```
# ## Credit EDA Case Study
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, iplot
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
app_data =pd.read_csv("application_data.csv");
# print(app_data.head())

# # object class, total rows, columns, datatypes of all column (combine), memory
# print(app_data.info())

# # (total rows, total columns)
# print(app_data.shape) # 122 Columns

# # describes the static (mean, std, min, max, summary (numeric columns))
# print(app_data.describe())

# # Most of the columns are of type integer or float.
# print(app_data.dtypes.value_counts())

# ### Data Cleaning
# #### Dropping Columns with high percentage of NULL values

# # Percentage of NULL Values in descending order
(app_data.isnull().mean()*100).sort_values(ascending=False)
# l1 = list(app_data.columns)
# print(l1)

# # # Display the Columns with NULL Values greater than 40%
s1= (app_data.isnull().mean()*100).sort_values(ascending=False)[app_data.isnull().mean()*100 > 40]
# print(s1)

# plotly.express
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()

# Get Column names with NULL percentage greater than 40%
cols = (app_data.isnull().mean()*100 > 40)[app_data.isnull().mean()*100 > 40].index.tolist()
cols

# We are good to delete 49 columns because NULL percentage for these columns is greater than 40%
len(cols)

# Drop 49 columns
app_data.drop(columns=cols,inplace=True)

app_data.shape # 307511 rows & 73 Columns

# NULL Values percentage in new dataset
s2= (app_data.isnull().mean()*100).sort_values(ascending=False)
s2

s2.head(10)

# #### Imputation of Missing Values
app_data.head()

'''
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].mode()[0]))
    print('Imputing the missing value with : {0}'.format(app_data[i].mode()[0]))
    app_data[i].fillna(app_data[i].mode()[0],inplace=True)
    print('NULL Values in {0} after imputation : {1}'.format(i,app_data[i].isnull().sum()))
```

```
        print()

# Missing value percentage of remaining columns
(app_data.isnull().mean()*100).sort_values(ascending=False)

# __Impute missing values for OCCUPATION_TYPE__

# We can impute missing values in 'OCCUPATION_TYPE' column with 'Laborers'
fig=px.bar(app_data.OCCUPATION_TYPE.value_counts(),color=app_data.OCCUPATION_TYPE.value_counts())
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=0.95),
                 title_font_size=20,
                 showlegend=False,
                 height =450,
                )
fig.show()

app_data.OCCUPATION_TYPE.fillna('Laborers',inplace=True)

# __Impute Missing values (XNA) in CODE_GENDER with mode__

app_data['CODE_GENDER'].value_counts()

app_data['CODE_GENDER'].replace(to_replace='XNA',value=app_data['CODE_GENDER'].mode()[0],inplace=True)

app_data['CODE_GENDER'].value_counts()

# __Impute missing values for EXT_SOURCE_3__

app_data.EXT_SOURCE_3.dtype

app_data.EXT_SOURCE_3.fillna(app_data.EXT_SOURCE_3.median(),inplace=True)

# Percentage of missing values after imputation
(app_data.isnull().mean()*100).sort_values(ascending=False)

# Replace 'XNA' with NaN
app_data = app_data.replace('XNA',np.NaN)

# __DELETE all flag columns__

app_data.columns

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

```
# 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]]

# __Impute Missing values for AMT_ANNUITY & AMT_GOODS_PRICE__

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()))

app_data['AMT_ANNUITY'].fillna(app_data['AMT_ANNUITY'].median(),inplace=True)
app_data['AMT_GOODS_PRICE'].fillna(app_data['AMT_GOODS_PRICE'].median(),inplace=True)
app_data['AMT_ANNUITY'].isnull().sum()
app_data['AMT_GOODS_PRICE'].isnull().sum()

# #### Correcting Data

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].unique()))
        print('NULL Values in {0} column : {1}'.format(i,app_data[i].isnull().sum()))
        print()

app_data[days]

