Import Libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
pd.options.display.float_format = "{:.2f}".format
sns.set_theme(color_codes=True)
```

Import Dataset

In [ ]:
```
df = pd.read_csv("application_data.csv")
df
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	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	
...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

307511 rows × 122 columns

DATA UNDERSTANDING

#Top 3 Rows

In [ ]:
```
df.head(3)
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_C
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	

3 rows × 122 columns

#Find Shape

In [ ]:
```
df.shape
```

Out[ ]:
```
(307511, 122)
```

#Find Information

In [ ]:
```
df.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
```

#Finding Data Types

In [ ]:
```
df.dtypes
```

Out[ ]:
```
SK_ID_CURR                     int64
TARGET                         int64
NAME_CONTRACT_TYPE            object
CODE_GENDER                   object
FLAG_OWN_CAR                  object
                               ...
AMT_REQ_CREDIT_BUREAU_DAY     float64
AMT_REQ_CREDIT_BUREAU_WEEK    float64
AMT_REQ_CREDIT_BUREAU_MON     float64
AMT_REQ_CREDIT_BUREAU_QRT     float64
AMT_REQ_CREDIT_BUREAU_YEAR    float64
Length: 122, dtype: object
```

In [ ]:
```
df.select_dtypes(include="object").columns
```

Out[ ]:
```
Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
       'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE', 'FONDKAPREMONT_MODE',
       'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE'],
      dtype='object')
```

In [ ]:
```
df.select_dtypes(include="int").columns
```

Out[ ]:
```
Index(['SK_ID_CURR', 'TARGET', 'CNT_CHILDREN', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
       'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE',
       'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', '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', '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'],
      dtype='object')
```

In [ ]:
```
df.select_dtypes(include="float").columns
```

Out[ ]:
```
Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
       'REGION_POPULATION_RELATIVE', 'DAYS_REGISTRATION', 'OWN_CAR_AGE',
       'CNT_FAM_MEMBERS', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       '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',
       'TOTALAREA_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE',
       'DEF_30_CNT_SOCIAL_CIRCLE', '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'],
      dtype='object')
```

#Statistical Summary

In [ ]:
```
df.describe()
```

Out[ ]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_A
count	307511.00	307511.00	307511.00	307511.00	307511.00	3
mean	278180.52	0.08	0.42	168797.92	599026.00	
std	102790.18	0.27	0.72	237123.15	402490.78	
min	100002.00	0.00	0.00	25650.00	45000.00	
25%	189145.50	0.00	0.00	112500.00	270000.00	
50%	278202.00	0.00	0.00	147150.00	513531.00	
75%	367142.50	0.00	1.00	202500.00	808650.00	
max	456255.00	1.00	19.00	117000000.00	4050000.00	2

8 rows × 106 columns

In [ ]:
```
df.describe(include="object").T
```

Out[ ]:

	count	unique	top	freq
NAME_CONTRACT_TYPE	307511	2	Cash loans	278232
CODE_GENDER	307511	3	F	202448
FLAG_OWN_CAR	307511	2	N	202924
FLAG_OWN_REALTY	307511	2	Y	213312
NAME_TYPE_SUITE	306219	7	Unaccompanied	248526
NAME_INCOME_TYPE	307511	8	Working	158774
NAME_EDUCATION_TYPE	307511	5	Secondary / secondary special	218391
NAME_FAMILY_STATUS	307511	6	Married	196432
NAME_HOUSING_TYPE	307511	6	House / apartment	272868
OCCUPATION_TYPE	211120	18	Laborers	55186
WEEKDAY_APPR_PROCESS_START	307511	7	TUESDAY	53901
ORGANIZATION_TYPE	307511	58	Business Entity Type 3	67992
FONDKAPREMONT_MODE	97216	4	reg oper account	73830
HOUSETYPE_MODE	153214	3	block of flats	150503
WALLSMATERIAL_MODE	151170	7	Panel	66040
EMERGENCYSTATE_MODE	161756	2	No	159428

#Checking Duplicate Values

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
0
```

#Finding Null Values

In [ ]:
```
df.isna().sum().sort_values(ascending=False)
```

Out[ ]:
```
COMMONAREA_MEDI             214865
COMMONAREA_AVG              214865
COMMONAREA_MODE             214865
NONLIVINGAPARTMENTS_MODE    213514
NONLIVINGAPARTMENTS_AVG     213514
                             ...
NAME_HOUSING_TYPE                0
NAME_FAMILY_STATUS               0
NAME_EDUCATION_TYPE              0
NAME_INCOME_TYPE                 0
SK_ID_CURR                       0
Length: 122, dtype: int64
```

DATA PREPROCESSING

#Drop Unnecessary Columns

In [ ]:
```
empty_list = []
for i in df.columns:
    if i.startswith("FLAG"):
        empty_list.append(i)
empty_list
```

Out[ ]:
```
['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 [ ]:
```
df.drop(columns=empty_list,inplace=True)
df
```

Out[ ]:

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

307511 rows × 94 columns

#Dealing With Missing Values (Columns 10 %)

In [ ]:

