subplot_titles=cols,
                     horizontal_spacing=0.1,
                     vertical_spacing=0.19
                   )
# Adding subplots
count=0
for i in range(1,4):
    for j in range(1,3):
        fig.add_trace(go.Bar(x=refused[cols[count]].value_counts().index,
                             y=refused[cols[count]].value_counts(),
                             name=cols[count],
                             textposition='auto',
                             text= [str(i) + '%' for i in (refused[cols[cou
                            ),
                      row=i,col=j)
        count+=1
fig.update_layout(
                    title=dict(text = "Analyze Categorical variables (Frequ
                    title_font_size=20,
                    showlegend=False,
                    width = 960,
                    height = 1400,
                  )
fig.show()
```



Analyze Categorical variables (Frequency
NAME_PORTFOLIO
55.7%
25.5%
18.7%
0.0%
100k
50k
0
Cash
POS
Cards
Cars
NAM
20k
15k
10k
5k
0
Mobile
Consumer Electronics
Furniture
Photo / Cinema Equipment
CHANNEL_TYPE
150k
100k
50k
0
51.8%
22.6%
7.6%
7.6%
5.4%
3.9%
1.2%
0.0%
Credit and cash offices
Country-wide
AP+ (Cash loan)
Stone
Contact center
Regional / Local
Channel of corporate sales
Car dealer
N
60k
40k
20k
0
34.7%
low_normal
NAME_PRODUCT_TYPE
NAME
15k

M

In [101]:
```python
appdata_merge = app_data.merge(appdata_previous,on='SK_ID_CURR',how='inner
appdata_merge.shape
```

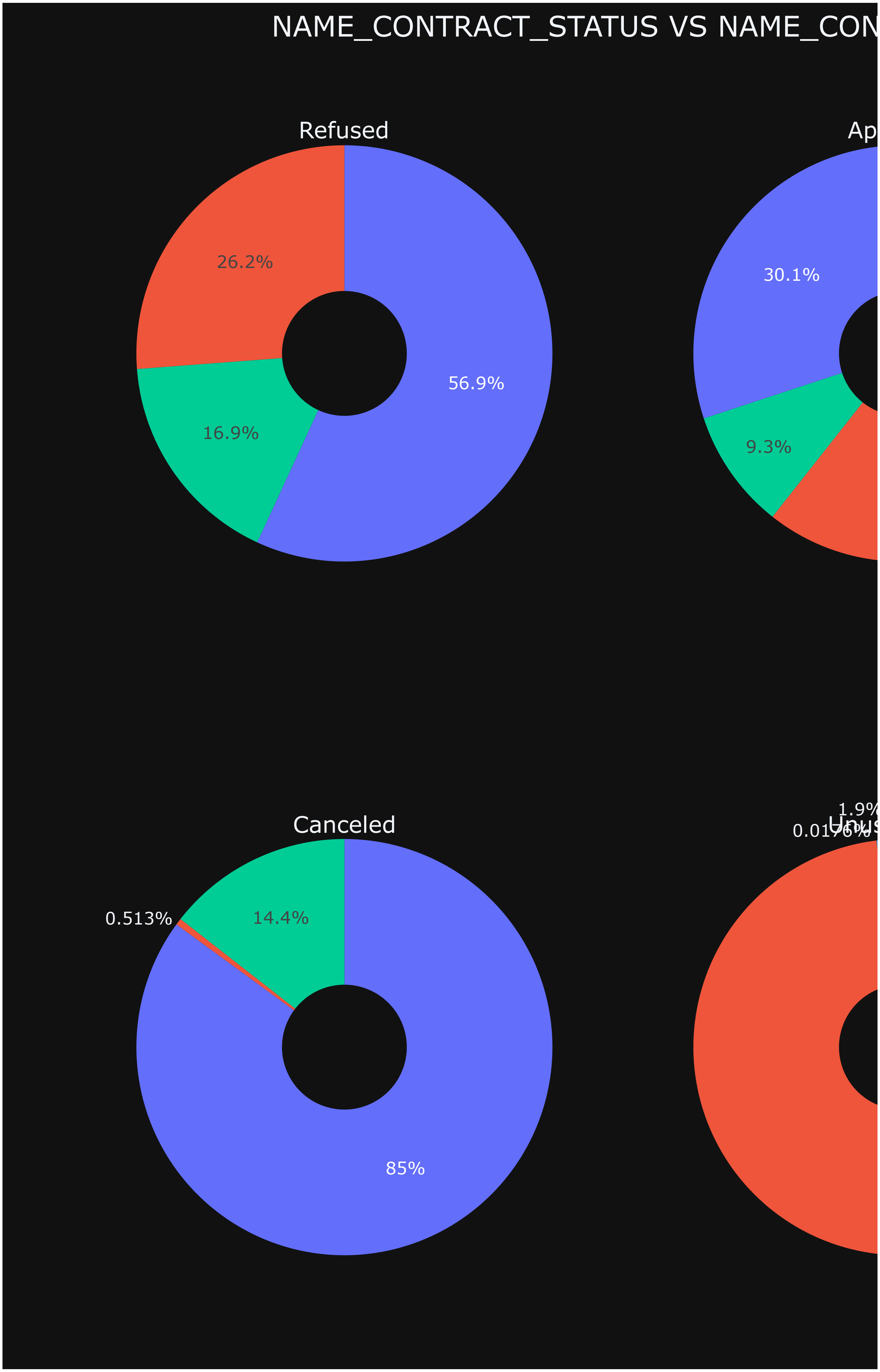
Out[101]: (1413670, 76)

A

In [102]:
```python
# Function for multiple plotting - Bar Chart
def plot_merge(appdata_merge,column_name):
    col_value = ['Refused','Approved', 'Canceled' , 'Unused offer']

    #Subplot initialization
    fig = make_subplots(
                     rows=2,
                     cols=2,
                     subplot_titles=col_value,
                     horizontal_spacing=0.1,
                     vertical_spacing=0.3
                    )
    # Adding subplots
    count=0
    for i in range(1,3):
        for j in range(1,3):
            fig.add_trace(go.Bar(x=appdata_merge[appdata_merge['NAME_CONTRAC
                                 y=appdata_merge[appdata_merge['NAME_CONTRACT_ST
                                 name=cols[count],
                                 textposition='auto',
                                 text= [str(i) + '%' for i in (appdata_merge[app