# Use absolute values in DAYS columns
app_data[days] = abs(app_data[days])
app_data[days]

# #### Binning

# 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', 'Medium' , 'High' , 'Very High'])
    print(app_data[i+'_Range'].value_counts())
    print()

app_data['YEARS_EMPLOYED']= app_data['DAYS_EMPLOYED']/365
app_data['Client_Age']= app_data['DAYS_BIRTH']/365

# Drop 'DAYS_EMPLOYED'& 'DAYS_BIRTH' column as we will be performing analysis on Year basis
app_data.drop(columns=['DAYS_EMPLOYED','DAYS_BIRTH'],inplace=True)

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','60-100']
```

```
                      )

app_data[['SK_ID_CURR','Client_Age','Age Group']]

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','25-30','30-100']
                      )

app_data[['SK_ID_CURR','YEARS_EMPLOYED','Work Experience']]

# ### Outlier Detection
# #### Analyzing AMT column for Outliers

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

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

#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

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()

app_data['CNT_CHILDREN'].value_counts()

app_data.shape[0]

# Remove all data points where CNT_CHILDREN is greater than 10
app_data= app_data[app_data['CNT_CHILDREN']<=10]
```

```
app_data.shape[0]

# Eight values dropped where number of children are greater than 10

# #### Analysing YEARS_EMPLOYED column for Outliers
sns.boxplot(y=app_data['YEARS_EMPLOYED'])

app_data['YEARS_EMPLOYED'].value_counts()

app_data.shape[0]

app_data['YEARS_EMPLOYED'][app_data['YEARS_EMPLOYED']>1000]=np.NaN

sns.boxplot(y=app_data['YEARS_EMPLOYED'])
plt.show()

app_data.isnull().sum().sort_values(ascending=False).head(10)

# #### Analyzing AMT_REQ_CREDIT columns for Outliers
cols = [i for i in  app_data.columns if 'AMT_REQ' in i]
cols

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

# Remove Outlier for AMT_REQ_CREDIT_BUREAU_QRT
app_data=app_data[app_data['AMT_REQ_CREDIT_BUREAU_QRT']<app_data['AMT_REQ_CREDIT_BUREAU_QRT'].max()]

# ### Univariate Analysis
app_data.columns

fig1=px.bar(app_data['OCCUPATION_TYPE'].value_counts(),color=app_data['OCCUPATION_TYPE'].value_counts())
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()

fig2=px.bar(app_data['ORGANIZATION_TYPE'].value_counts(),color=app_data['ORGANIZATION_TYPE'].value_counts())
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.

cols = ['Age Group','NAME_CONTRACT_TYPE', 'NAME_INCOME_TYPE','NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE','CODE_GENDER','Work Experience']

#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[count]].value_counts(normalize=True)*100).round(1).tolist()],
                    ),
                  row=i,col=j)
       count+=1
fig.update_layout(
                title=dict(text = "Analyze Categorical variables (Frequency / Percentage)",x=0.5,y=0.99),
                title_font_size=20,
                showlegend=False,
                width = 960,
                height = 1600,
              )
fig.show()

# __Insights__
#
# - Bank has recieved majority of the loan application from  __30-40__ & __40-50__ Age groups.
#
#
# - More than __50%__ of clients who have applied for loan belong to __Working Income Type__.
#
```

```
#
# - 88.7% clients with __Secondary/Secondary Special__ education type have applied for the loan.
#
#
# - Married people tend to apply more for loans. __63.9%__ clients who are have applied for loan are married.
#
#
# - Majority of the Clients who have applied for the loan have their own __house/apartment__.  Around 88.7% clients are owning either a house or an apartment.
#
#
# - __Female__ loan applications are more as compared to __males__. This may be because banks charge less rate of interest for females.
#
#
# - Clients with work experience between __0-5__ years have applied most for loan application.
#
#
# - 90.5% Applicants have requested for Cash loans

app_data.nunique().sort_values()

# #### Checking Imbalance
app_data['TARGET'].value_counts(normalize=True)

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()

app_target0 = app_data.loc[app_data.TARGET == 0]
app_target1 = app_data.loc[app_data.TARGET == 1]

app_target0.shape

app_target1.shape

cols = ['Age Group','NAME_CONTRACT_TYPE', 'NAME_INCOME_TYPE','NAME_EDUCATION_TYPE']