```
empty_col = df.isna().sum()
empty_col = empty_col.loc[empty_col.values>df.shape[0]*0.10]
df = df.drop(columns=empty_col.index)
df
```

Out[ ]:

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

307511 rows × 37 columns

#Dealing With Missing Values (Rows 10 %)

In [ ]:
```
df = df.dropna(thresh=df.shape[1]-df.shape[1]*0.10)
df
```

Out[ ]:

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

306490 rows × 37 columns

#Dealing With XNA Gender

In [ ]:
```
df["CODE_GENDER"].value_counts()
```

Out[ ]:
```
F      201793
M      104693
XNA         4
Name: CODE_GENDER, dtype: int64
```

In [ ]:
```
df.loc[df["CODE_GENDER"] == "XNA","CODE_GENDER"] = "F"
df["CODE_GENDER"].value_counts()
```

Out[ ]:
```
F    201797
M    104693
Name: CODE_GENDER, dtype: int64
```

#Dealing With Organization Type

In [ ]:
```
df["ORGANIZATION_TYPE"].value_counts().head()
```

Out[ ]:
```
Business Entity Type 3    67726
XNA                       55202
Self-employed             38293
Other                     16635
Medicine                  11150
Name: ORGANIZATION_TYPE, dtype: int64
```

In [ ]:
```
df = df.loc[df["ORGANIZATION_TYPE"]!="XNA"]
df
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	CNT_CHILDREN	A
0	100002	1	Cash loans	M	0	
1	100003	0	Cash loans	F	0	
2	100004	0	Revolving loans	M	0	
3	100006	0	Cash loans	F	0	
4	100007	0	Cash loans	M	0	
...	...	...	...	...	...	
307504	456248	0	Cash loans	F	0	
307506	456251	0	Cash loans	M	0	
307508	456253	0	Cash loans	F	0	
307509	456254	1	Cash loans	F	0	
307510	456255	0	Cash loans	F	0	

251288 rows × 37 columns

#Finding Outliers And Deleting Them In Income Column

In [ ]:
```
q1,q2,q3 = np.percentile(df["AMT_INCOME_TOTAL"],[25,50,75])
iqr = q3-q1
Lower_extreme = q1-1.5*iqr
Upper_extreme = q3+1.5*iqr
df = df.loc[(df["AMT_INCOME_TOTAL"]<=Upper_extreme) & (df["AMT_INCOME_TOTAL"]>=Lowe
df
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	CNT_CHILDREN	A
0	100002	1	Cash loans	M	0	
1	100003	0	Cash loans	F	0	
2	100004	0	Revolving loans	M	0	
3	100006	0	Cash loans	F	0	
4	100007	0	Cash loans	M	0	
...	...	...	...	...	...	
307504	456248	0	Cash loans	F	0	
307506	456251	0	Cash loans	M	0	
307508	456253	0	Cash loans	F	0	
307509	456254	1	Cash loans	F	0	
307510	456255	0	Cash loans	F	0	

242627 rows × 37 columns

#Creating Income Range

In [ ]:

```
bins = np.arange(0,df["AMT_INCOME_TOTAL"].max()+25000,25000)

slots = ['0-25000', '25000-50000','50000-75000','75000-100000','100000-125000', '12
       '200000-225000','225000-250000','250000-275000','275000-300000','300000-3250

df["AMT_INCOME_RANGE"]= pd.cut(df["AMT_INCOME_TOTAL"],bins=bins,labels=slots)
df
```

```
C:\Users\varsh\AppData\Local\Temp\ipykernel_14560\3152651353.py:6: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df["AMT_INCOME_RANGE"]= pd.cut(df["AMT_INCOME_TOTAL"],bins=bins,labels=slots)
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	CNT_CHILDREN	A
0	100002	1	Cash loans	M	0	
1	100003	0	Cash loans	F	0	
2	100004	0	Revolving loans	M	0	
3	100006	0	Cash loans	F	0	
4	100007	0	Cash loans	M	0	
...	...	...	...	...	...	
307504	456248	0	Cash loans	F	0	
307506	456251	0	Cash loans	M	0	
307508	456253	0	Cash loans	F	0	
307509	456254	1	Cash loans	F	0	
307510	456255	0	Cash loans	F	0	