                                ),
                          row=i,col=j)
            count+=1
    fig.update_layout(
                    title=dict(text = "NAME_CONTRACT_STATUS VS "+column_name
                    title_font_size=20,
                    showlegend=False,
                    width = 960,
                    height = 960,
                  )
    fig.show()
```

In [103]:
```
# Function for multiple plotting - Pie Chart
def plot_pie_merge(appdata_merge,column_name):
    col_value = ['Refused','Approved', 'Canceled' , 'Unused offer']

    #Subplot initialization
    fig = make_subplots(
                    rows=2,
                    cols=2,
                    subplot_titles=col_value,
                    specs=[[{"type": "pie"}, {"type": "pie"}],[{"type": "p
                  )
    # Adding subplots
    count=0
    for i in range(1,3):
        for j in range(1,3):
            fig.add_trace(go.Pie(labels=appdata_merge[appdata_merge['NAME_C
                            values=appdata_merge[appdata_merge['NAME_CONTR
                            textinfo='percent',
                            insidetextorientation='auto',
                            hole=.3
                            ),
                     row=i,col=j)
            count+=1
    fig.update_layout(
                   title=dict(text = "NAME_CONTRACT_STATUS VS "+column_nam
                   title_font_size=20,
                   width = 960,
                   height = 960,
                  )
    fig.show()
```

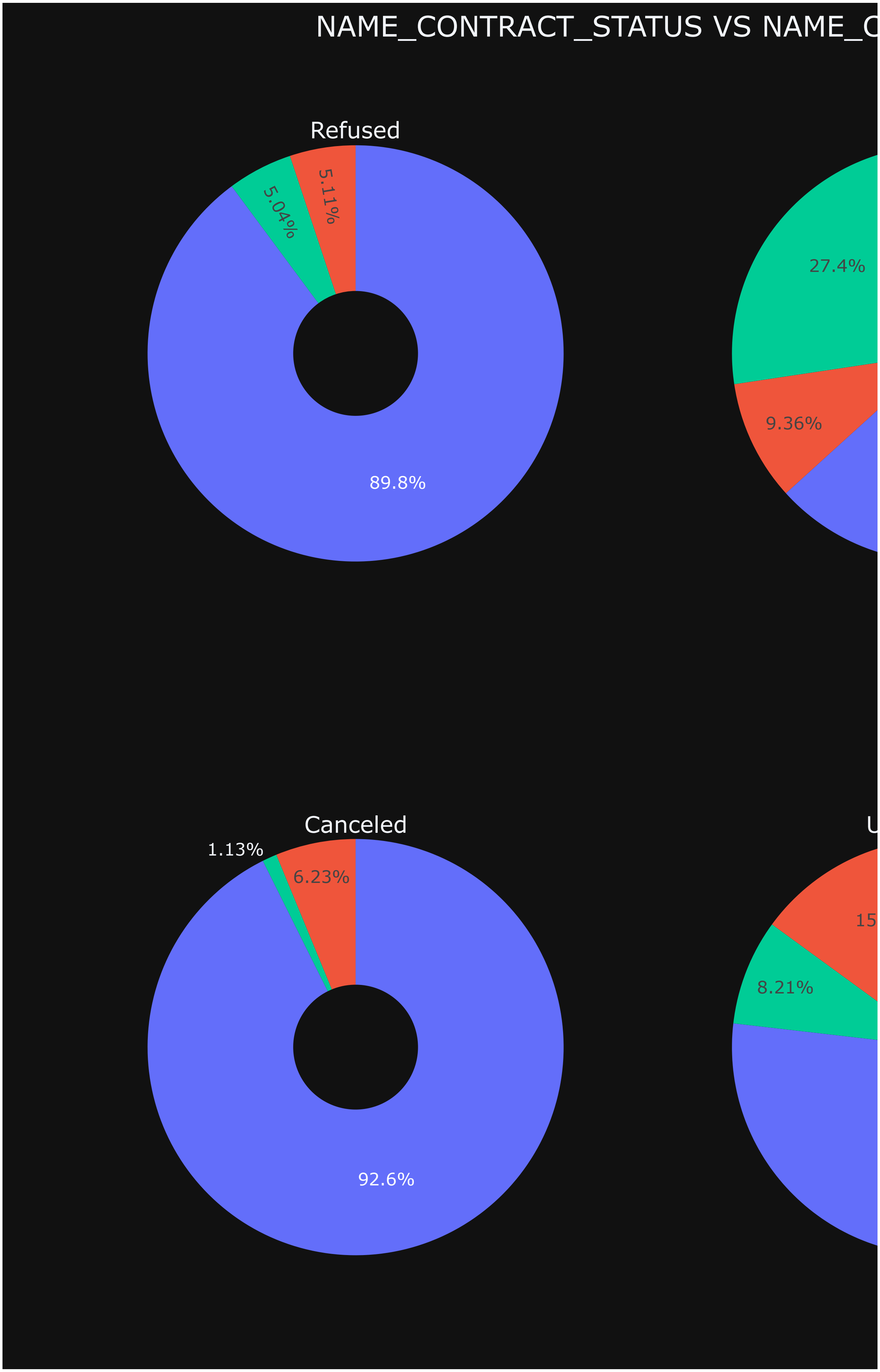
In [104]:
```
plot_pie_merge(appdata_merge,'NAME_CONTRACT_TYPE_y')
```


NAME_CONTRACT_STATUS VS NAME_CON
Refused
26.2%
56.9%
16.9%
Ap
30.1%
9.3%
Canceled
0.513%
14.4%
85%
1.9%
0.0176%

Insights

- Banks mostly approve Consumer Loans
- Most of the _*Refused* & **Cancelled** loans are **cash loans.**

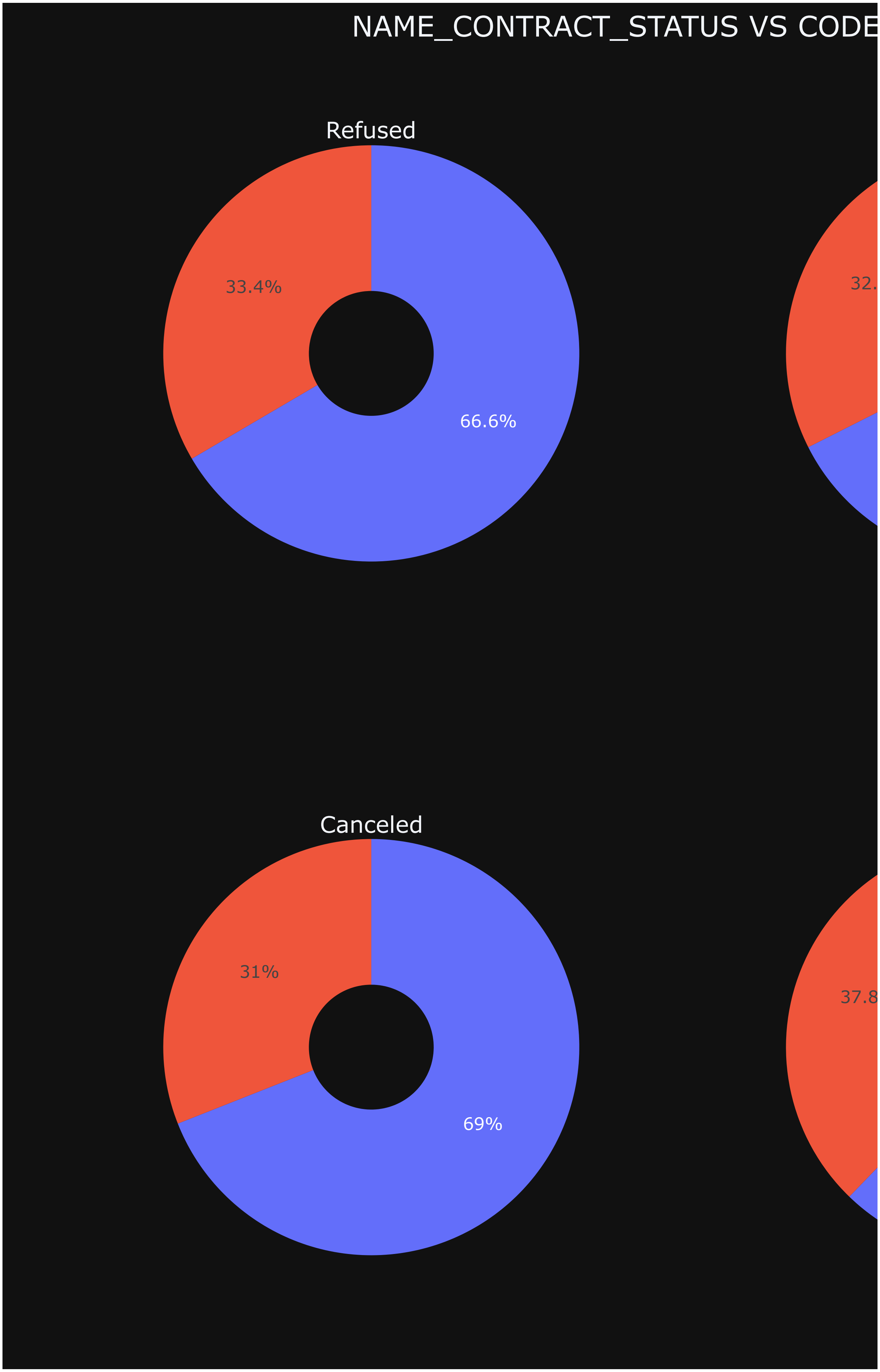
In [105]:
```
plot_pie_merge(appdata_merge,'NAME_CLIENT_TYPE')
```


NAME_CONTRACT_STATUS VS NAME_C
Refused
5.11%
5.04%
89.8%
27.4%
9.36%
Canceled
1.13%
6.23%
92.6%
U
15
8.21%

Insights

- Most of the approved , refused & canceled loans belong to the old clients.
- Almost **27.4%** loans were provided to new customers.

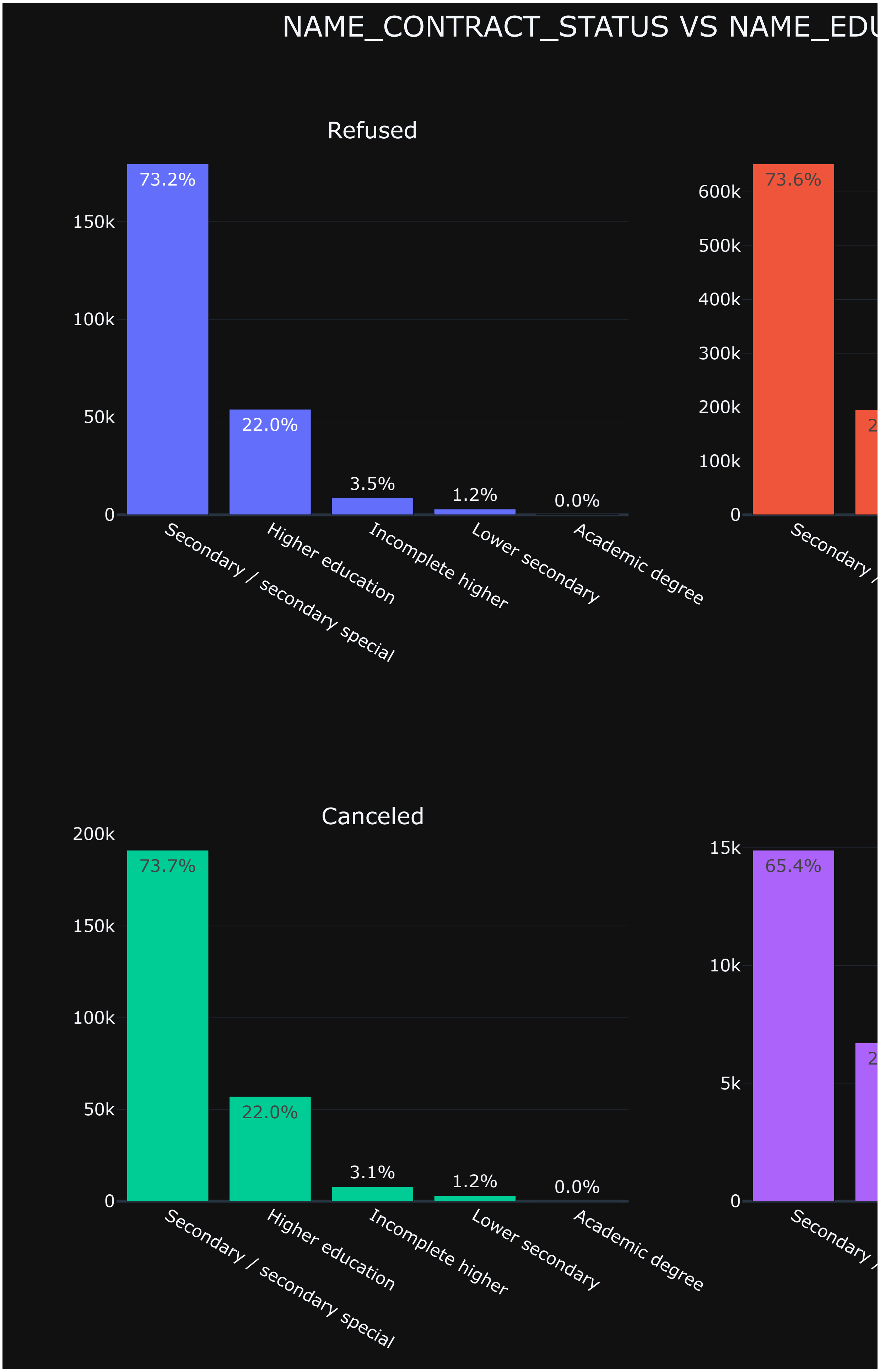
In [106]:
```
plot_pie_merge(appdata_merge,'CODE_GENDER')
```


NAME_CONTRACT_STATUS VS CODE
Refused
33.4%
66.6%
32.
