```
from google.colab import drive
from scipy.stats import zscore, boxcox
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Define the path to the folder you want to mount
drive_path = "/content/drive"
data_path = os.path.join(drive_path, "My Drive/dsa_project")

drive.mount(drive_path)
os.listdir(data_path)

TGT = "TARGET"
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

**Question 1:** SK_ID_CURR Analysis:Get the count of unique values of SK_ID_CURR in file application_train.csv and compare this count to the number of rows in application_train.csv. Compare this with the total row count. Investigate if SK_ID_CURR serves as the table's primary key.?

**Analysis:**

Since the number of rows is same as number of unique values of SK_ID_CURR i.e. 307511, we can confirm that SK_ID_CURR is the primary id.

```
app_train = pd.read_csv(os.path.join(data_path, "application_train.csv"))
print(f"app_train.shape: {app_train.shape[0]} | len(app_train['SK_ID_CURR'].unique()): {len(app_train['SK_ID_CURR'].unique())}")
```

```
app_train.shape: 307511 | len(app_train['SK_ID_CURR'].unique()): 307511
```

**Question 2:** TARGET Column Analysis: Identify and quantify the unique values within the TARGET column. Assess the dataset's balance by evaluating the proportions of each target value.

**Analysis:**

There is a clear imbalance in the Target Variables as evident by the percentage of each unique value.

```
print(app_train[TGT].value_counts())
print(app_train[TGT].value_counts(normalize=True))
```

```
0    282686
1     24825
Name: TARGET, dtype: int64
0    0.919271
1    0.080729
Name: TARGET, dtype: float64
```

```
all_num_cols = app_train.select_dtypes(include=['number']).columns.tolist()
```

There is an imbalance in the TARGET values with 0 being 91% of the data.

```
null_df = list()

for c in all_num_cols:
  _miss_count = len(app_train[app_train[c].isnull()])
  _miss_pct = round(_miss_count/len(app_train), 2) * 100

  null_df.append({"col": c, "miss": _miss_count, "miss_pct": _miss_pct})

null_df = pd.DataFrame(null_df).sort_values("miss").reset_index(drop=True)
null_df = null_df[null_df["col"].isin([c for c in null_df["col"].tolist() if "FLAG" not in c.upper()])]
null_df.head(25)
```

| | col | miss | miss_pct |
|---|---|---|---|
| **0** | SK_ID_CURR | 0 | 0.0 |
| **1** | REG_CITY_NOT_WORK_CITY | 0 | 0.0 |
| **22** | REG_CITY_NOT_LIVE_CITY | 0 | 0.0 |
| **23** | LIVE_REGION_NOT_WORK_REGION | 0 | 0.0 |
| **24** | LIVE_CITY_NOT_WORK_CITY | 0 | 0.0 |
| **25** | REG_REGION_NOT_LIVE_REGION | 0 | 0.0 |
| **26** | AMT_CREDIT | 0 | 0.0 |
| **27** | REGION_POPULATION_RELATIVE | 0 | 0.0 |
| **28** | DAYS_BIRTH | 0 | 0.0 |
| **29** | DAYS_EMPLOYED | 0 | 0.0 |
| **30** | DAYS_REGISTRATION | 0 | 0.0 |
| **31** | DAYS_ID_PUBLISH | 0 | 0.0 |
| **32** | REG_REGION_NOT_WORK_REGION | 0 | 0.0 |
| **35** | AMT_INCOME_TOTAL | 0 | 0.0 |
| **39** | REGION_RATING_CLIENT | 0 | 0.0 |
| **40** | TARGET | 0 | 0.0 |
| **41** | REGION_RATING_CLIENT_W_CITY | 0 | 0.0 |
| **42** | HOUR_APPR_PROCESS_START | 0 | 0.0 |
| **44** | CNT_CHILDREN | 0 | 0.0 |
| **45** | DAYS_LAST_PHONE_CHANGE | 1 | 0.0 |
| **46** | CNT_FAM_MEMBERS | 2 | 0.0 |
| **47** | AMT_ANNUITY | 12 | 0.0 |
| **48** | AMT_GOODS_PRICE | 278 | 0.0 |
| **49** | EXT_SOURCE_2 | 660 | 0.0 |
| **50** | DEF_60_CNT_SOCIAL_CIRCLE | 1021 | 0.0 |

```
selected_num_col = list()
selected_num_col.append('TARGET')
selected_num_col.append("CNT_CHILDREN")
selected_num_col.append("AMT_INCOME_TOTAL")
selected_num_col.append("AMT_CREDIT")
selected_num_col.append("AMT_ANNUITY")
selected_num_col.append("REGION_POPULATION_RELATIVE")
selected_num_col.append("DAYS_BIRTH")
selected_num_col.append("DAYS_EMPLOYED")
selected_num_col.append("DAYS_REGISTRATION")
selected_num_col.append("DAYS_ID_PUBLISH")
selected_num_col.append("HOUR_APPR_PROCESS_START")

# Not sure if these columns should be considered numerical or categorical
# selected_num_col.append("REGION_RATING_CLIENT")
# selected_num_col.append("REG_REGION_NOT_WORK_REGION")

# Too many missing values
# selected_num_col.append('YEARS_BUILD_AVG')
# selected_num_col.append('DAYS_LAST_PHONE_CHANGE')
# selected_num_col.append('AMT_REQ_CREDIT_BUREAU_YEAR')
# selected_num_col.append('OWN_CAR_AGE')
# selected_num_col.append('AMT_GOODS_PRICE')

null_df[null_df["col"].isin(set(selected_num_col))]
```

| | col | miss | miss_pct |
|---|---|---|---|
| **26** | AMT_CREDIT | 0 | 0.0 |
| **27** | REGION_POPULATION_RELATIVE | 0 | 0.0 |
| **28** | DAYS_BIRTH | 0 | 0.0 |
| **29** | DAYS_EMPLOYED | 0 | 0.0 |
| **30** | DAYS_REGISTRATION | 0 | 0.0 |
| **31** | DAYS_ID_PUBLISH | 0 | 0.0 |
| **35** | AMT_INCOME_TOTAL | 0 | 0.0 |
| **40** | TARGET | 0 | 0.0 |
| **42** | HOUR_APPR_PROCESS_START | 0 | 0.0 |
| **44** | CNT_CHILDREN | 0 | 0.0 |
| **47** | AMT_ANNUITY | 12 | 0.0 |