242627 rows × 38 columns

EXPLORATORY DATA ANALYSIS

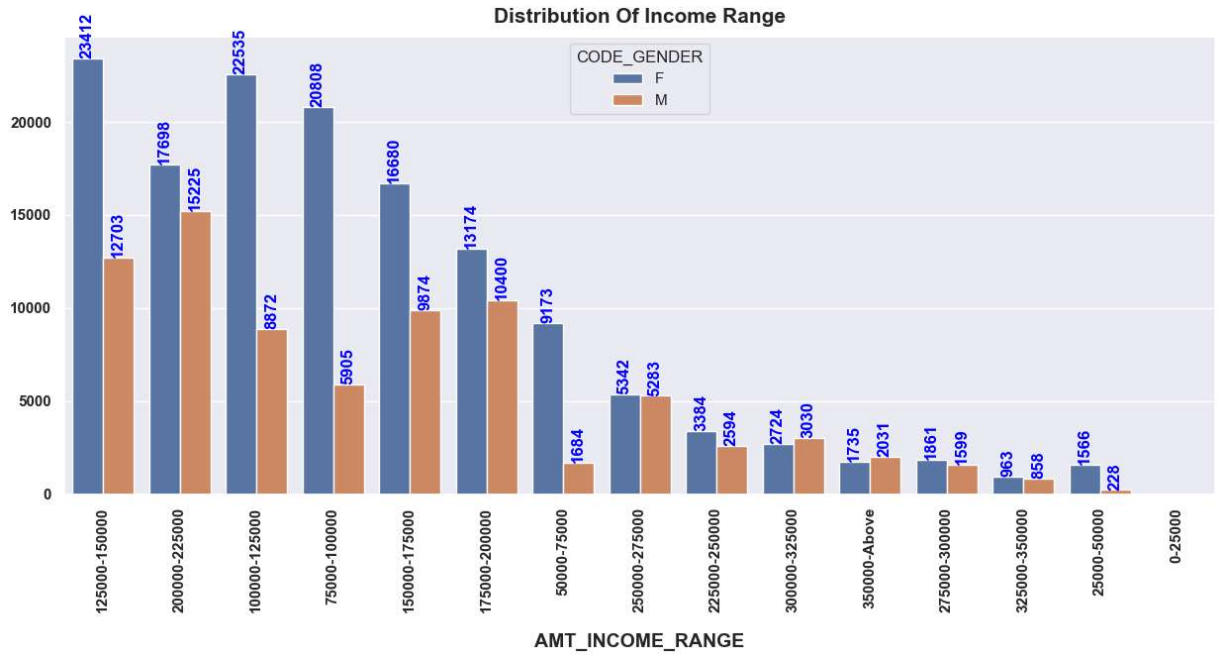
#Filtering Target 0 And 1

In [ ]:
```
df0 = df.loc[df["TARGET"]==0]
df1= df.loc[df["TARGET"]==1]
```

ANALYSIS FOR TARGET 0

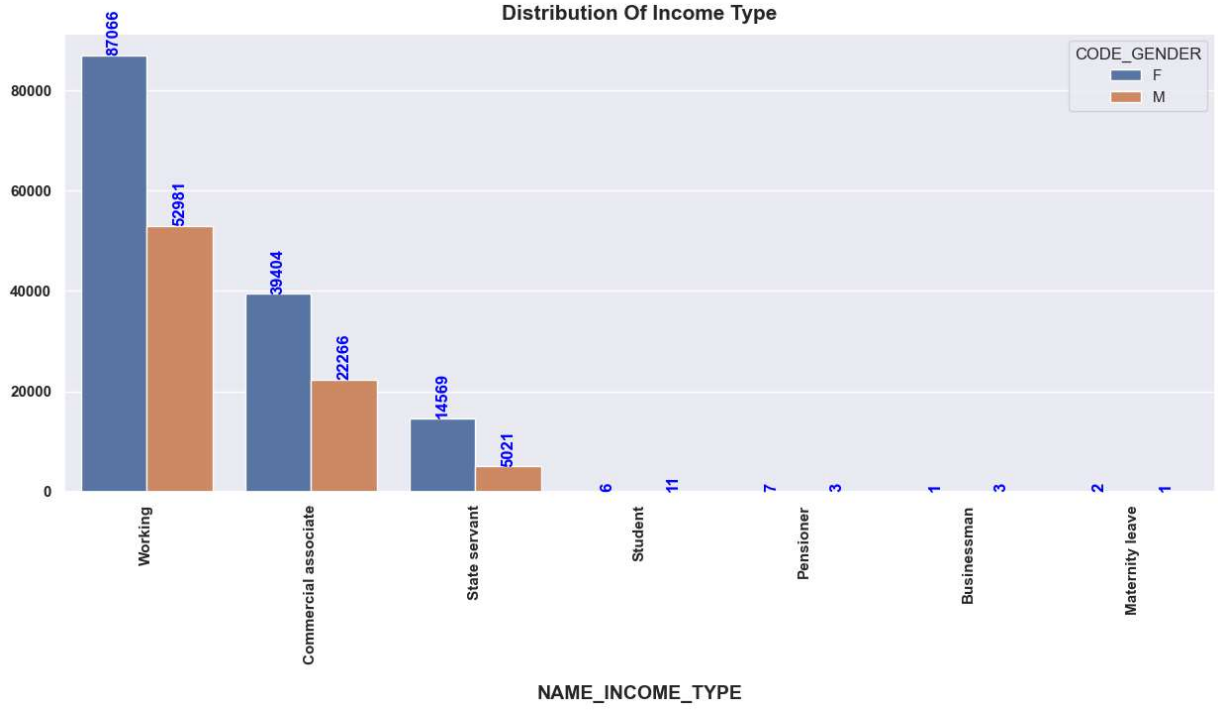
#Distribution Of Income Range

In [ ]:
```
plt.figure(figsize=(15,6))
ax = sns.countplot(data=df0,x=df0["AMT_INCOME_RANGE"], hue=df0["CODE_GENDER"] ,orde
plt.title("Distribution Of Income Range",weight="bold",fontsize=15,pad=10)
plt.xticks(rotation =90,weight="bold")
plt.yticks(weight="bold")
plt.xlabel("AMT_INCOME_RANGE",weight="bold",fontsize=14,labelpad=15)
plt.ylabel("",weight="bold",fontsize=12,labelpad=15)
for i in ax.containers:
    i.datavalues
    ax.bar_label(i,label_type="edge",weight="bold",color ="blue",rotation=90)
plt.show()
```