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().index,
                         y=app_target0[cols[count]].value_counts(),
                         name=cols[count],
                         textposition='auto',
                         text= [str(i) + '%' for i in
(app_target0[cols[count]].value_counts(normalize=True)*100).round(1).tolist()],
                        ),
                  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]].value_counts(normalize=True)*100).round(1).tolist()],
                        ),
                  row=i,col=j)
            count+=1
fig.update_layout(
                title=dict(text = "Analyze Categorical variables (Payment/ Non-Payment Difficulties)",x=0.5,y=0.99),
                title_font_size=20,
                showlegend=False,
                height = 1600,
               )
fig.show()

cols = ['NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE','CODE_GENDER','Work Experience']

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().index,
                         y=app_target0[cols[count]].value_counts(),
                         name=cols[count],
```

```
                    textposition='auto',
                    text= [str(i) + '%' for i in
(app_target0[cols[count]].value_counts(normalize=True)*100).round(1).tolist()],
                   ),
               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]].value_counts(normalize=True)*100).round(1).tolist()],
                   ),
               row=i,col=j)
        count+=1
fig.update_layout(
              title=dict(text = "Analyze Categorical variables (Payment/ Non-Payment Difficulties)",x=0.5,y=0.99),
              title_font_size=20,
              showlegend=False,
              height = 1600,
             )
fig.show()

cols = ['OCCUPATION_TYPE', 'ORGANIZATION_TYPE' ,'AMT_INCOME_TOTAL_Range','AMT_CREDIT_Range']

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().index,
                    y=app_target0[cols[count]].value_counts(),
                    name=cols[count],
                    textposition='auto',
                    text= [str(i) + '%' for i in
(app_target0[cols[count]].value_counts(normalize=True)*100).round(1).tolist()],
                   ),
               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]].value_counts(normalize=True)*100).round(1).tolist()],
                       ),
                  row=i,col=j)
          count+=1
fig.update_layout(
                title=dict(text = "Analyze Categorical variables (Payment/ Non-Payment Difficulties)",x=0.5,y=0.99),
                title_font_size=20,
                showlegend=False,
                height = 1600,
               )
fig.show()

# ### Bivariate  / Multivariate Analysis
# Group data by 'AMT_CREDIT_Range' & 'CODE_GENDER'
df1=app_data.groupby(by=['AMT_CREDIT_Range','CODE_GENDER']).count().reset_index()[['AMT_CREDIT_Range','CODE
_GENDER','SK_ID_CURR']]
df1

# Group data by 'AMT_INCOME_TOTAL_Range' & 'CODE_GENDER'
df2=app_data.groupby(by=['AMT_INCOME_TOTAL_Range','CODE_GENDER']).count().reset_index()[['AMT_INCOME_TOT
AL_Range','CODE_GENDER','SK_ID_CURR']]
df2

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 Range",x=0.5,y=0.95),
                title_font_size=20,
               )
fig1.show()

# __Insights__
#
# - Females are mostly applying for __Very Low__ credit loans.
# - Males are applying for __Medium__ & __High__ credit loans.

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 Income Range",x=0.5,y=0.95),
                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)__

fig = px.box(app_target0, x="NAME_EDUCATION_TYPE", y="AMT_CREDIT", color='NAME_FAMILY_STATUS',
           title="Education Type VS Credit Amount (Non Payment Difficulties)")
fig.show()

fig = px.box(app_target1, x="NAME_EDUCATION_TYPE", y="AMT_CREDIT", color='NAME_FAMILY_STATUS',
           title="Education Type VS Credit Amount (Payment Difficulties)")
fig.show()