```
app_train[selected_num_col].dtypes
```

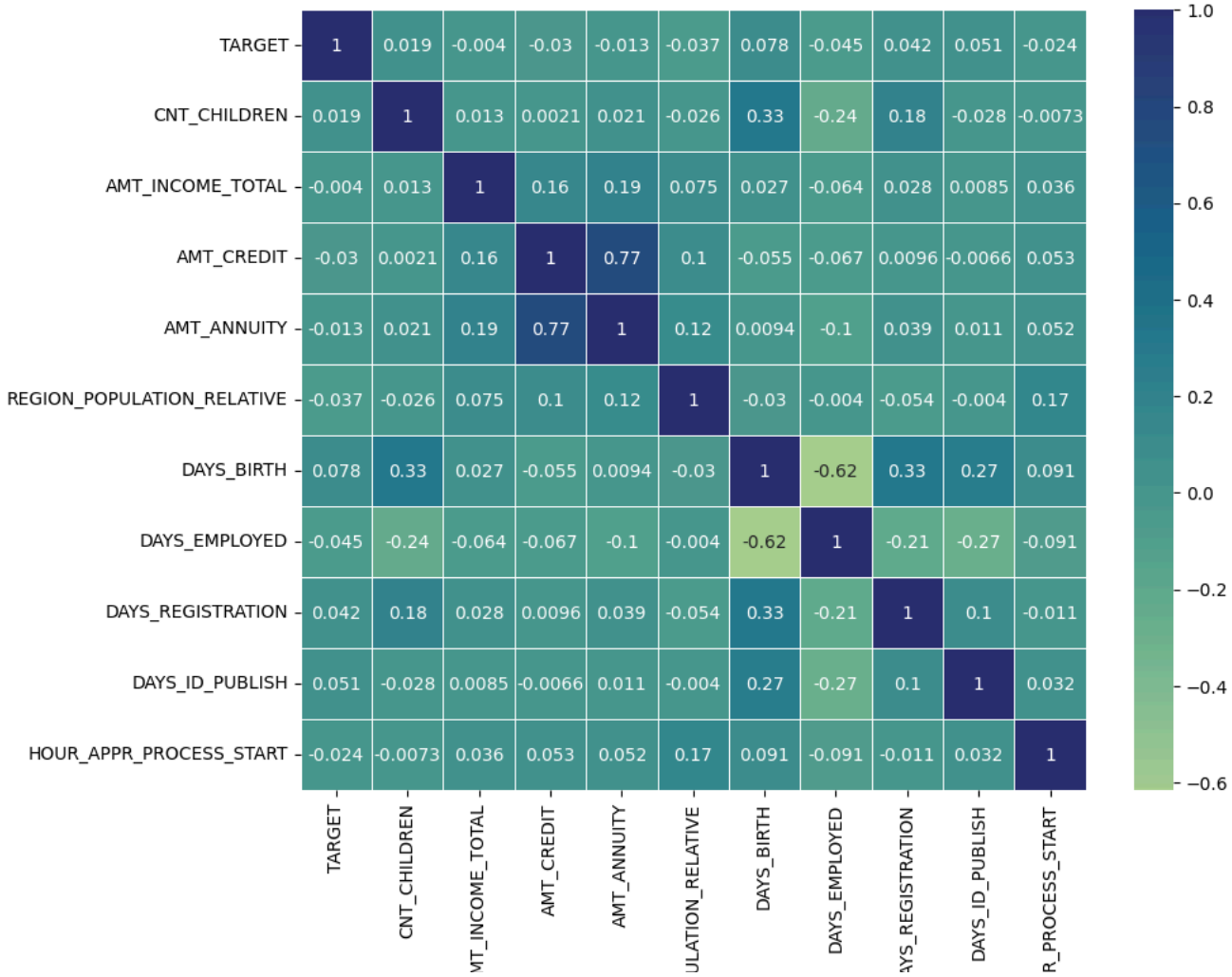
```
TARGET                          int64
CNT_CHILDREN                    int64
AMT_INCOME_TOTAL              float64
AMT_CREDIT                    float64
AMT_ANNUITY                   float64
REGION_POPULATION_RELATIVE    float64
DAYS_BIRTH                      int64
DAYS_EMPLOYED                   int64
DAYS_REGISTRATION             float64
DAYS_ID_PUBLISH                 int64
HOUR_APPR_PROCESS_START         int64
dtype: object
```

**Question 3:** Correlation Analysis: Generate a Pearson correlation matrix and heatmap (for any 10 numeric variables of choice) on application_tain.csv. Write code to list the top 5 features correlated with the TARGET column.