Canceled
31%
69%
37.8

Insights

- Approved percentage of loans provided to females is more as compared to refused percentage.

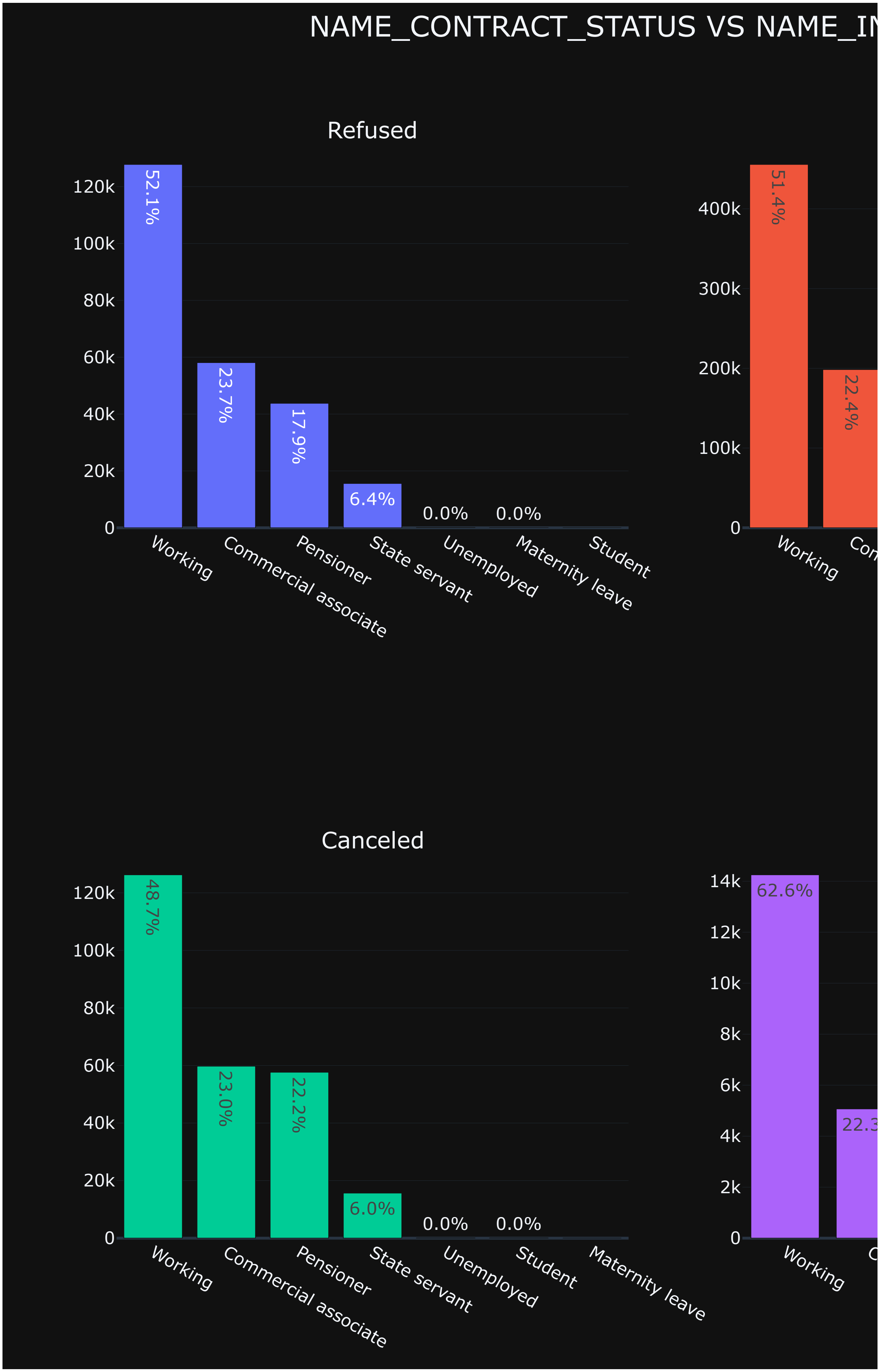
In [107]:
```
plot_merge(appdata_merge,'NAME_EDUCATION_TYPE')
```


NAME_CONTRACT_STATUS VS NAME_EDU
Refused
73.2%
22.0%
3.5%
1.2%
0.0%
150k
100k
50k
0
Secondary / secondary special
Higher education
Incomplete higher
Lower secondary
Academic degree
73.6%
600k
500k
400k
300k
200k
100k
0
Secondary /
Canceled
200k
150k
100k
50k
0
73.7%
22.0%
3.1%
1.2%
0.0%
Secondary / secondary special
Higher education
Incomplete higher
Lower secondary
Academic degree
15k
10k
5k
0
65.4%
Secondary /

Insights

- Most of the approved loans belong to applicants with **Secondary / Secondary Special** education type.

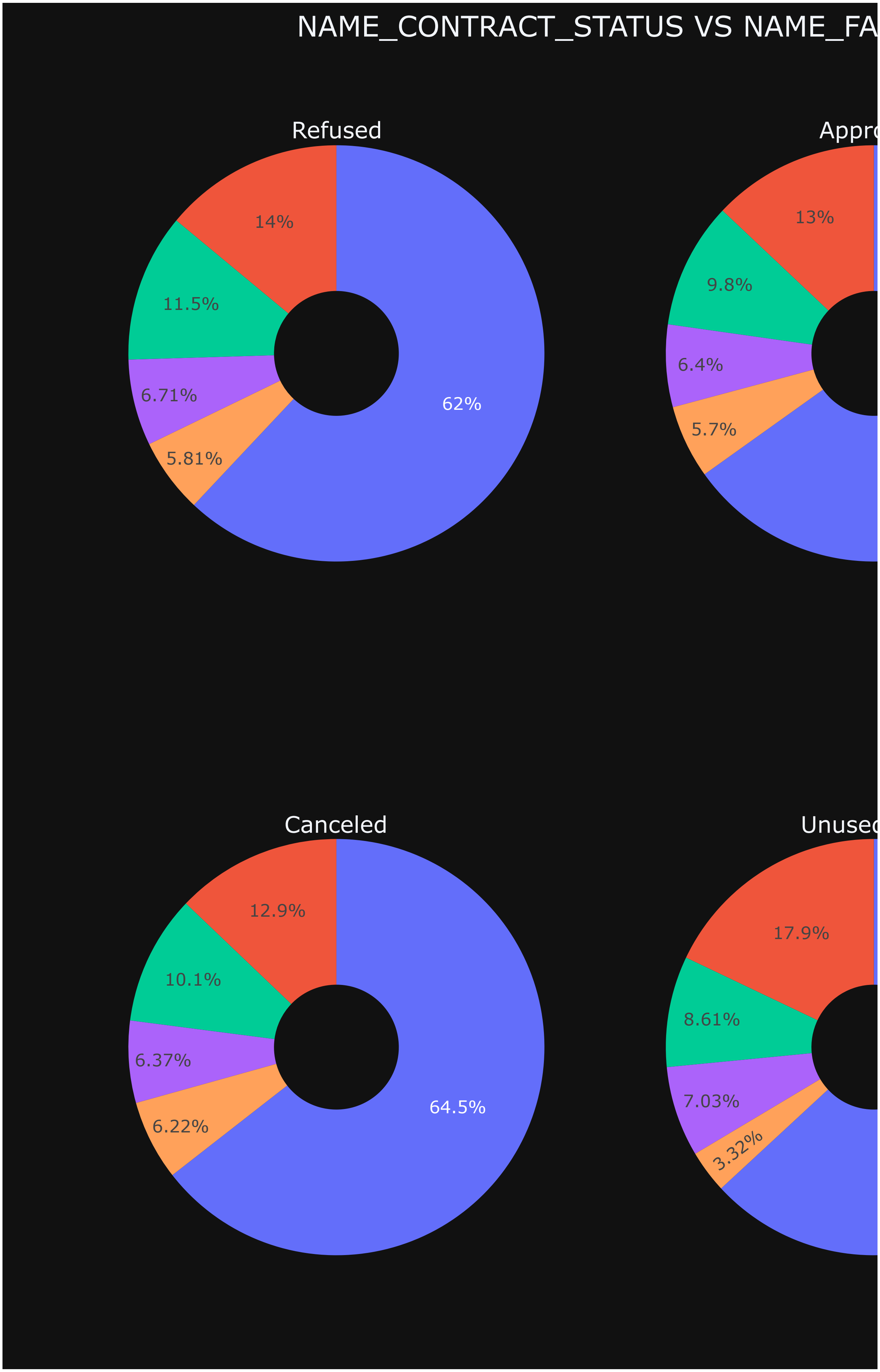
In [108]:
```
plot_merge(appdata_merge,'NAME_INCOME_TYPE')
```


NAME_CONTRACT_STATUS VS NAME_IN
Refused
120k
100k
80k
60k
40k
20k
0
52.1%
23.7%
17.9%
6.4%
0.0%
0.0%
Working
Commercial associate
Pensioner
State servant
Unemployed
Maternity leave
Student
400k
300k
200k
100k
0
51.4%
22.4%
Working
Canceled
120k
100k
80k
60k
40k
20k
0
48.7%
23.0%
22.2%
6.0%
0.0%
0.0%
Working
Commercial associate
Pensioner
State servant
Unemployed
Student
Maternity leave
14k
12k
10k
8k
6k
4k
2k
0
62.6%
Working

Insights

- Across all Contract Status (Approved , Refused , Canceled , Unused Offer) people with **Working** income type are leading. So it is quite evident that majority of the loans are coming from this income type class.