# __Income VS Credit Amount (Payment / Non Payment Difficulties)__

fig = px.box(app_target0, x="AMT_INCOME_TOTAL_Range", y="AMT_CREDIT", color='NAME_FAMILY_STATUS',
           title="Income Range VS Credit Amount (Non-Payment Difficulties)")
fig.show()

fig = px.box(app_target1, x="AMT_INCOME_TOTAL_Range", y="AMT_CREDIT", color='NAME_FAMILY_STATUS',
           title="Income Range VS Credit Amount (Payment Difficulties)")
fig.show()

# __Age Group VS Credit Amount (Payment / Non Payment Difficulties)__

fig = px.box(app_target0, x="Age Group", y="AMT_CREDIT", color='NAME_FAMILY_STATUS',
           title="Age Group VS Credit Amount (Non-Payment Difficulties)")
fig.show()

fig = px.box(app_target1, x="Age Group", y="AMT_CREDIT", color='NAME_FAMILY_STATUS',
           title="Age Group VS Credit Amount (Payment Difficulties)")
fig.show()

# __Work Experience VS Credit Amount (Payment / Non Payment Difficulties)__

fig = px.box(app_target0, x="Work Experience", y="AMT_CREDIT", color='NAME_FAMILY_STATUS',
           title="Work Experience VS Credit Amount (Non-Payment Difficulties)")
fig.show()

fig = px.box(app_target1, x="Work Experience", y="AMT_CREDIT", color='NAME_FAMILY_STATUS',
           title="Work Experience VS Credit Amount (Payment Difficulties)")
fig.show()

# __Numerical vs Numerical Variables__
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__
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_RELATIVE']].corr(), annot=True, cmap="RdYlGn")
plt.title('Correlation matrix for Non-Payment Difficulties')
plt.show()

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_RELATIVE']].corr(), annot=True, cmap='RdYlGn')
plt.title('Correlation Matrix for Payment Difficulties')
plt.show()

# ### Data Analysis on Previous Application dataset
appdata_previous = pd.read_csv("../input/credit-eda-case-study/previous_application.csv");
appdata_previous.head()

# __Drop Columns with NULL Values greater than 40%__
s1= (appdata_previous.isnull().mean()*100).sort_values(ascending=False)[appdata_previous.isnull().mean()*100 > 40]
s1

appdata_previous.shape

appdata_previous.drop(columns = s1.index,inplace=True)

appdata_previous.shape

# __Changing negative values in the DAYS columns to positive values__

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[i].unique()))
        print()

appdata_previous[days]= abs(appdata_previous[days])

appdata_previous[days]

# 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__

appdata_previous.columns

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().index,
                       y=appdata_previous[cols[count]].value_counts(),
                       name=cols[count],
                       textposition='auto',
                       text= [str(i) + '%' for i in
(appdata_previous[cols[count]].value_counts(normalize=True)*100).round(1).tolist()],
                      ),
                 row=i,col=j)
      count+=1
fig.update_layout(
                title=dict(text = "Analyze Categorical variables (Frequency / Percentage)",x=0.5,y=0.99),
                title_font_size=20,
                showlegend=False,
                width = 960,
                height = 1200,
              )
fig.show()

# #### Approved Loans
approved=appdata_previous[appdata_previous['NAME_CONTRACT_STATUS']=='Approved']

cols = ['NAME_PORTFOLIO','NAME_GOODS_CATEGORY',
      'CHANNEL_TYPE','NAME_YIELD_GROUP' , 'NAME_PRODUCT_TYPE','NAME_CASH_LOAN_PURPOSE']

#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=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]].value_counts(normalize=True)*100).round(1).tolist()],
                            ),
                      row=i,col=j)
        count+=1
fig.update_layout(
                    title=dict(text = "Analyze Categorical variables (Frequency / Percentage)",x=0.5,y=0.99),
                    title_font_size=20,
                    showlegend=False,
                    width = 960,
                    height = 1400,
                  )
fig.show()

# #### Refused Loans

refused=appdata_previous[appdata_previous['NAME_CONTRACT_STATUS']=='Refused']

cols = ['NAME_PORTFOLIO','NAME_GOODS_CATEGORY',
        'CHANNEL_TYPE','NAME_YIELD_GROUP' , 'NAME_PRODUCT_TYPE','NAME_CASH_LOAN_PURPOSE']