```
corr = app_train[selected_num_col].corr()
```

```
plt.figure(figsize = (10,8))
sns.heatmap(corr, annot = True, cmap = 'crest', linewidth = 0.5)
```

<Axes: >

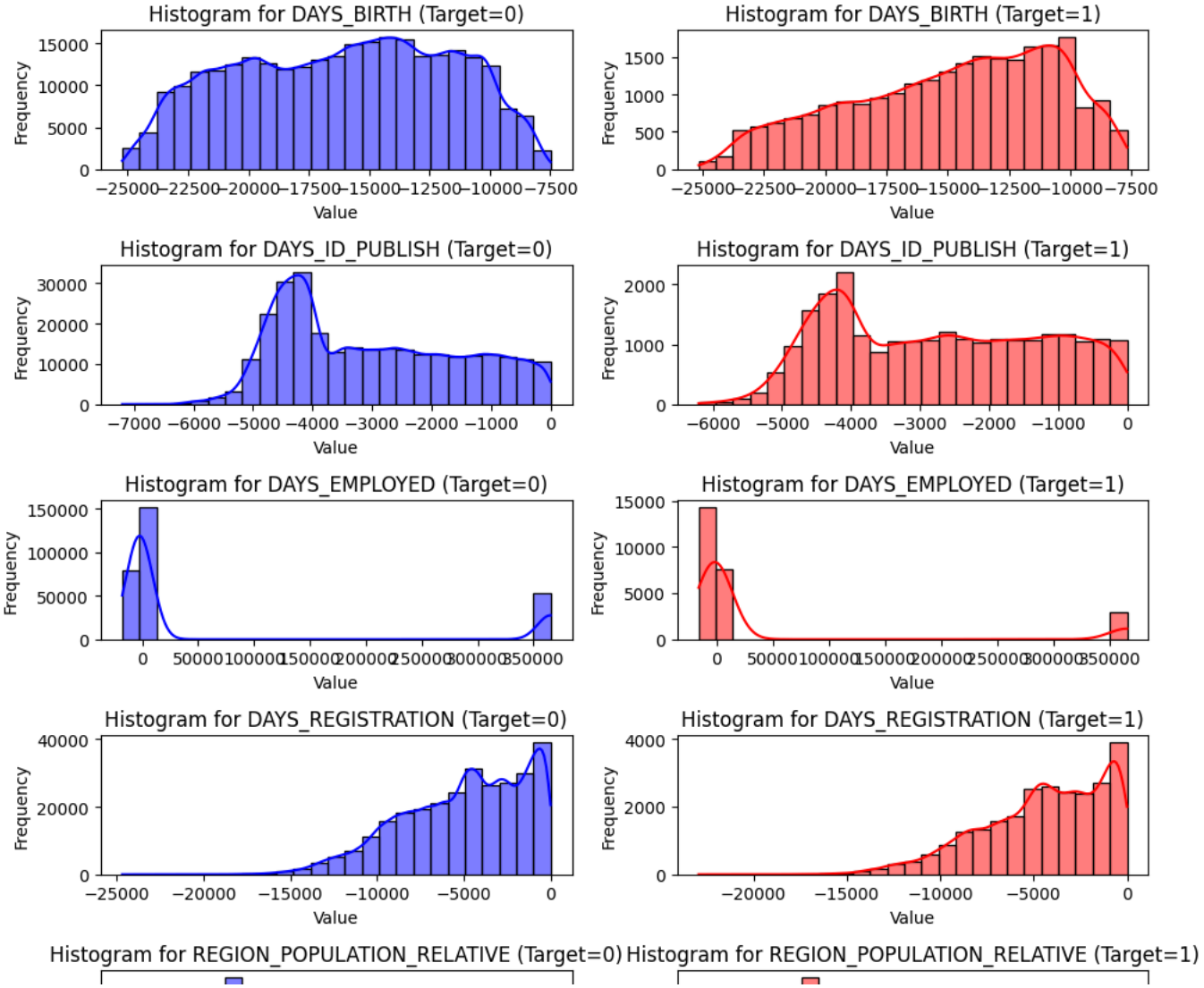



```
corr_vars = corr[TGT].sort_values(ascending = False)
corr_vars = corr_vars.abs()
corr_vars = corr_vars.reset_index().rename(columns={"index": "vars"})
corr_vars = corr_vars[(corr_vars["vars"] != TGT)]