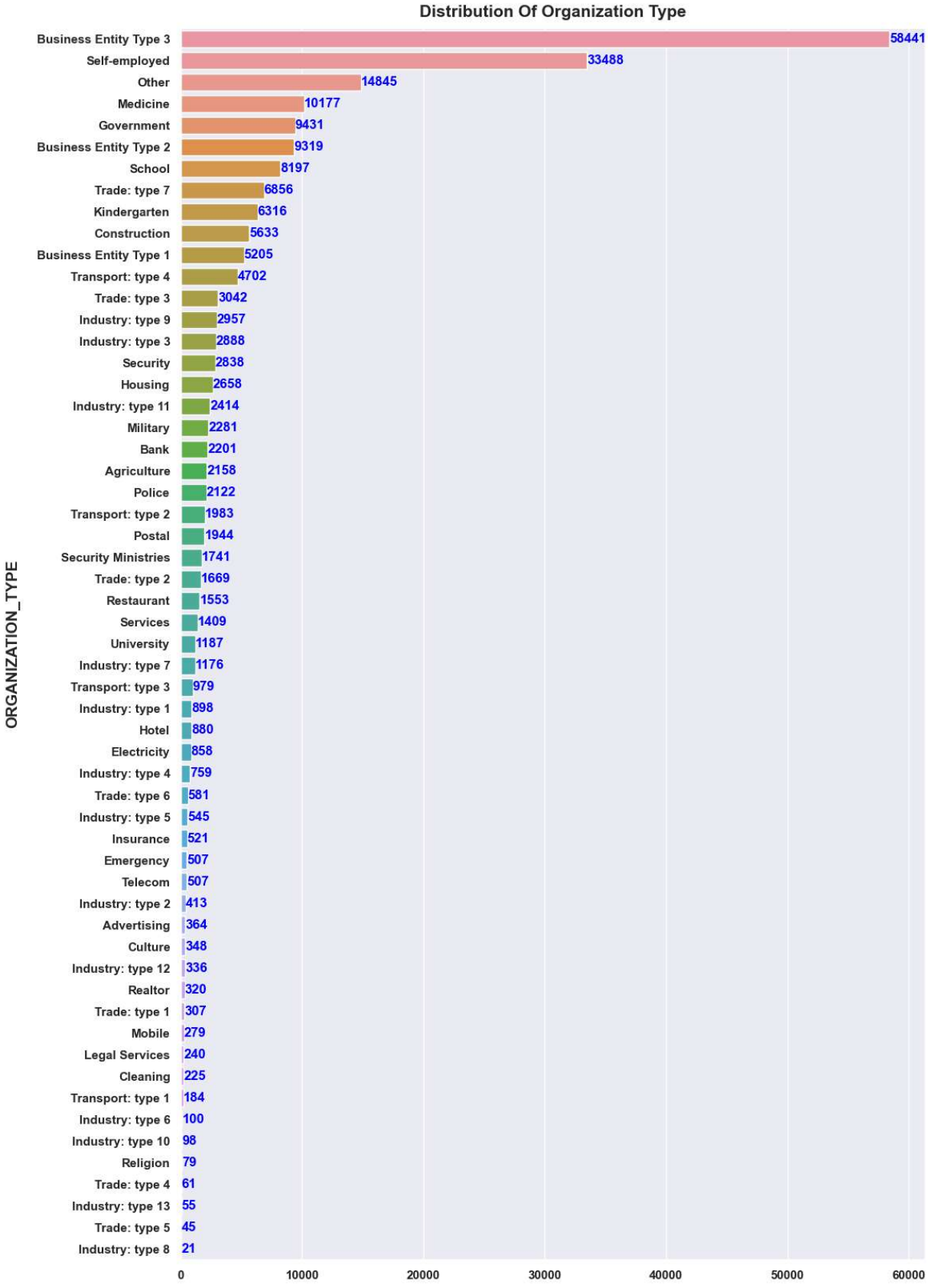
#Distribution Of Income Type

In [ ]:
```
plt.figure(figsize=(15,6))
ax = sns.countplot(data=df0,x=df0["NAME_INCOME_TYPE"], hue=df0["CODE_GENDER"] ,orde
plt.title("Distribution Of Income Type",weight="bold",fontsize=15,pad=10)
plt.xticks(rotation =90,weight="bold")
plt.yticks(weight="bold")
plt.xlabel("NAME_INCOME_TYPE",weight="bold",fontsize=14,labelpad=15)
plt.ylabel("",weight="bold",fontsize=12,labelpad=15)
for i in ax.containers:
    i.datavalues
    ax.bar_label(i,label_type="edge",weight="bold",color ="blue",rotation=90)
plt.show()
```



#Distribution Of Organization Type

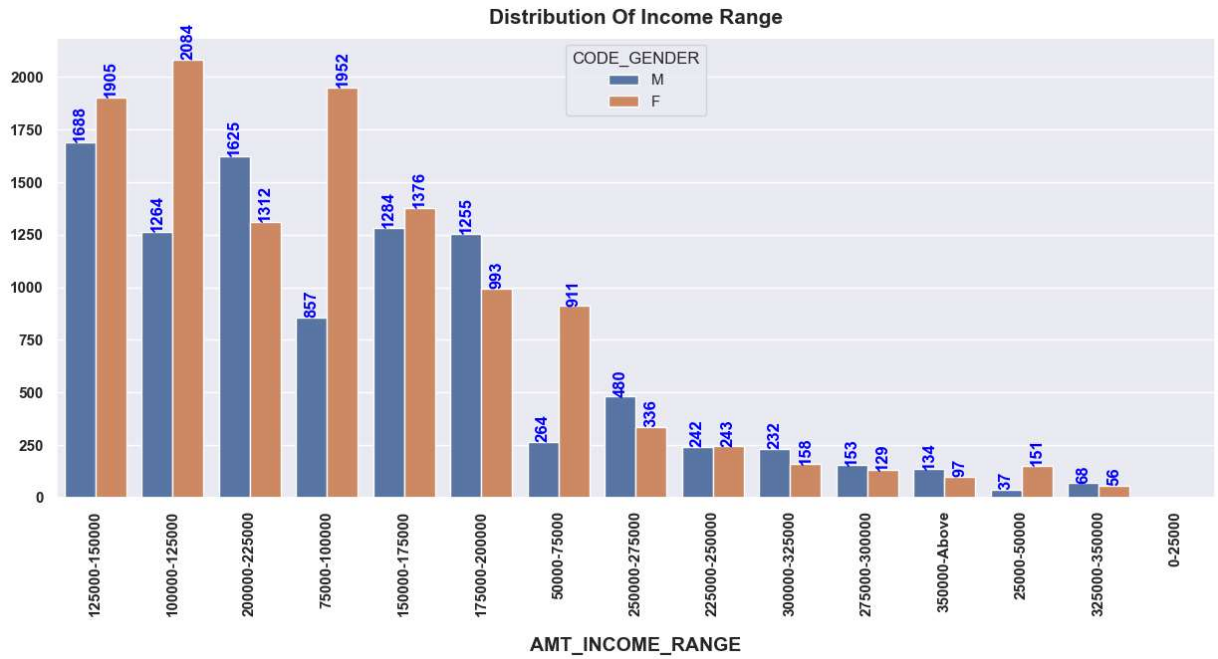
In [ ]:
```
plt.figure(figsize=(12,20))
ax = sns.countplot(data=df0,y=df0["ORGANIZATION_TYPE"],order = df0["ORGANIZATION_TY
plt.title("Distribution Of Organization Type",weight="bold",fontsize=15,pad=10)
plt.xticks(weight="bold")
plt.yticks(weight="bold")
plt.ylabel("ORGANIZATION_TYPE",weight="bold",fontsize=14,labelpad=15)
plt.xlabel("",weight="bold",fontsize=12,labelpad=15)
for i in ax.containers:
    i.datavalues
    ax.bar_label(i,label_type="edge",weight="bold",color ="blue")
plt.show()
```