#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[count]].value_counts(normalize=True)*100).round(1).tolist()],
                            ),
                      row=i,col=j)
        count+=1
fig.update_layout(
                    title=dict(text = "Analyze Categorical variables (Frequency / Percentage)",x=0.5,y=0.99),
                    title_font_size=20,
                    showlegend=False,
                    width = 960,
                    height = 1400,
```

```
            )
fig.show()

# ## Merging Application & Previous Application Data

appdata_merge = app_data.merge(appdata_previous,on='SK_ID_CURR', how='inner')
appdata_merge.shape

# __Analysis of Merged Data__

# 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_CONTRACT_STATUS']==col_value[count]][column_nam
e].value_counts().index,

y=appdata_merge[appdata_merge['NAME_CONTRACT_STATUS']==col_value[count]][column_name].value_counts(),
                    name=cols[count],
                    textposition='auto',
                    text= [str(i) + '%' for i in
(appdata_merge[appdata_merge['NAME_CONTRACT_STATUS']==col_value[count]][column_name].value_counts(norma
lize=True)*100).round(1).tolist()],
                   ),
                row=i,col=j)
            count+=1
    fig.update_layout(
             title=dict(text = "NAME_CONTRACT_STATUS VS "+column_name,x=0.5,y=0.99),
             title_font_size=20,
             showlegend=False,
             width = 960,
             height = 960,
            )
    fig.show()

# 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": "pie"}, {"type": "pie"}]],
                 )
    # 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_CONTRACT_STATUS']==col_value[count]][column_
name].value_counts().index,

values=appdata_merge[appdata_merge['NAME_CONTRACT_STATUS']==col_value[count]][column_name].value_counts(
),
                          textinfo='percent',
                          insidetextorientation='auto',
                          hole=.3
                         ),
                  row=i,col=j)
          count+=1
    fig.update_layout(
                 title=dict(text = "NAME_CONTRACT_STATUS VS "+column_name,x=0.5,y=0.99),
                 title_font_size=20,
                 width = 960,
                 height = 960,
               )
    fig.show()

plot_pie_merge(appdata_merge,'NAME_CONTRACT_TYPE_y')

# __Insights__
#
# - Banks mostly approve Consumer Loans
# - Most of the __Refused_ & __Cancelled__ loans are __cash loans__.

plot_pie_merge(appdata_merge,'NAME_CLIENT_TYPE')

# __Insights__
#
# - Most of the approved , refused & canceled loans belong to the old clients.
# - Almost __27.4%__ loans were provided to new customers.

plot_pie_merge(appdata_merge,'CODE_GENDER')

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

plot_merge(appdata_merge,'NAME_EDUCATION_TYPE')

# __Insights__
```

```
#
# - Most of the approved loans belong to applicants with __Secondary / Secondary Special__ education type.

plot_merge(appdata_merge,'NAME_INCOME_TYPE')

# __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.

plot_pie_merge(appdata_merge,'NAME_FAMILY_STATUS')

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

plot_pie_merge(appdata_merge,'NAME_PORTFOLIO')

# __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

plot_merge(appdata_merge,'OCCUPATION_TYPE')

plot_merge(appdata_merge,'NAME_GOODS_CATEGORY')

plot_merge(appdata_merge,'PRODUCT_COMBINATION')

# __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.

plot_merge(appdata_merge,'NAME_PAYMENT_TYPE')

plot_merge(appdata_merge,'CHANNEL_TYPE')

# __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.
#

plot_pie_merge(appdata_merge,'NAME_YIELD_GROUP')

# __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.

plot_pie_merge(appdata_merge,'NAME_HOUSING_TYPE')

plot_merge(appdata_merge,'Age Group')

plot_merge(appdata_merge,'Work Experience')

plot_merge(appdata_merge,'AMT_CREDIT_Range')

# __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.

plot_merge(appdata_merge,'AMT_INCOME_TOTAL_Range')

# __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
```