top5_corr_vars = corr_vars.sort_values(TGT, ascending=False).head(5)["vars"].tolist()
corr_vars.sort_values(TGT, ascending=False).head(5)[["vars", "TARGET"]]
```

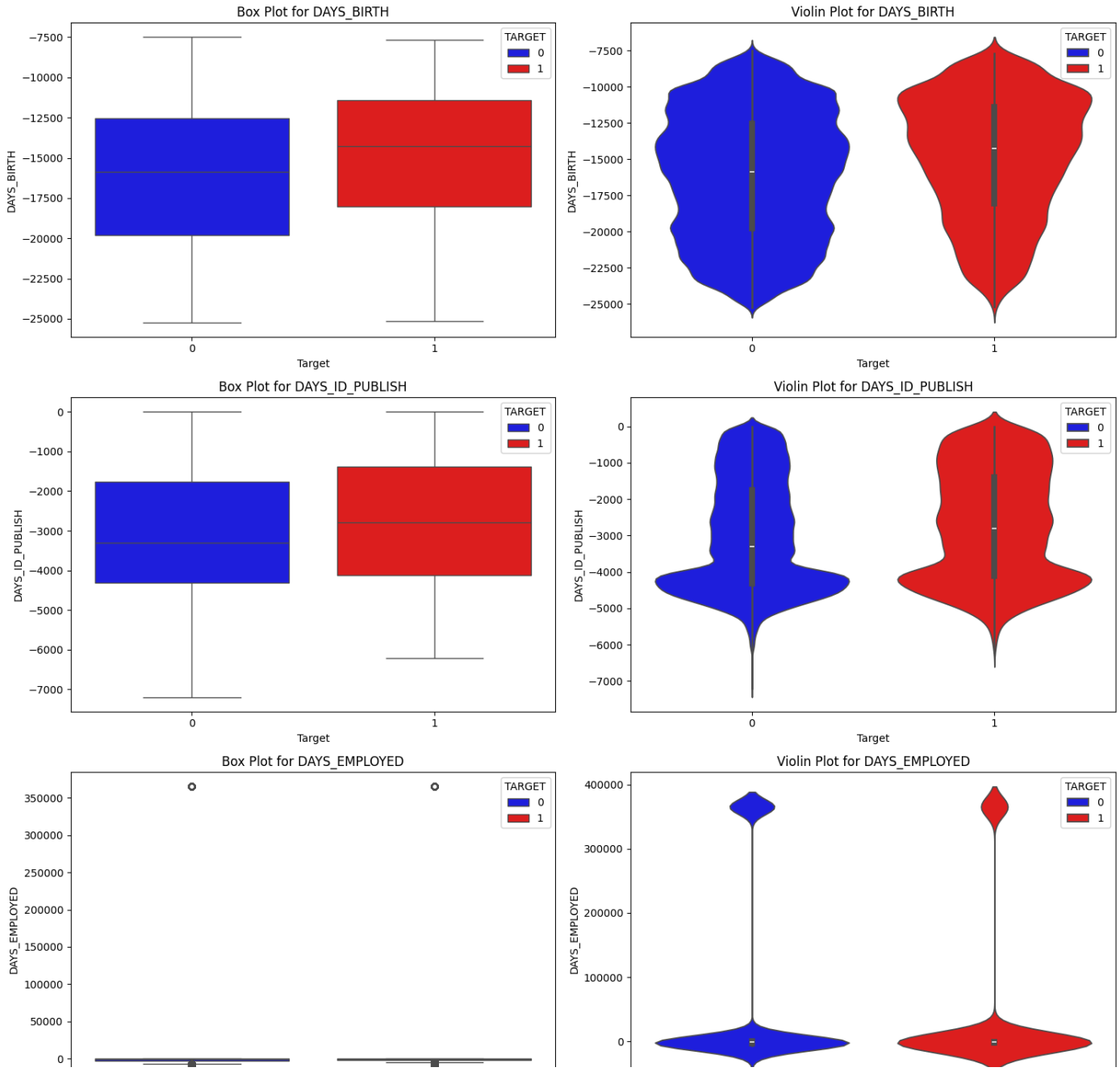
| | vars | TARGET |
|---|---|---|
| 1 | DAYS_BIRTH | 0.078239 |
| 2 | DAYS_ID_PUBLISH | 0.051457 |
| 10 | DAYS_EMPLOYED | 0.044932 |
| 3 | DAYS_REGISTRATION | 0.041975 |
| 9 | REGION_POPULATION_RELATIVE | 0.037227 |

**Question 4:** Histogram: Generate histograms for any five numerical features in application_train.csv, and comment on whether they seem Gaussian, or have severe skews. Visualize the relationship between each of these numeric variables and the target variable.

```
plt.figure(figsize = (10,10))

app_train0 = app_train[(app_train[TGT] == 0)]
app_train1 = app_train[(app_train[TGT] == 1)]

for i, _v in enumerate(top5_corr_vars):

    plt.subplot(len(top5_corr_vars), 2, 2*i+1)
    sns.histplot(x=app_train0[_v], bins=25, kde=True, color="blue")
    plt.title(f'Histogram for {_v} (Target=0)')
    plt.xlabel('Value')
    plt.ylabel('Frequency')

    plt.subplot(len(top5_corr_vars), 2, 2*i+2)
    sns.histplot(x=app_train1[_v], bins=25, kde=True, color="red")
    plt.title(f'Histogram for {_v} (Target=1)')
    plt.xlabel('Value')
    plt.ylabel('Frequency')

# Adjust layout to prevent overlapping titles
plt.tight_layout()

# Show the plots
plt.show()
```

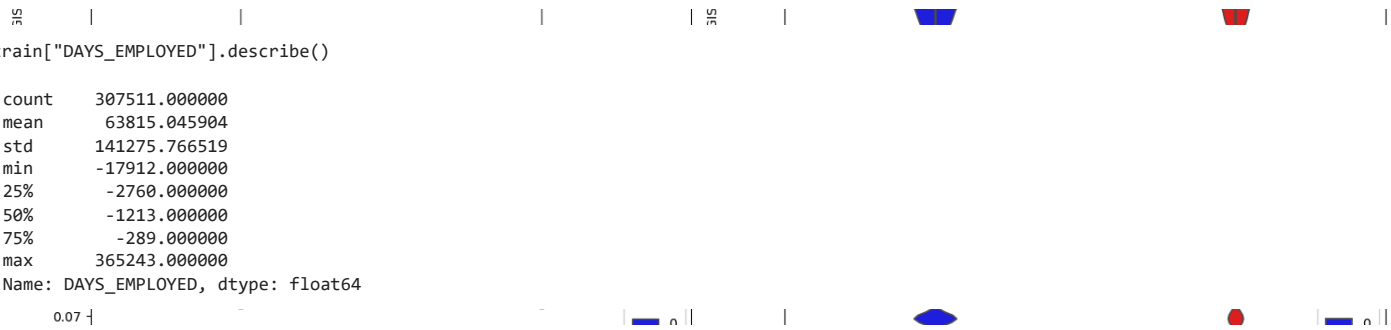


```
plt.figure(figsize=(15, 5 * len(top5_corr_vars)))

for i, var in enumerate(top5_corr_vars):
    # Box plot
    plt.subplot(len(top5_corr_vars), 2, 2 * i + 1)
    sns.boxplot(x=app_train[TGT], y=app_train[var], hue=app_train[TGT], palette={0: 'blue', 1: 'red'})
    plt.title(f'Box Plot for {var}')
    plt.xlabel('Target')
    plt.ylabel(var)

    # Violin plot
    plt.subplot(len(top5_corr_vars), 2, 2 * i + 2)
    sns.violinplot(x=app_train[TGT], y=app_train[var], hue=app_train[TGT], palette={0: 'blue', 1: 'red'})
    plt.title(f'Violin Plot for {var}')
    plt.xlabel('Target')
    plt.ylabel(var)