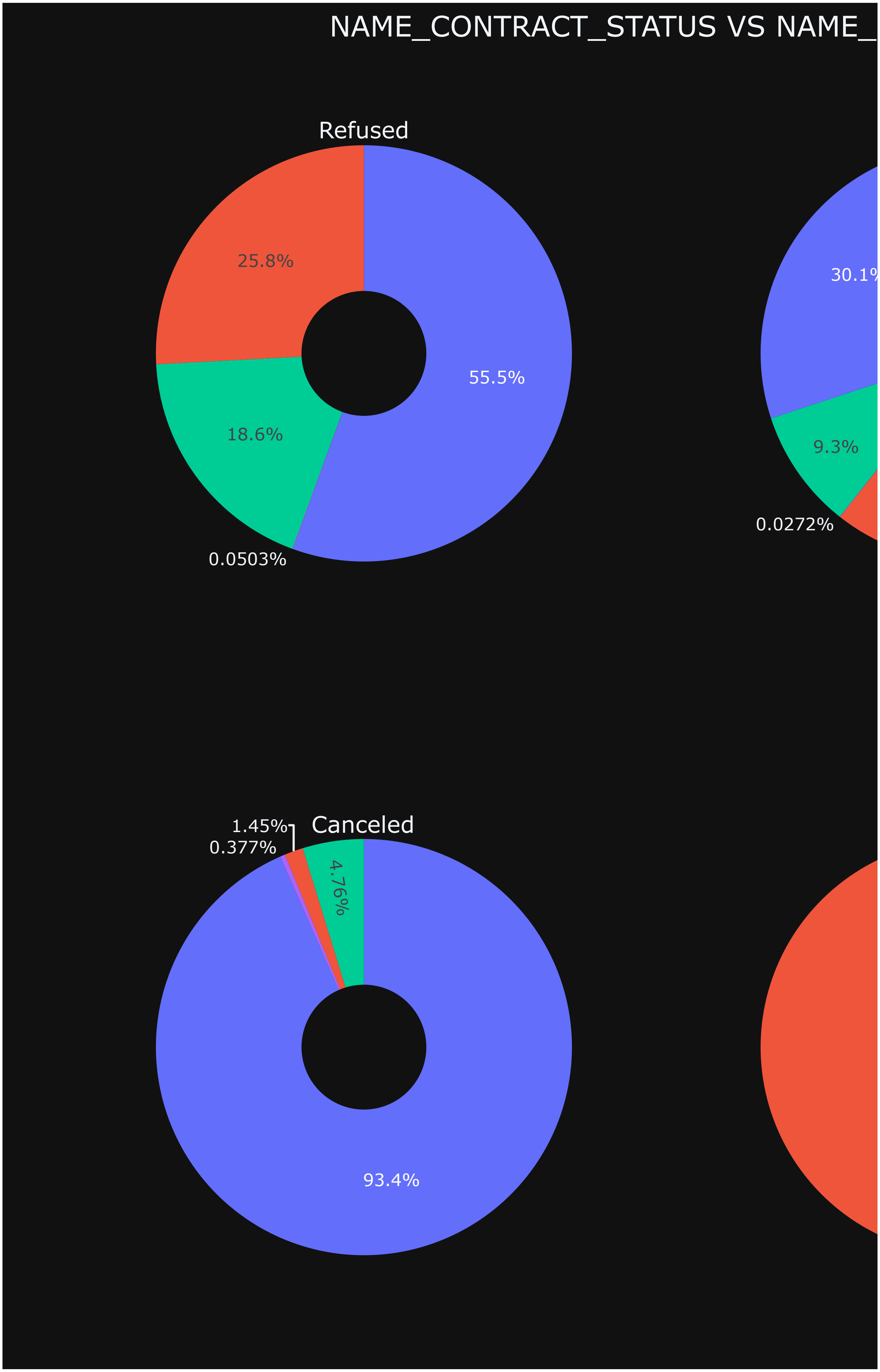
In [109]:
```
plot_pie_merge(appdata_merge,'NAME_FAMILY_STATUS')
```


NAME_CONTRACT_STATUS VS NAME_FA
Refused
14%
11.5%
6.71%
5.81%
62%
Appr
13%
9.8%
6.4%
5.7%
Canceled
12.9%
10.1%
6.37%
6.22%
64.5%
Unuse
17.9%
8.61%
7.03%
3.32%

Insights

- Approved percentage of loans for married applicants is higher than the rest of the contract status (refused , canceled etc.).

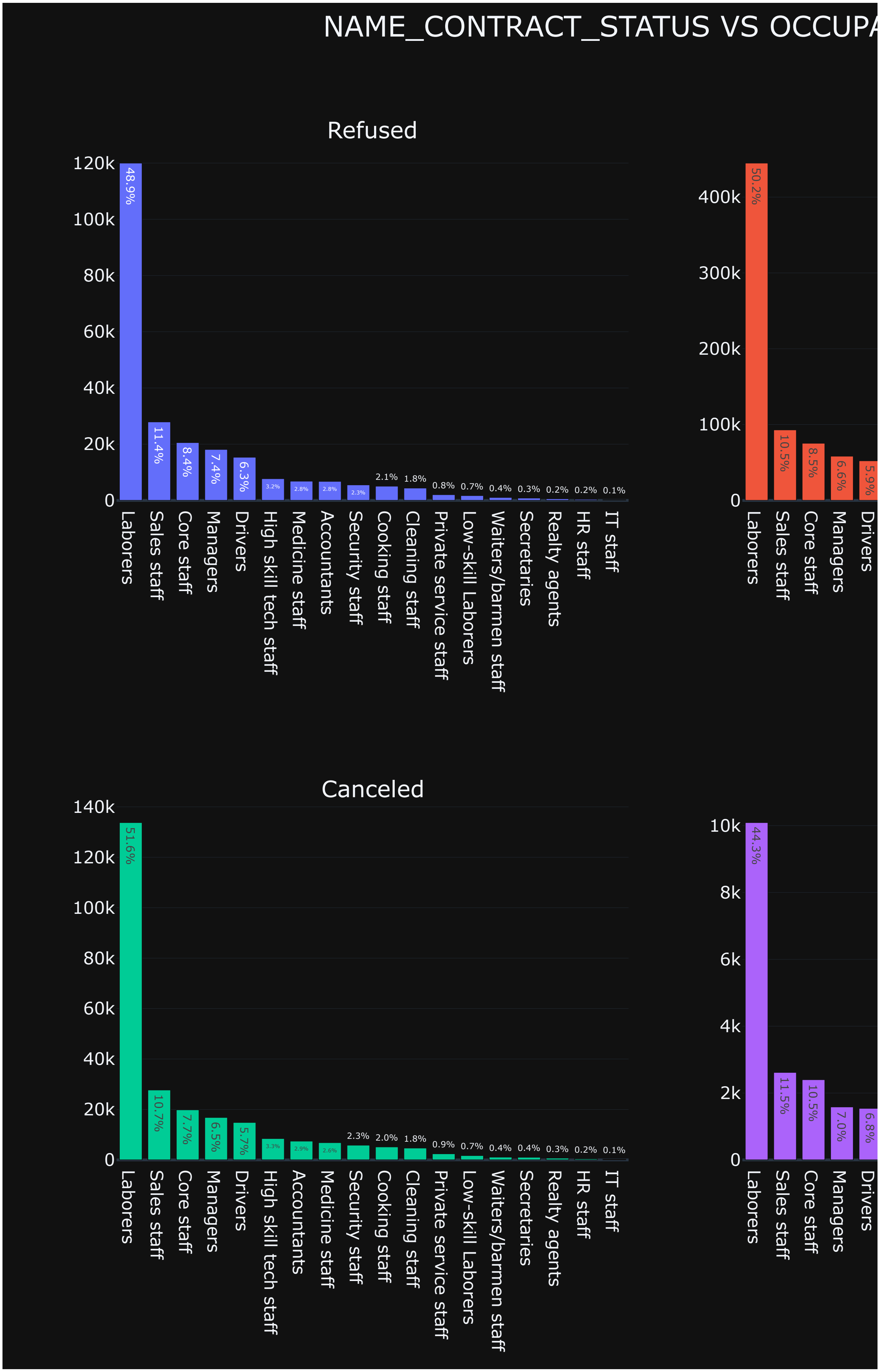
In [110]:
```
plot_pie_merge(appdata_merge,'NAME_PORTFOLIO')
```


NAME_CONTRACT_STATUS VS NAME_
Refused
25.8%
55.5%
18.6%
0.0503%
30.1%
9.3%
0.0272%
Canceled
1.45%
0.377%
4.76%
93.4%

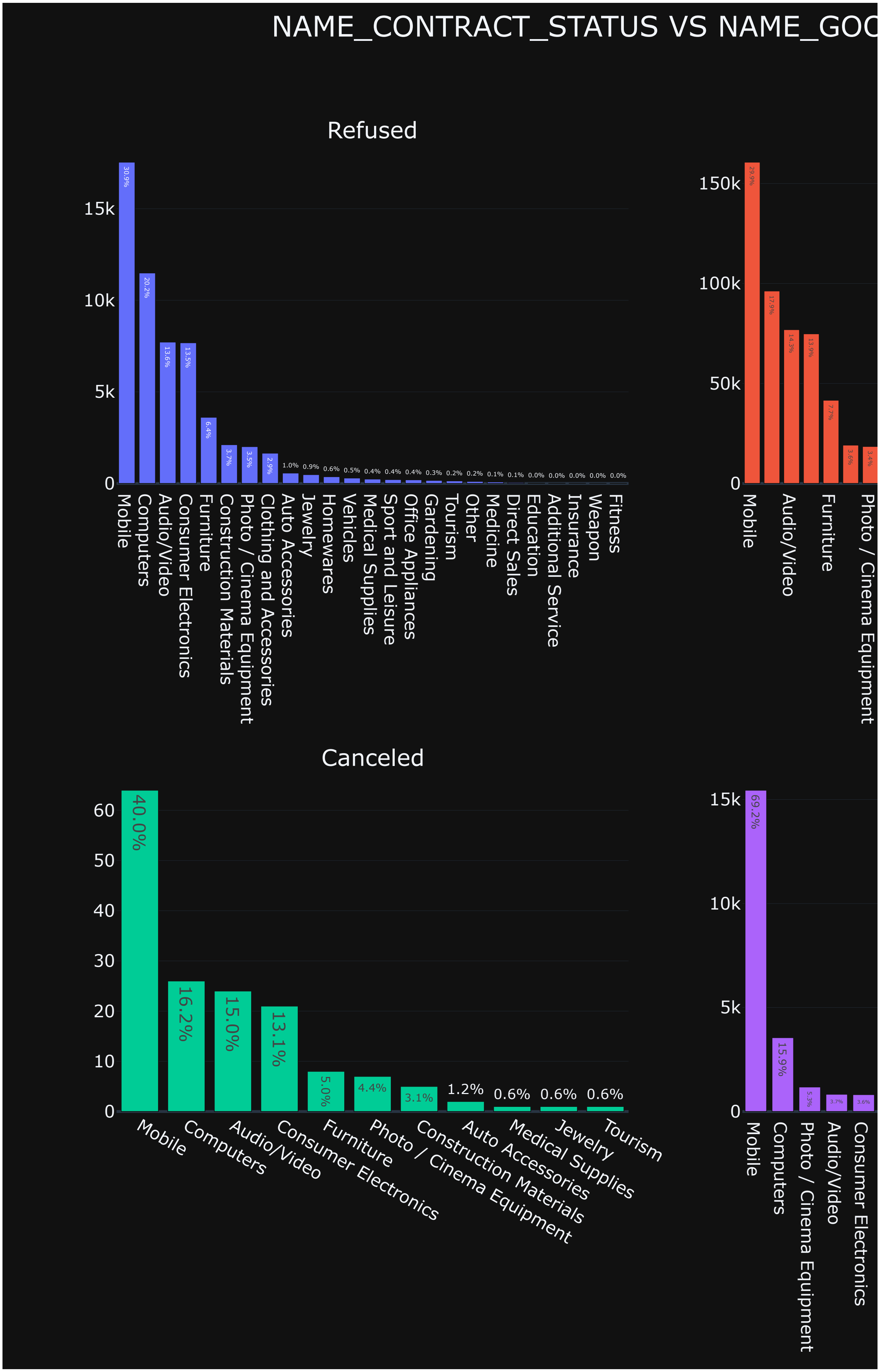
Insights

- 60.6% previous approved loans belong to **POS** name portfolio.
- Majority of the loans refused were cash loans.
- 93.4% loans that belong to **POS** were canceled

In [111]:
```
plot_merge(appdata_merge,'OCCUPATION_TYPE')
```

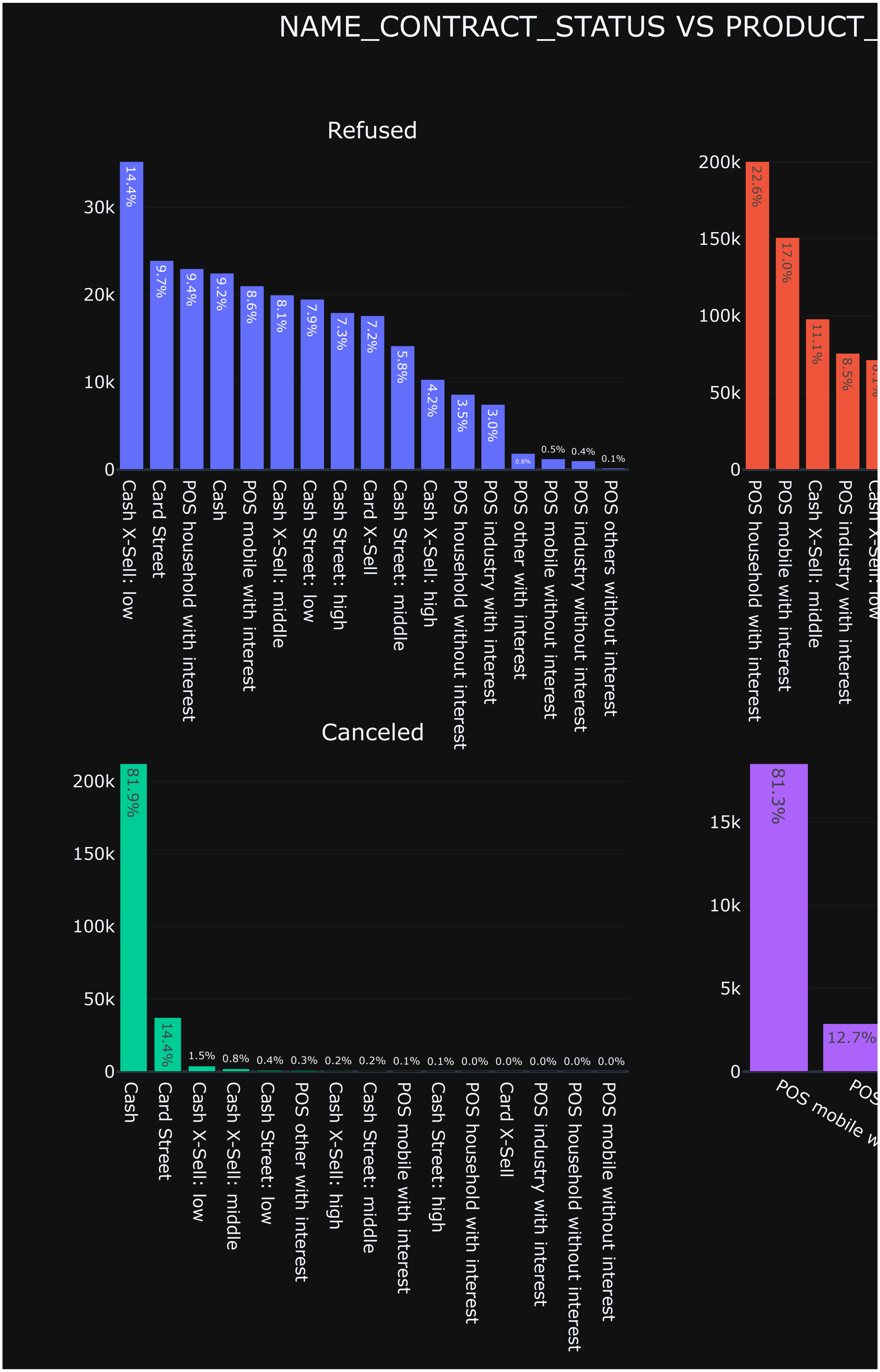

NAME_CONTRACT_STATUS VS OCCUPA
Refused
Canceled
Laborers
Sales staff
Core staff
Managers
Drivers
High skill tech staff
Medicine staff
Accountants
Security staff
Cooking staff
Cleaning staff
Private service staff
Low-skill Laborers