ANALYSIS FOR TARGET 1

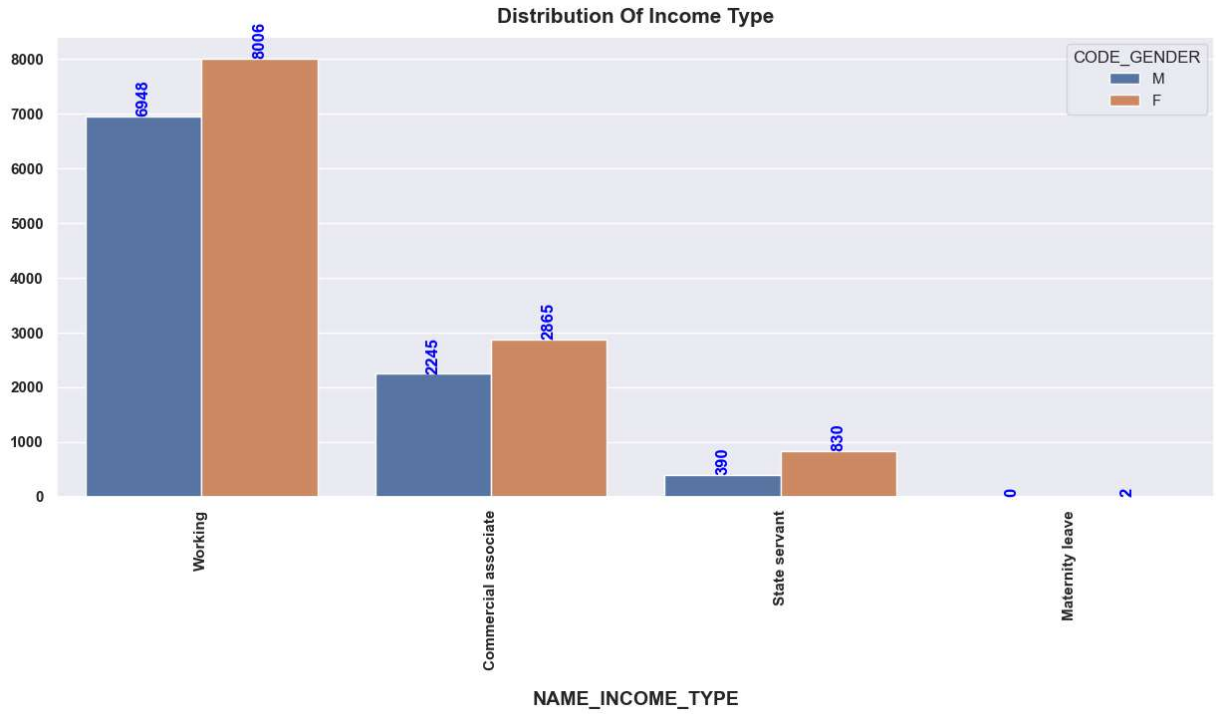
#Distribution Of Income Range

In [ ]:
```
plt.figure(figsize=(15,6))
ax = sns.countplot(data=df1,x=df1["AMT_INCOME_RANGE"], hue=df1["CODE_GENDER"] ,orde
plt.title("Distribution Of Income Range",weight="bold",fontsize=15,pad=10)
plt.xticks(rotation =90,weight="bold")
plt.yticks(weight="bold")
plt.xlabel("AMT_INCOME_RANGE",weight="bold",fontsize=14,labelpad=15)
plt.ylabel("",weight="bold",fontsize=12,labelpad=15)
for i in ax.containers:
    i.datavalues
    ax.bar_label(i,label_type="edge",weight="bold",color ="blue",rotation=90)
plt.show()
```