# Adjust layout to prevent overlapping titles
plt.tight_layout()

# Show the plots
plt.show()
```



```
for c in top5_corr_vars:
  _miss_count = len(app_train[app_train[c].isnull()])
  print(f"Null values in {c}: {_miss_count}, {round(_miss_count/len(app_train), 2) * 100} %")
```

```
Null values in DAYS_BIRTH: 0, 0.0 %
Null values in DAYS_ID_PUBLISH: 0, 0.0 %
Null values in DAYS_EMPLOYED: 0, 0.0 %
Null values in DAYS_REGISTRATION: 0, 0.0 %
Null values in REGION_POPULATION_RELATIVE: 0, 0.0 %
```



```
app_train["DAYS_EMPLOYED"].describe()
```

```
count    307511.000000
mean      63815.045904
std      141275.766519
min      -17912.000000
25%       -2760.000000
50%       -1213.000000
75%        -289.000000
max      365243.000000
Name: DAYS_EMPLOYED, dtype: float64
```



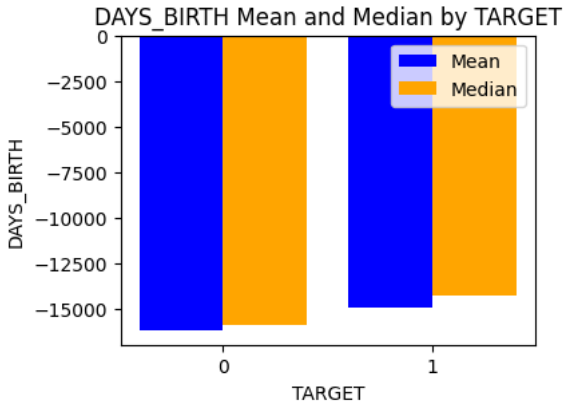
**Question 5:** Outlier Analysis: Perform outlier analysis on the chosen variables.

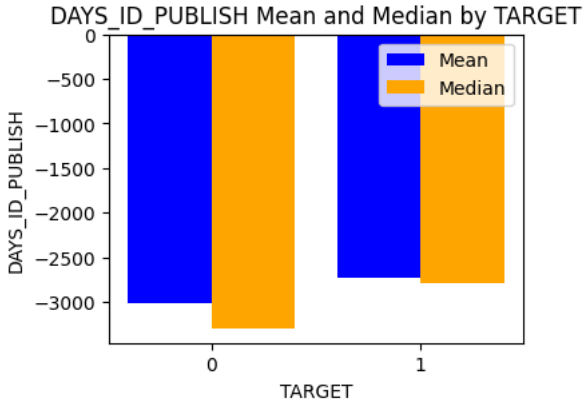
```
# Record count of rows where Z score of target column in higher than 2 and 3 separately
suff = "z_score"
z_score_cols = list()
outlier_df = list()

for c in top5_corr_vars:
  app_train[f"{c}_{suff}"] = zscore(app_train[c])
  over_2 = len(app_train[(app_train[f"{c}_{suff}"].abs() > 2)])
  over_3 = len(app_train[(app_train[f"{c}_{suff}"].abs() > 3)])
  outlier_df.append({"col": c, "over_2": over_2, "over_3": over_3})

outlier_df = pd.DataFrame(outlier_df)
outlier_df
```

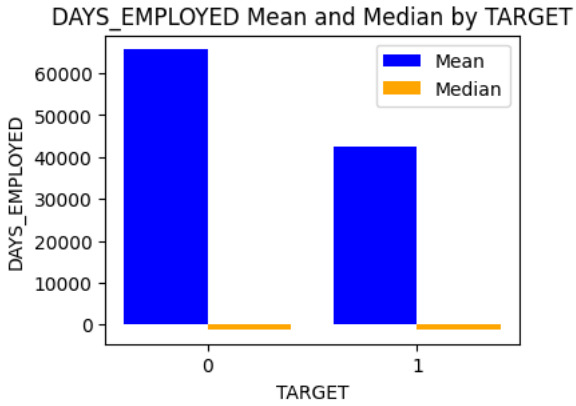
| | col | over_2 | over_3 |
|---|---|---|---|
| **0** | DAYS_BIRTH | 1210 | 0 |
| **1** | DAYS_ID_PUBLISH | 390 | 0 |
| **2** | DAYS_EMPLOYED | 55374 | 0 |
| **3** | DAYS_REGISTRATION | 11330 | 749 |
| **4** | REGION_POPULATION_RELATIVE | 8412 | 8412 |

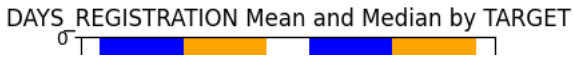
**Question 6:** Transformation of Nuemric Variables: If skewed, perform suitable transformations on these five numerical variables. Check the relationship of each of these numeric variables with the target variable using bar charts. Visualize the relationship between each of these numeric variables and the target variable. Perform outlier analysis on the transformed variables and report any differences before and after transformation.

```
# Calculate the selected metrics for each numeric variable grouped by the target variable
metrics = ['mean', 'median']
central_tend = {}
for metric in metrics:
    if metric == 'mean':
        t = app_train.groupby(TGT)[top5_corr_vars].mean()
    elif metric == 'median':
        t = app_train.groupby(TGT)[top5_corr_vars].median()
    elif metric == 'std':
        t = app_train.groupby(TGT)[top5_corr_vars].std()
    elif metric == 'count':
        t = app_train.groupby(TGT)[top5_corr_vars].count()
    central_tend[metric] = t

# Merge the DataFrames on 'TARGET' column
merged_df = pd.merge(central_tend["mean"],
                     central_tend["median"],\
                     left_index=True, right_index=True,
                     suffixes=('_mean', '_median'))

merged_df = merged_df.reset_index()