Waiters/barmen staff
Secretaries
Realty agents
HR staff
IT staff
48.9%
11.4%
8.4%
7.4%
6.3%
50.2%
10.5%
8.5%
6.6%
5.9%
51.6%
10.7%
7.7%
6.5%
5.7%
44.3%
11.5%
10.5%
7.0%
6.8%

In [112]:
```
plot_merge(appdata_merge,'NAME_GOODS_CATEGORY')
```



NAME_CONTRACT_STATUS VS NAME_GOO
Refused
Canceled
Mobile
Computers
Audio/Video
Consumer Electronics
Furniture
Construction Materials
Photo / Cinema Equipment
Clothing and Accessories
Auto Accessories
Jewelry
Homewares
Vehicles
Medical Supplies
Sport and Leisure
Office Appliances
Gardening
Tourism
Other
Medicine
Direct Sales
Education
Additional Service
Insurance
Weapon
Fitness
30.9%
20.2%
13.6%
13.5%
6.4%
3.7%
3.5%
2.9%
1.0%
0.9%
0.6%
0.5%
0.4%
0.4%
0.4%
0.3%
0.2%
0.2%
0.1%
0.1%
0.0%
0.0%
0.0%
0.0%
0.0%
40.0%
16.2%
15.0%
13.1%
5.0%
4.4%
3.1%
1.2%
0.6%
0.6%
0.6%
69.2%
15.9%

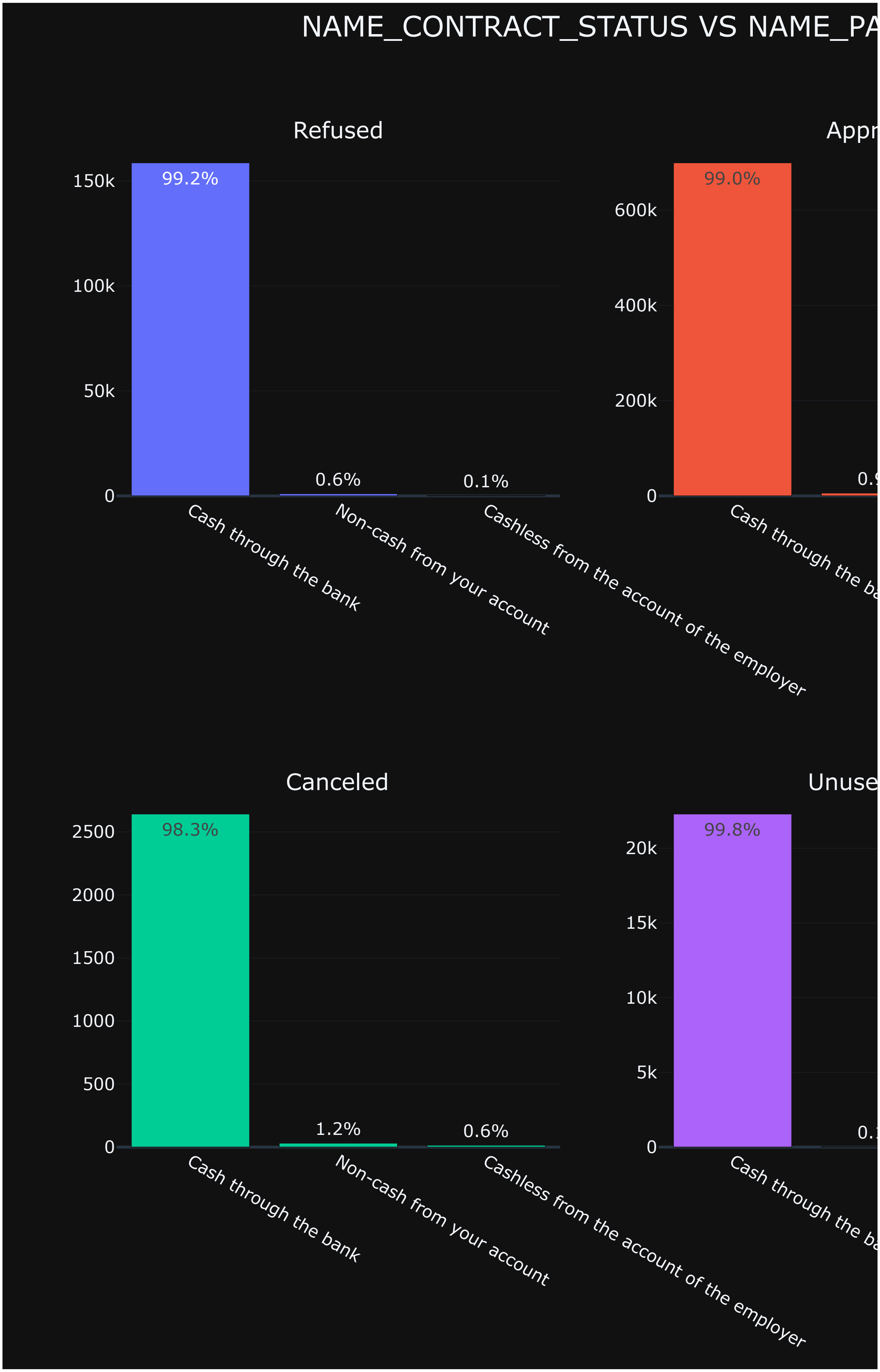
In [113]:
```
plot_merge(appdata_merge,'PRODUCT_COMBINATION')
```


NAME_CONTRACT_STATUS VS PRODUCT_
Refused
30k
20k
10k
0
14.4%
9.7%
9.4%
9.2%
8.6%
8.1%
7.9%
7.3%
7.2%
5.8%
4.2%
3.5%
3.0%
0.8%
0.5%
0.4%
0.1%
Cash X-Sell: low
Card Street
POS household with interest
Cash
POS mobile with interest
Cash X-Sell: middle
Cash Street: low