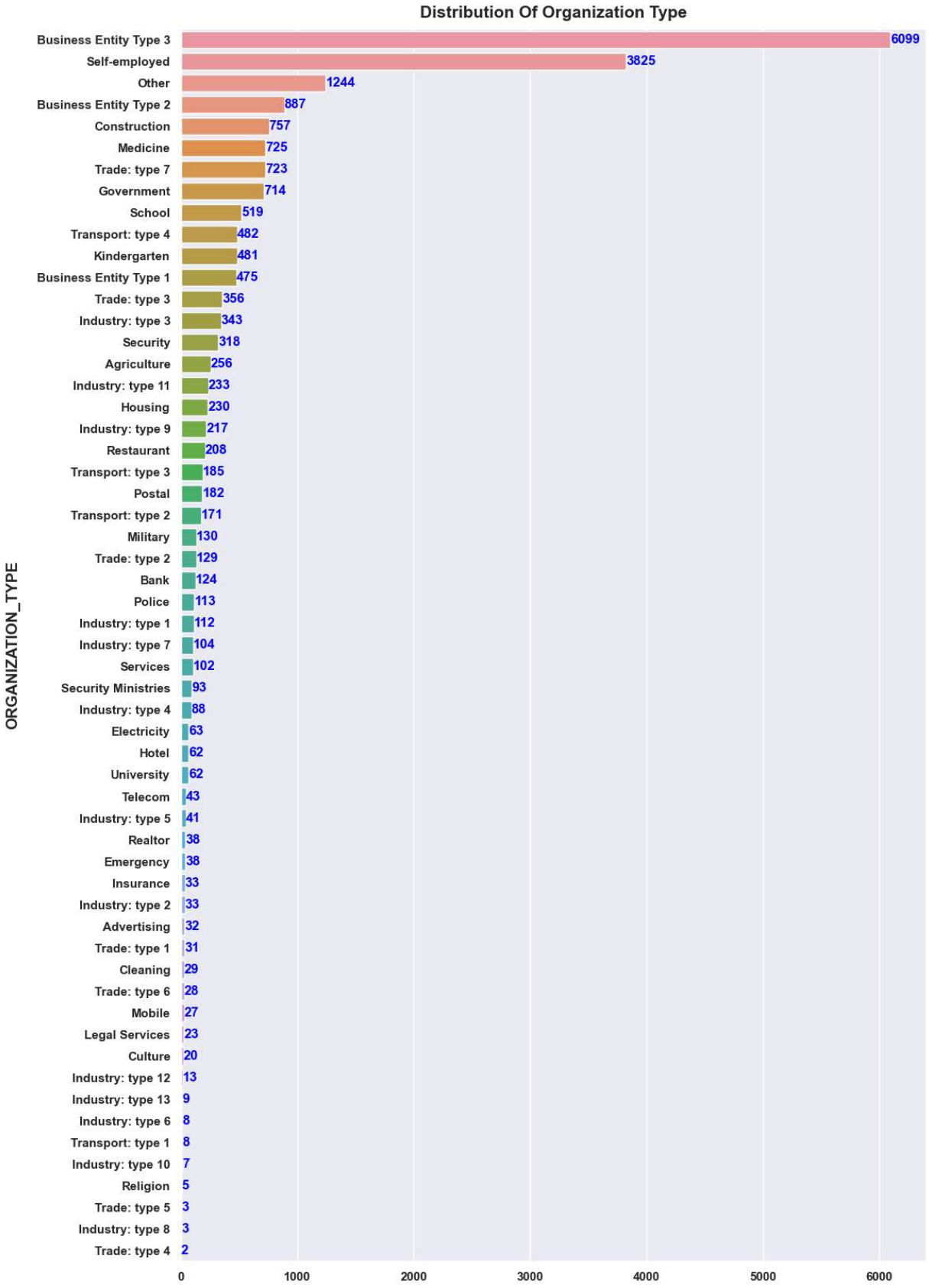
#Distribution Of Income Type

In [ ]:
```
plt.figure(figsize=(15,6))
ax = sns.countplot(data=df1,x=df1["NAME_INCOME_TYPE"], hue=df1["CODE_GENDER"] ,orde
plt.title("Distribution Of Income Type",weight="bold",fontsize=15,pad=10)
plt.xticks(rotation =90,weight="bold")
plt.yticks(weight="bold")
plt.xlabel("NAME_INCOME_TYPE",weight="bold",fontsize=14,labelpad=15)
plt.ylabel("",weight="bold",fontsize=12,labelpad=15)
for i in ax.containers:
    i.datavalues
    ax.bar_label(i,label_type="edge",weight="bold",color ="blue",rotation=90)
plt.show()
```



#Distribution Of Organization Type

In [ ]:
```
plt.figure(figsize=(12,20))
ax = sns.countplot(data=df1,y=df1["ORGANIZATION_TYPE"],order = df1["ORGANIZATION_TY
plt.title("Distribution Of Organization Type",weight="bold",fontsize=15,pad=10)
plt.xticks(weight="bold")
plt.yticks(weight="bold")
plt.ylabel("ORGANIZATION_TYPE",weight="bold",fontsize=14,labelpad=15)
plt.xlabel("",weight="bold",fontsize=12,labelpad=15)
for i in ax.containers:
    i.datavalues
    ax.bar_label(i,label_type="edge",weight="bold",color ="blue")
plt.show()
```



DATA PREPROCESSING

#Finding Outliers And Deleting Them In Credit Column

In [ ]:
```
q1,q2,q3 = np.percentile(df["AMT_CREDIT"],[25,50,75])
iqr = q3-q1
lower_extreme = q1-1.5*iqr
upper_extreme = q3+1.5*iqr
df = df.loc[(df["AMT_CREDIT"]>=lower_extreme) & (df["AMT_CREDIT"]<=upper_extreme)]
df
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	CNT_CHILDREN	A
0	100002	1	Cash loans	M	0	
1	100003	0	Cash loans	F	0	
2	100004	0	Revolving loans	M	0	
3	100006	0	Cash loans	F	0	
4	100007	0	Cash loans	M	0	
...	...	...	...	...	...	
307504	456248	0	Cash loans	F	0	
307506	456251	0	Cash loans	M	0	
307508	456253	0	Cash loans	F	0	
307509	456254	1	Cash loans	F	0	
307510	456255	0	Cash loans	F	0	

238102 rows × 38 columns

#Creating Credit Range

In [ ]:
```
bins = np.arange(0,df["AMT_CREDIT"].max()+250000,250000)
slots = ("0-250000","250000-500000","500000-750000","750000-1000000","1000000-12500
         "1500000-1750000")
df["AMT_CREDIT_RANGE"] = pd.cut(df["AMT_CREDIT"],bins= bins,labels=slots)
```