# Columns to plot
columns_to_plot = list(merged_df.columns)[1:]

# Plot bar graphs
for column in top5_corr_vars:
    plt.figure(figsize=(4, 3))
    plt.bar(merged_df['TARGET'], merged_df[column + '_mean'], label='Mean', width=0.4, color='blue')
    plt.bar(merged_df['TARGET'] + 0.4, merged_df[column + '_median'], label='Median', width=0.4, color='orange')
    plt.xlabel('TARGET')
    plt.ylabel(column)
    plt.title(f'{column} Mean and Median by TARGET')
    plt.xticks(merged_df['TARGET'] + 0.2, merged_df['TARGET'])
    plt.legend()
    plt.show()
```

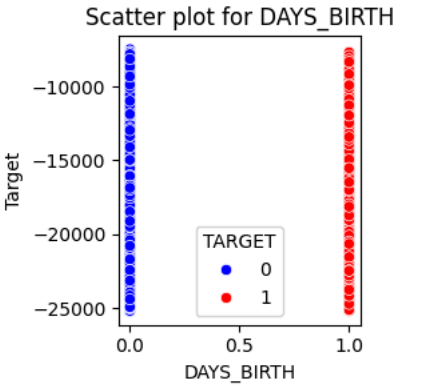


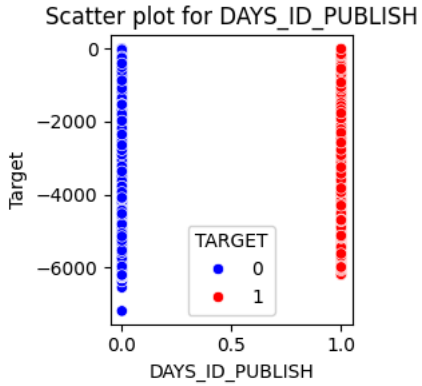



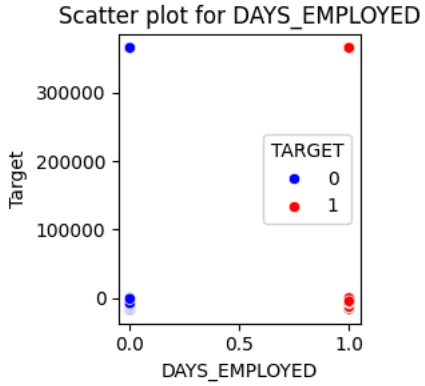



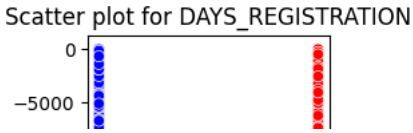




```
plt.figure(figsize=(3, 15))

for i, var in enumerate(top5_corr_vars):
    plt.subplot(len(top5_corr_vars), 1, i + 1)
    sns.scatterplot(x=app_train[TGT], y=app_train[var], hue=app_train[TGT], palette={0: 'blue', 1: 'red'})
    plt.title(f'Scatter plot for {var}')
    plt.xlabel(var)
    plt.ylabel('Target')

# Adjust layout to prevent overlapping titles
plt.tight_layout()

# Show the plots
plt.show()
```


Scatter plot for DAYS_BIRTH
Target
−10000
−15000
−20000
−25000
TARGET
0
1
0.0
0.5
1.0
DAYS_BIRTH


Scatter plot for DAYS_ID_PUBLISH
Target
0
−2000
−4000
−6000
TARGET
0
1
0.0
0.5
1.0
DAYS_ID_PUBLISH

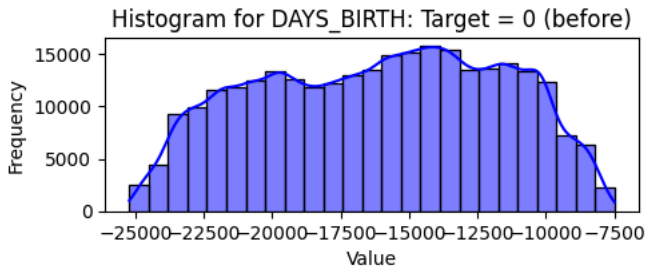

Scatter plot for DAYS_EMPLOYED
Target
300000
200000
100000
0
TARGET
0
1
0.0
0.5
1.0
DAYS_EMPLOYED

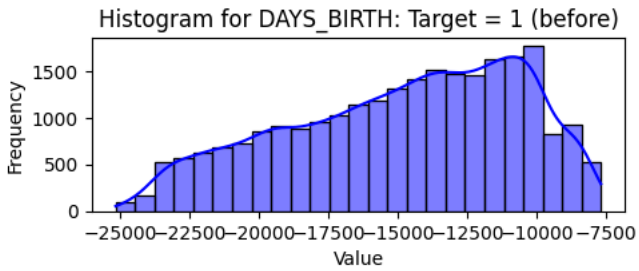

Scatter plot for DAYS_REGISTRATION
0
−5000

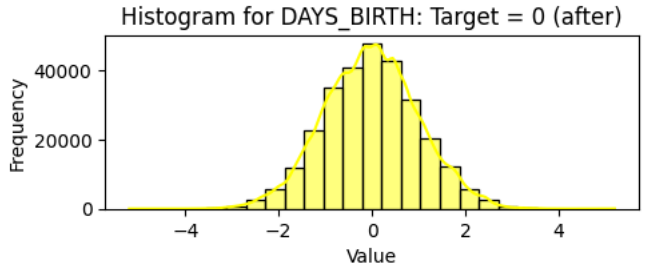
```
from sklearn.preprocessing import QuantileTransformer