Cash Street: high
Card X-Sell
Cash Street: middle
Cash X-Sell: high
POS household without interest
POS industry with interest
POS other with interest
POS mobile without interest
POS industry without interest
POS others without interest
200k
150k
100k
50k
0
22.6%
17.0%
11.1%
8.5%
POS household with interest
POS mobile with interest
Cash X-Sell: middle
POS industry with interest
Canceled
200k
150k
100k
50k
0
81.9%
14.4%
1.5%
0.8%
0.4%
0.3%
0.2%
0.2%
0.1%
0.1%
0.0%
0.0%
0.0%
0.0%
0.0%
Cash
Card Street
Cash X-Sell: low
Cash X-Sell: middle
Cash Street: low
POS other with interest
Cash X-Sell: high
Cash Street: middle
POS mobile with interest
Cash Street: high
POS household with interest
Card X-Sell
POS industry with interest
POS household without interest
POS mobile without interest
15k
10k
5k
0
81.3%
12.7%

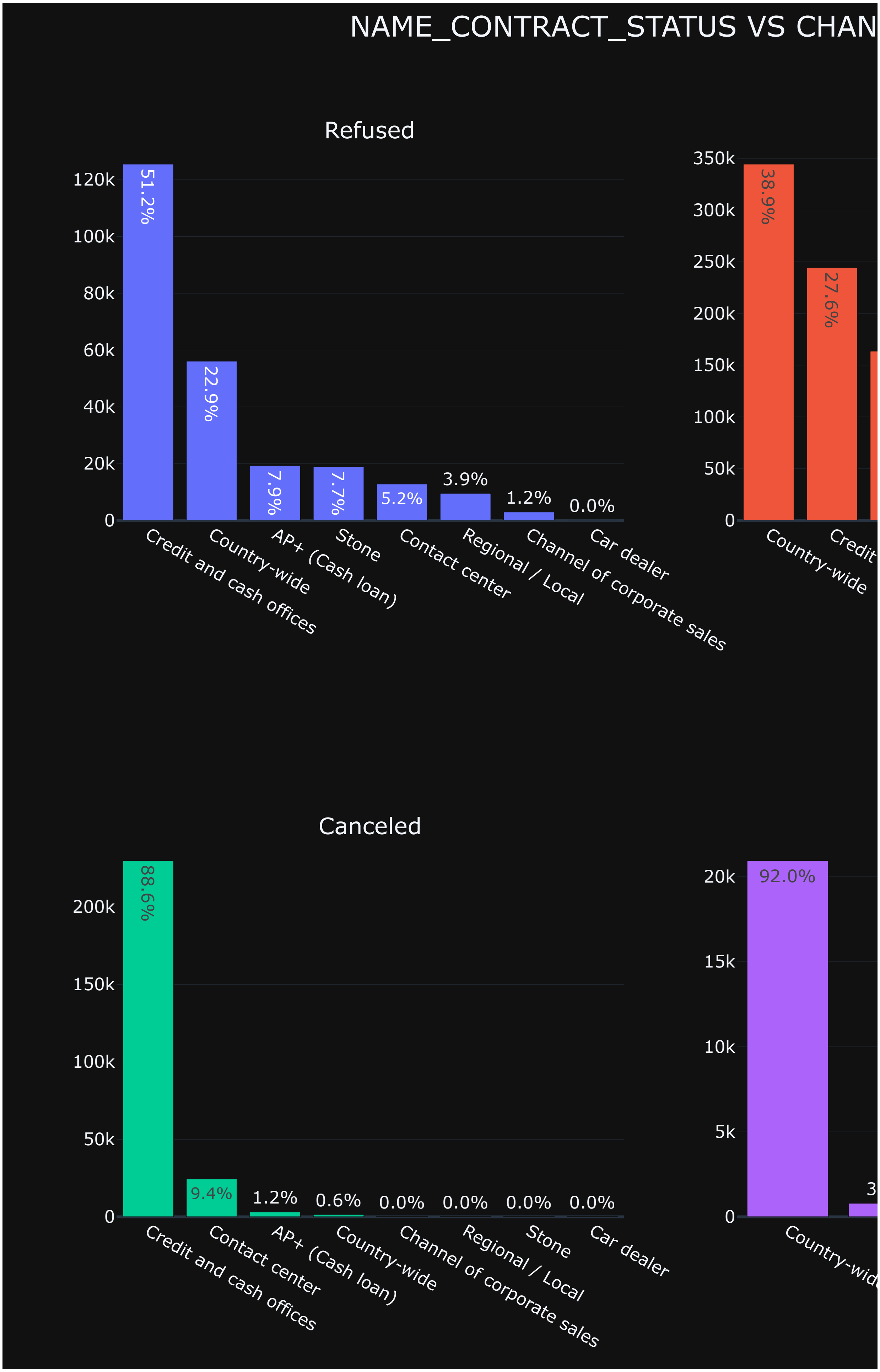
Insights

- Most of the approved loans belong to **POS hosehold with interest** & **POS mobile with interest** product combination.
- 15% refused loans belong to **Cash X-Sell: low** product combination.
- Most of the canceled loans belong to **Cash** category.
- 81.3% **Unused Offer** loans belong to POS mobile with interest.