```
C:\Users\varsh\AppData\Local\Temp\ipykernel_14560\3606332851.py:4: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df["AMT_CREDIT_RANGE"] = pd.cut(df["AMT_CREDIT"],bins= bins,labels=slots)
```

EXPLORATORY DATA ANALYSIS

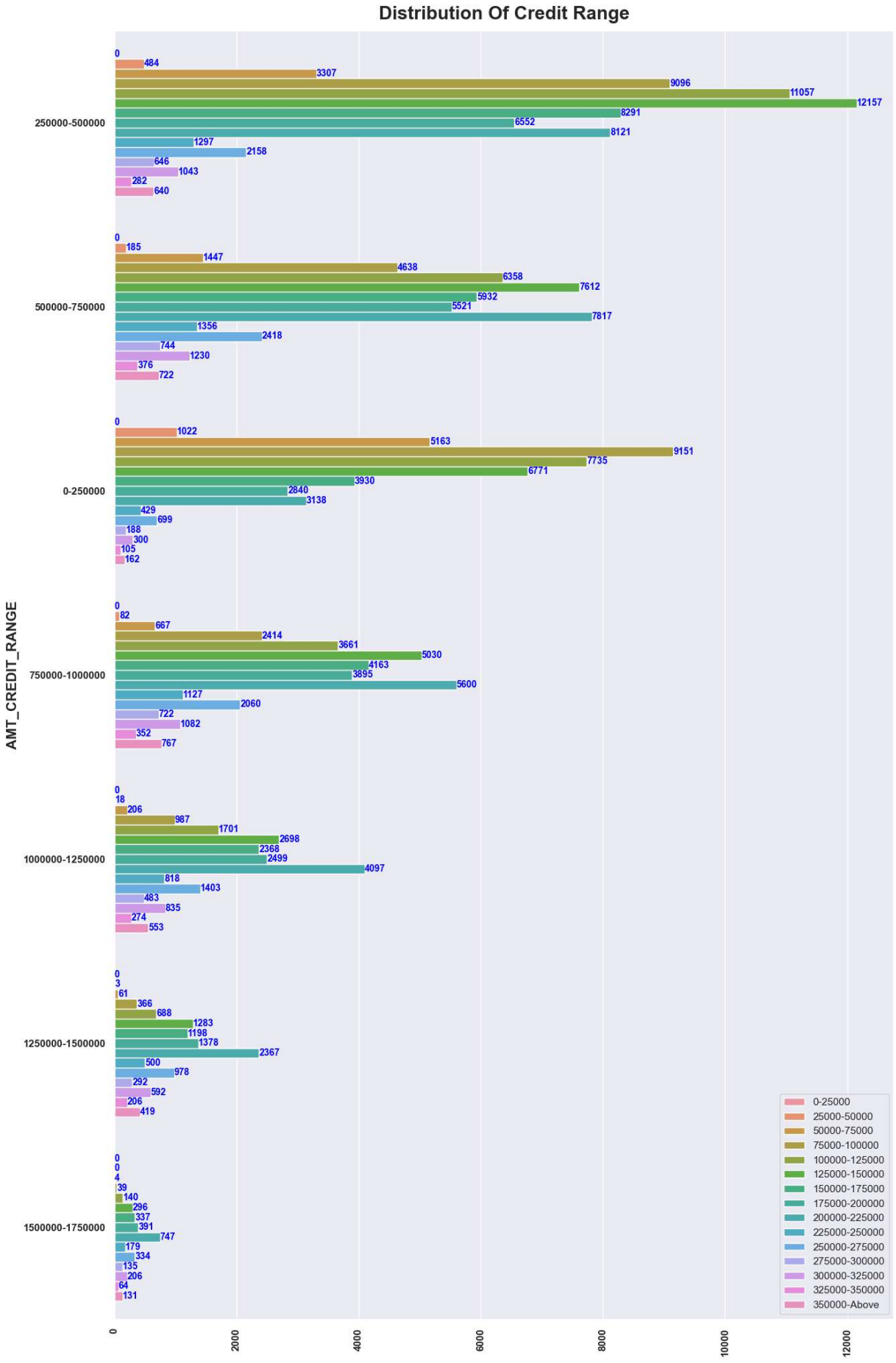
#Filtering Target 0 And 1

In [ ]:
```
df0 = df.loc[df["TARGET"]==0]
df1= df.loc[df["TARGET"]==1]
```

ANALYSIS FOR TARGET 0

#Distribution Of Credit Range

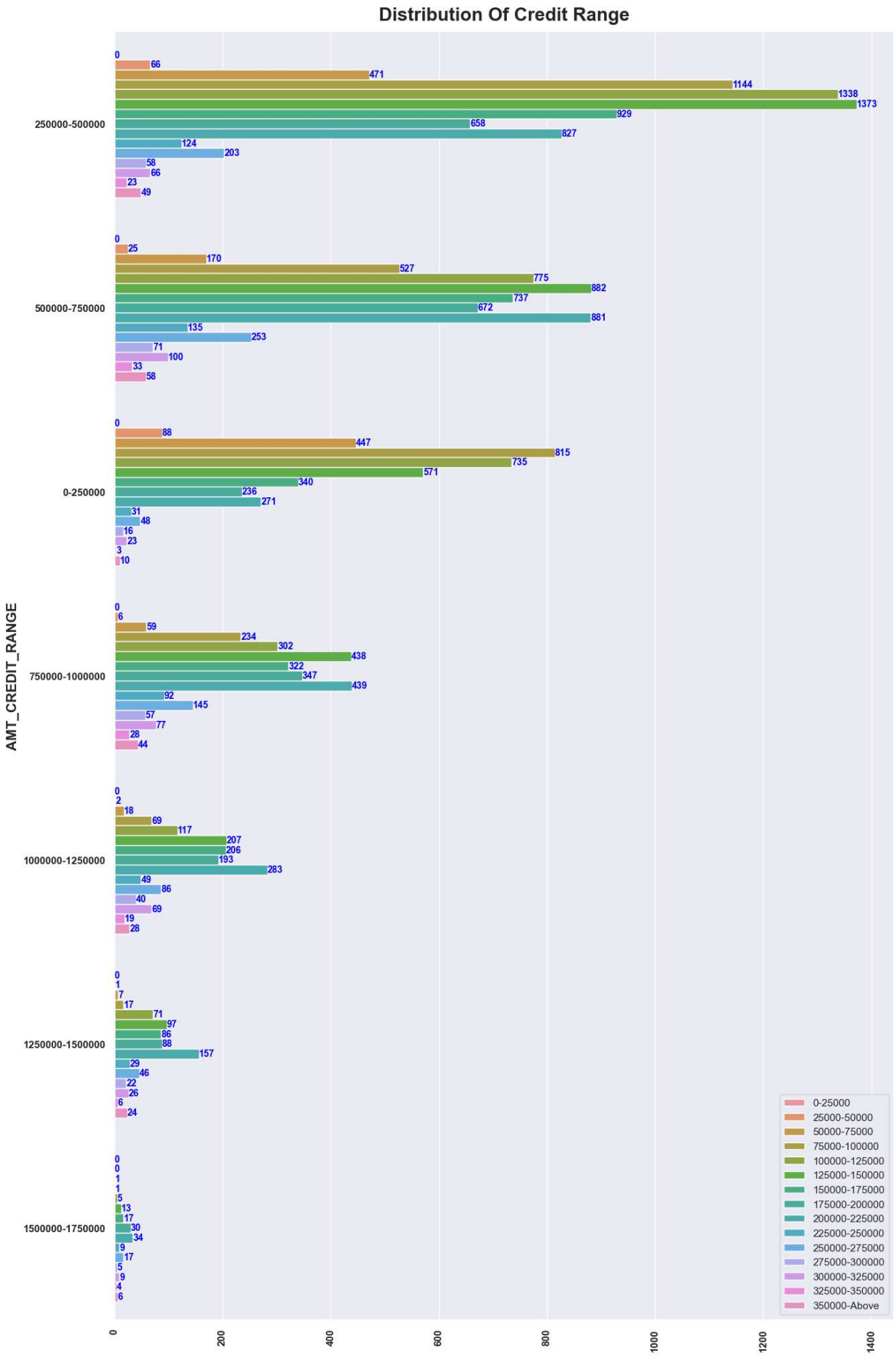
In [ ]:
```
plt.figure(figsize=(15,25))
ax = sns.countplot(data=df0,y=df0["AMT_CREDIT_RANGE"],order=df0["AMT_CREDIT_RANGE"]
plt.title("Distribution Of Credit Range",weight="bold",pad=12,fontsize=20)
plt.ylabel("AMT_CREDIT_RANGE",weight="bold",fontsize=15)
plt.xlabel("")
plt.xticks(weight="bold",rotation=90)
plt.yticks(weight="bold")
plt.legend(loc=4)
for i in ax.containers:
    i.datavalues
    ax.bar_label(i,fontsize=10,weight="bold",color = "blue",label_type="edge")
plt.show()
```


Distribution Of Credit Range
AMT_CREDIT_RANGE
250000-500000
500000-750000
0-250000
750000-1000000
1000000-1250000
1250000-1500000
1500000-1750000
0-25000
25000-50000