# Transformation for DAYS_REGISTRATION
# Increasing - Positively skewed
t_map = dict()
t_map["DAYS_ID_PUBLISH"] = "quantile"
t_map["DAYS_REGISTRATION"] = "quantile"
t_map["DAYS_BIRTH"] = "quantile"
t_map["DAYS_EMPLOYED"] = "quantile"
t_map["REGION_POPULATION_RELATIVE"] = "quantile"

offset_map = dict()
offset_map["DAYS_REGISTRATION"] = 50000
offset_map["DAYS_ID_PUBLISH"] = "min"
offset_map["DAYS_BIRTH"] = "min"
offset_map["DAYS_EMPLOYED"] = "min"

def transform_var(df, var, t_type):
  if t_type == "log":
    df[var] = df[var].apply(lambda x: np.log(x + 1))
  elif t_type == "sqrt":
    df[var] = df[var].apply(lambda x: np.sqrt(x + 1))
  elif t_type == "cube_root":
    df[var] = df[var].apply(lambda x: x**(1/3))
  elif t_type == "exp":
    df[var] = df[var].apply(lambda x: np.exp(x))
  elif t_type == "inv":
    df[var] = df[var].apply(lambda x: 1/x)
  elif t_type == "logit":
    df[var] = df[var].apply(lambda x: np.log(x/(1+x)))
  elif t_type == "boxcox":
    df[var], _ = boxcox(df[var])
  elif t_type == "quantile":
    df[var] = QuantileTransformer(output_distribution='normal').fit_transform(np.array(df[var]).reshape(-1, 1))
  return df

def plot_histogram(df, col, col_i, all_cols, sp_i, sp_j, tgt_val, suffix, color="blue"):
  plt.subplot(len(all_cols), sp_i, sp_j)
  sns.histplot(x=df[col], bins=25, kde=True, color=color)
  plt.title(f'Histogram for {col}: Target = {tgt_val} ({suffix})')
  plt.xlabel('Value')
  plt.ylabel('Frequency')


def transform_var_and_plot(var, df, all_vars):

    i = 1
    plt.figure(figsize = (10,10))

    print(f"Plotting: {var}")

    if var in t_map:

      t_type = t_map[var]

      # ================================
      # Plot before transformation
      # ================================
      plot_histogram(df=df[(df[TGT] == 0)], col_i=i, col=var, suffix="before",
                     sp_i=2, sp_j=2*i+1, all_cols=all_vars, tgt_val="0")
      plot_histogram(df=app_train[(app_train[TGT] == 1)], col_i=i, col=var,
                     sp_i=2, sp_j=2*i+2, all_cols=all_vars, tgt_val="1", suffix="before")

      # ================================
      # Transform data
      # ================================
      if var in offset_map:
        if str(offset_map[var]) == "min":
          df[var] = df[var] + abs(df[var].min())
        else:
          df[var] = df[var] + offset_map[var]

      df = transform_var(df=df, var=var, t_type=t_type)

      # Increment the value of i
      i += 1
```

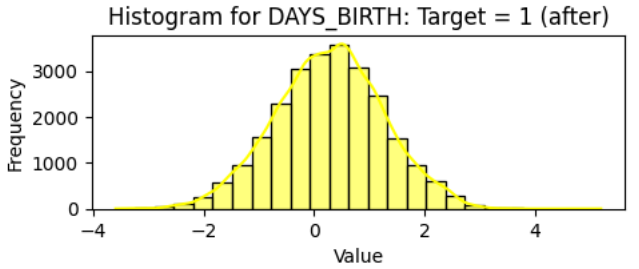
```
    else:
      print(f"No Transformation required")

    # ================================
    # Plot after transformation
    # ================================
    plot_histogram(df=df[(df[TGT] == 0)], col_i=i, col=var,
                   sp_i=2, sp_j=2*i+1, all_cols=all_vars, tgt_val="0",
                   color="yellow", suffix="after")
    plot_histogram(df=df[(df[TGT] == 1)], col_i=i, col=var,
                   sp_i=2, sp_j=2*i+2, all_cols=all_vars, tgt_val="1",
                   color="yellow", suffix="after")

    # Increment the value of i
    i += 1

    # Adjust layout to prevent overlapping titles
    plt.tight_layout()

    # Show the plots
    plt.show()

for var in top5_corr_vars:
  transform_var_and_plot(var=var, all_vars=top5_corr_vars, df=app_train)
```

```
Plotting: DAYS_BIRTH
```

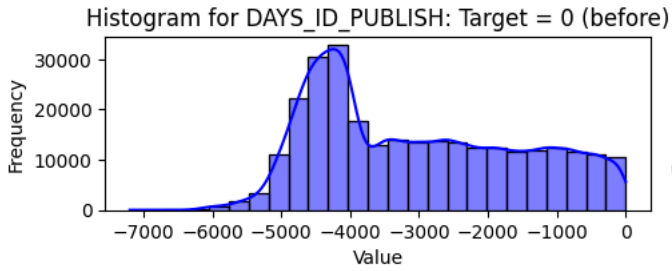


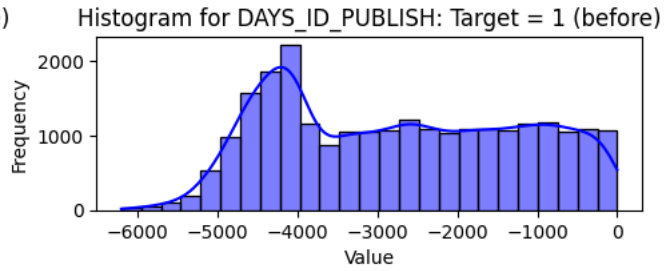



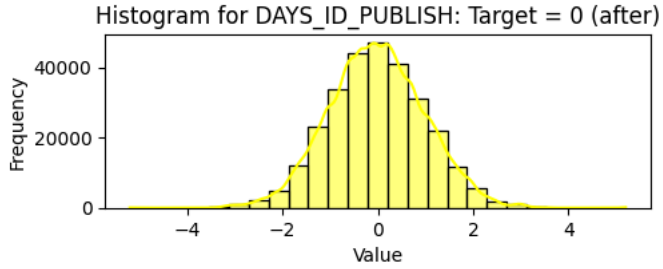



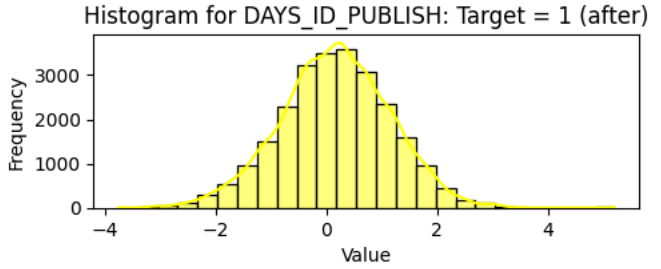