In [114]:
```
plot_merge(appdata_merge,'NAME_PAYMENT_TYPE')
```


NAME_CONTRACT_STATUS VS NAME_PA
Refused
99.2%
0.6%
0.1%
150k
100k
50k
0
Cash through the bank
Non-cash from your account
Cashless from the account of the employer
Appr
99.0%
600k
400k
200k
0
Cash through the ba
Canceled
98.3%
1.2%
0.6%
2500
2000
1500
1000
500
0
Cash through the bank
Non-cash from your account
Cashless from the account of the employer
Unuse
99.8%
20k
15k
10k
5k
0
Cash through the ba

In [115]:
```
plot_merge(appdata_merge,'CHANNEL_TYPE')
```

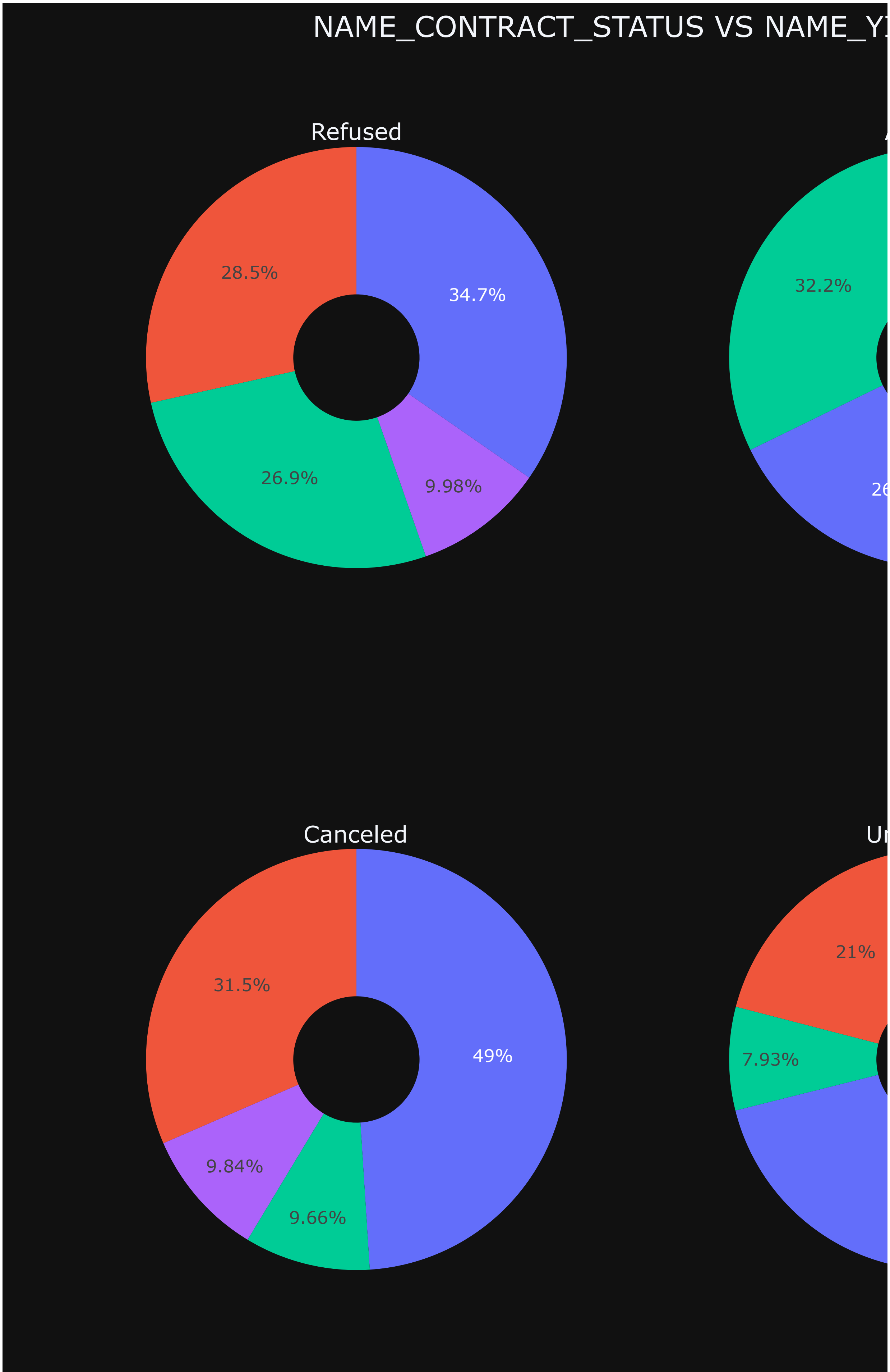


NAME_CONTRACT_STATUS VS CHAN
Refused
51.2%
22.9%
7.9%
7.7%
5.2%
3.9%
1.2%
0.0%
0
20k
40k
60k
80k
100k
120k
Credit and cash offices
Country-wide
AP+ (Cash loan)
Stone
Contact center
Regional / Local
Channel of corporate sales
Car dealer
0
50k
100k
150k
200k
250k
300k
350k
38.9%
27.6%
Country-wide
Credit
Canceled
88.6%
9.4%
1.2%
0.6%
0.0%
0.0%
0.0%
0.0%
0
50k
100k
150k
200k
Credit and cash offices
Contact center
AP+ (Cash loan)
Country-wide
Channel of corporate sales
Regional / Local
Stone
Car dealer
0
5k
10k
15k
20k
92.0%
Country-wid

Insights

- Most of the approved loans belong to either **Country-wide** or **Credit & cash offices** channel type.
- More than 50% refused loans belong to **Credit & cash offices** channel type.
- **Credit & cash offices** channel type loans are getting canceled the most.
- More than 90% **Unused Offer** loans belong to Country-wide channel type.

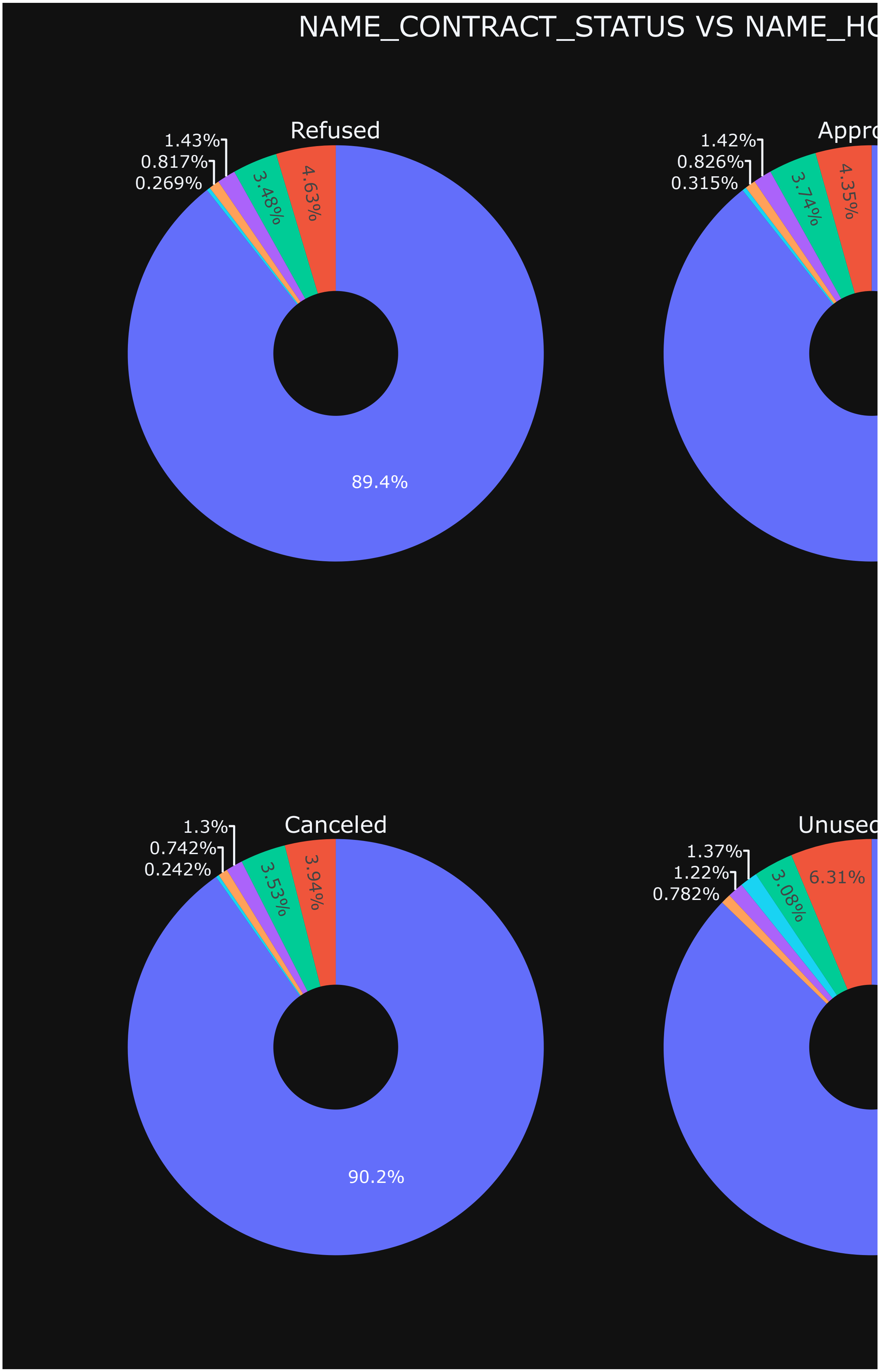
In [116]:
```
plot_pie_merge(appdata_merge,'NAME_YIELD_GROUP')
```


NAME_CONTRACT_STATUS VS NAME_Y
Refused
28.5%
34.7%
26.9%
9.98%
32.2%
Canceled
31.5%
49%
9.84%
9.66%
21%
7.93%

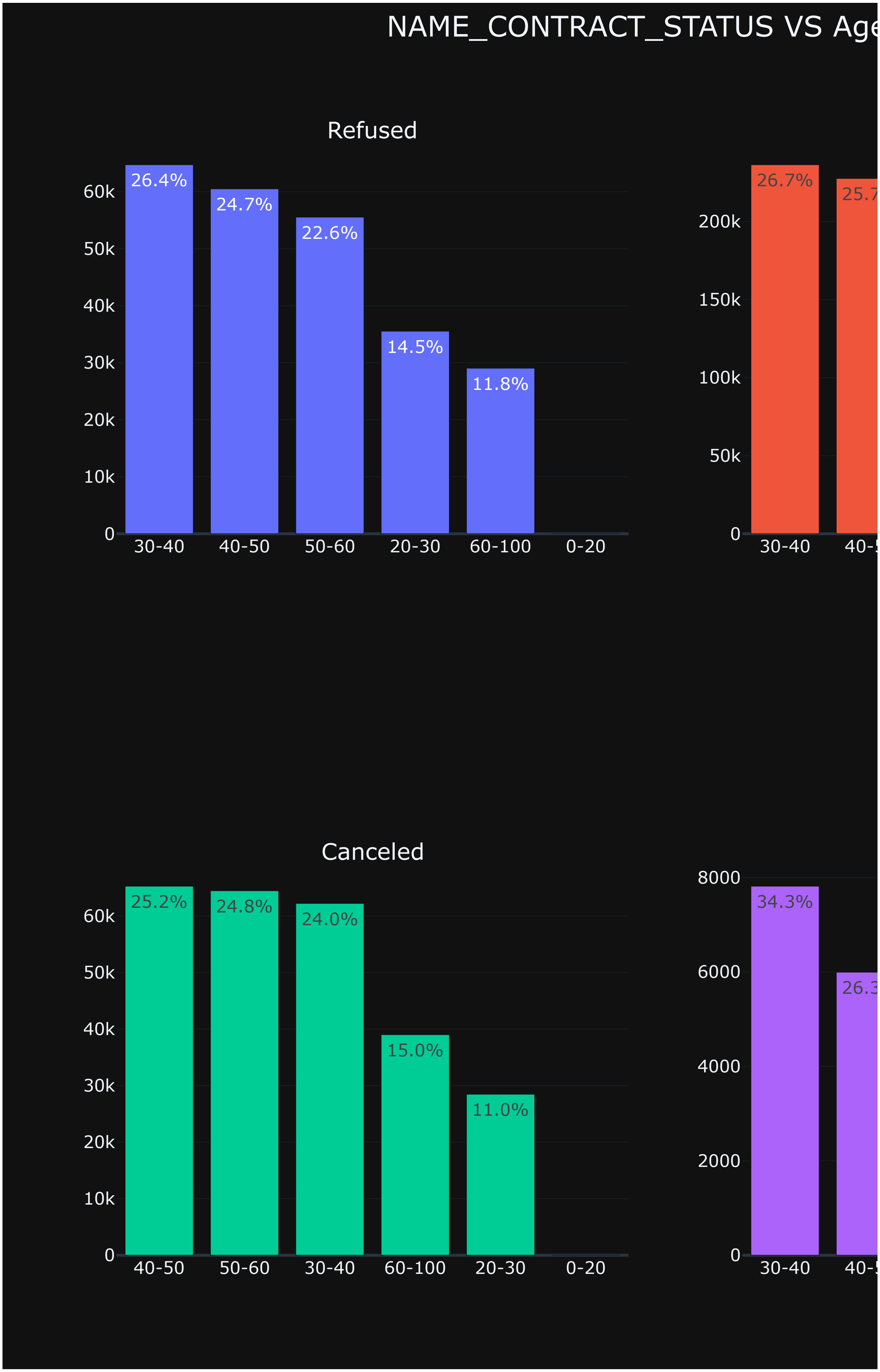
Insights

- Most of the approved loans have medium grouped interest rate.