50000-75000
75000-100000
100000-125000
125000-150000
150000-175000
175000-200000
200000-225000
225000-250000
250000-275000
275000-300000
300000-325000
325000-350000
350000-Above
0
2000
4000
6000
8000
10000
12000

ANALYSIS FOR TARGET 1

#Distribution Of Credit Range

In [ ]:
```
plt.figure(figsize=(15,25))
ax = sns.countplot(data=df1,y=df1["AMT_CREDIT_RANGE"],order=df1["AMT_CREDIT_RANGE"]
plt.title("Distribution Of Credit Range",weight="bold",pad=12,fontsize=20)
plt.ylabel("AMT_CREDIT_RANGE",weight="bold",fontsize=15)
plt.xlabel("")
plt.xticks(weight="bold",rotation=90)
plt.yticks(weight="bold")
plt.legend(loc=4)
for i in ax.containers:
    i.datavalues
    ax.bar_label(i,fontsize=10,weight="bold",color = "blue",label_type="edge")
plt.show()
```


Distribution Of Credit Range
AMT_CREDIT_RANGE
250000-500000
500000-750000
0-250000
750000-1000000
1000000-1250000
1250000-1500000
1500000-1750000
0-25000
25000-50000
50000-75000
75000-100000
100000-125000
125000-150000
150000-175000
175000-200000
200000-225000
225000-250000
250000-275000
275000-300000
300000-325000
325000-350000
350000-Above
0
200
400
600
800
1000
1200
1400