- Loans with low or normal interest rate are getting refused or canceled the most.

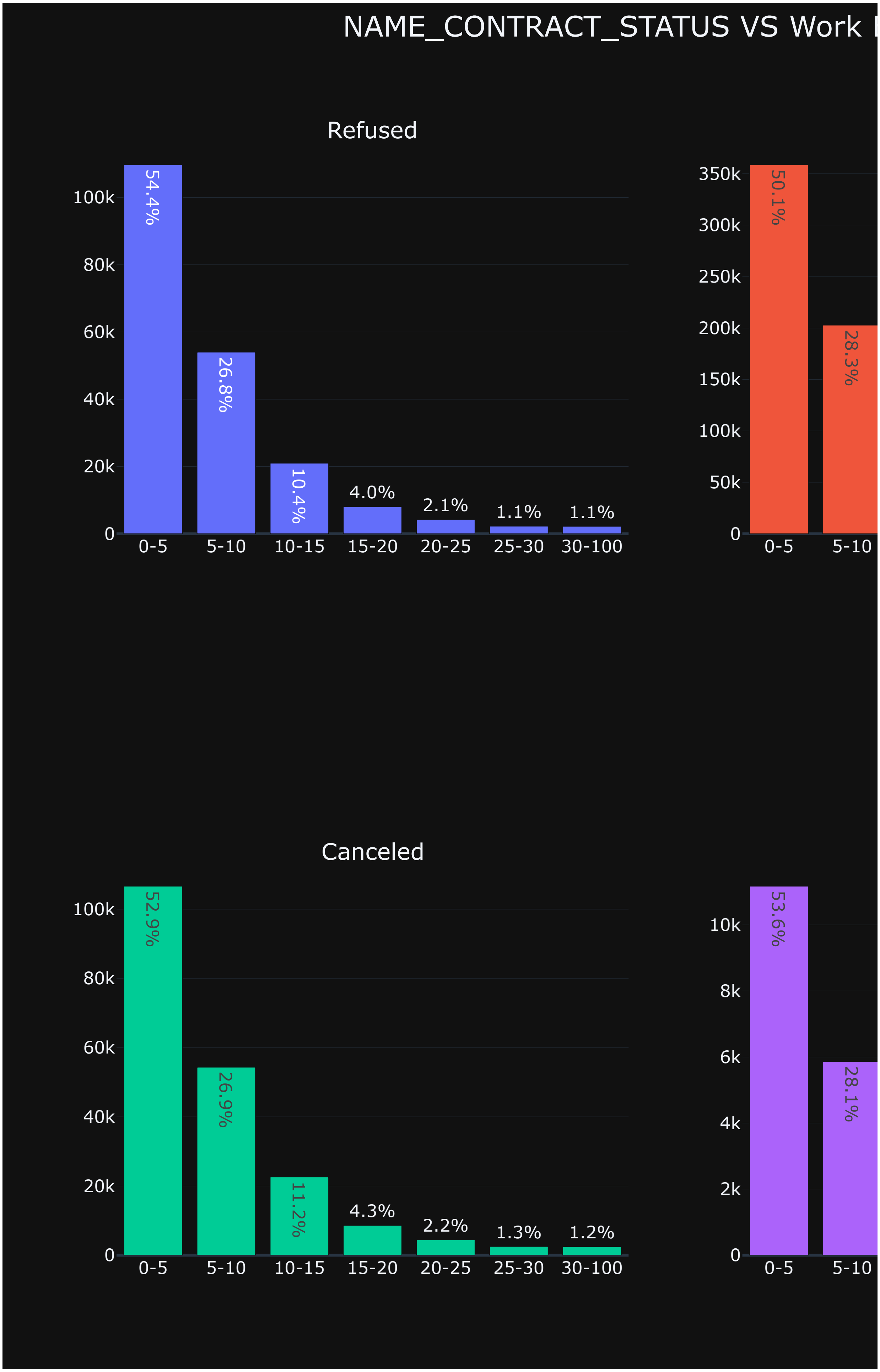
In [117]:
```
plot_pie_merge(appdata_merge,'NAME_HOUSING_TYPE')
```


NAME_CONTRACT_STATUS VS NAME_HO
Refused
1.43%
0.817%
0.269%
3.48%
4.63%
89.4%
Appro
1.42%
0.826%
0.315%
3.74%
4.35%
Canceled
1.3%
0.742%
0.242%
3.53%
3.94%
90.2%
Unused
1.37%
1.22%
0.782%
3.08%
6.31%

In [118]:
```
plot_merge(appdata_merge,'Age Group')
```


NAME_CONTRACT_STATUS VS Age
Refused
26.4%
24.7%
22.6%
14.5%
11.8%
60k
50k
40k
30k
20k
10k
0
30-40
40-50
50-60
20-30
60-100
0-20
26.7%
200k
150k
100k
50k
0
30-40
Canceled
25.2%
24.8%
24.0%
15.0%
11.0%
60k
50k
40k
30k
20k
10k
0
40-50
50-60
30-40
60-100
20-30
0-20
8000
6000
4000
2000
0
34.3%
30-40

In [119]:
```
plot_merge(appdata_merge,'Work Experience')
```


NAME_CONTRACT_STATUS VS Work
Refused
Canceled
54.4%
26.8%
10.4%
4.0%
2.1%
1.1%
1.1%
50.1%
28.3%
52.9%
26.9%
11.2%
4.3%
2.2%
1.3%
1.2%
53.6%
28.1%
0-5
5-10
10-15
15-20
20-25
25-30
30-100
0
20k
40k
60k
80k
100k
50k
150k
200k
250k
300k
350k
2k
4k
6k
8k
10k

In [120]:
```
plot_merge(appdata_merge,'AMT_CREDIT_Range')
```


NAME_CONTRACT_STATUS VS AMT_CR
Refused
22.9%
21.1%
20.3%
20.1%
15.7%
50k
40k
30k
20k
10k
0
Medium
Very Low
Low
High
Very High
21.2%
150k
100k
50k
0
Very Low
Canceled
21.3%
20.6%
20.6%
20.1%
17.5%
50k
40k
30k
20k
10k
0
Medium
Very Low
High
Low
Very High
5000
4000
3000
2000
1000
0
21.8%
Very Low

Insights

- Most of the approved loans belong to **Very Low** & **High** Credit range.
- **Medium & Very Low** credit range loans are canceled and rejected the most.

In [121]:
```
plot_merge(appdata_merge,'AMT_INCOME_TOTAL_Range')
```


NAME_CONTRACT_STATUS VS AMT_INCOM
Refused
70k
60k
50k
40k
30k
20k
10k
0
27.9%
26.1%
18.3%
15.6%
12.1%
High
Low
Very High
Very Low
Medium
250k
200k
150k
100k
50k
0
27.6%
Low
Canceled
28.0%
26.6%
17.9%
15.1%
12.4%
6000
5000
4000
3000
2000
1000
28.1%

Insights

- Most of the loans are getting approved for Applicants with **Low** Income range. May be they are opting for low credit loans.
- Almost 28% loan applications are either getting rejected or canceled even though applicant belong to HIGH Income range. May be they have requested for quite HIGH credit range.

END