```
Plotting: DAYS_ID_PUBLISH
```

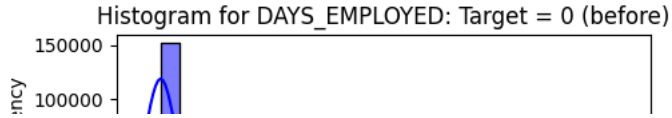


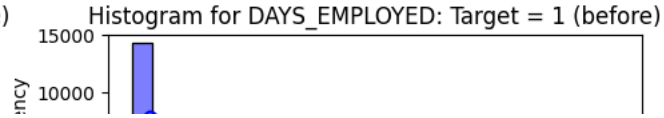




```
Plotting: DAYS_EMPLOYED
```




```
# Running outlier analysis post transformation again
suff = "z_score"
z_score_cols = list()
outlier_df = list()

for c in top5_corr_vars:
  app_train[f"{c}_{suff}"] = zscore(app_train[c])
  over_2 = len(app_train[(app_train[f"{c}_{suff}"].abs() > 2)])
  over_3 = len(app_train[(app_train[f"{c}_{suff}"].abs() > 3)])
  outlier_df.append({"col": c, "over_2": over_2, "over_3": over_3})

outlier_df = pd.DataFrame(outlier_df)
outlier_df
```

| | col | over_2 | over_3 |
|---|---|---|---|
| 0 | DAYS_BIRTH | 13708 | 770 |
| 1 | DAYS_ID_PUBLISH | 14253 | 801 |
| 2 | DAYS_EMPLOYED | 55386 | 0 |
| 3 | DAYS_REGISTRATION | 13929 | 995 |
| 4 | REGION_POPULATION_RELATIVE | 11383 | 8453 |

```
app_train.select_dtypes(include=['object']).columns
```

```
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')
```

**Question 7:** Categorical Features: Check cardinality and rare values of at least five categorical features. Discuss whether each of them is ordinal or nominal. Discuss the suitable methods for encoding each of them.

```
cat_var = list()
cat_var.append("NAME_CONTRACT_TYPE")
cat_var.append("CODE_GENDER")
cat_var.append("ORGANIZATION_TYPE")
cat_var.append("WEEKDAY_APPR_PROCESS_START")
cat_var.append("EMERGENCYSTATE_MODE")

card = list()
card_pct = list()

for v in cat_var:
    u_count = len(app_train[v].unique())
    cat_pct = app_train[column].value_counts(normalize=True) * 100

    cat_pct = pd.DataFrame(app_train[v].value_counts(normalize=True) * 100)
    cat_pct = cat_pct.reset_index().rename(columns={"index": "val", v: "pct"})
    cat_pct["var"] = v

    card_pct.append(cat_pct)
    card.append({"var": v, "cardinality": u_count})

card = pd.DataFrame(card)
card_pct = pd.concat(card_pct, ignore_index=True).reset_index(drop=True)
card_pct.head()
```

| | val | pct | var |
|---|---|---|---|
| 0 | Cash loans | 90.478715 | NAME_CONTRACT_TYPE |
| 1 | Revolving loans | 9.521285 | NAME_CONTRACT_TYPE |
| 2 | F | 65.834393 | CODE_GENDER |
| 3 | M | 34.164306 | CODE_GENDER |
| 4 | XNA | 0.001301 | CODE_GENDER |

```
card
```

| | var | cardinality |
|---|---|---|
| 0 | NAME_CONTRACT_TYPE | 2 |
| 1 | CODE_GENDER | 3 |
| 2 | ORGANIZATION_TYPE | 58 |
| 3 | WEEKDAY_APPR_PROCESS_START | 7 |
| 4 | EMERGENCYSTATE_MODE | 3 |

```
# All values which have percentages less than 5
# Our threshold for categorization intorarity is 5 percent
rare_df = card_pct[card_pct["pct"] < 5]
rare_df
```

| | val | pct | var |
|---|---|---|---|
| **4** | XNA | 0.001301 | CODE_GENDER |
| **9** | Medicine | 3.639870 | ORGANIZATION_TYPE |
| **10** | Business Entity Type 2 | 3.431747 | ORGANIZATION_TYPE |
| **11** | Government | 3.383294 | ORGANIZATION_TYPE |
| **12** | School | 2.891929 | ORGANIZATION_TYPE |
| **13** | Trade: type 7 | 2.546576 | ORGANIZATION_TYPE |
| **14** | Kindergarten | 2.237318 | ORGANIZATION_TYPE |
| **15** | Construction | 2.185613 | ORGANIZATION_TYPE |
| **16** | Business Entity Type 1 | 1.945947 | ORGANIZATION_TYPE |
| **17** | Transport: type 4 | 1.755384 | ORGANIZATION_TYPE |
| **18** | Trade: type 3 | 1.135569 | ORGANIZATION_TYPE |
| **19** | Industry: type 9 | 1.095245 | ORGANIZATION_TYPE |
| **20** | Industry: type 3 | 1.065978 | ORGANIZATION_TYPE |
| **21** | Security | 1.055897 | ORGANIZATION_TYPE |
| **22** | Housing | 0.961917 | ORGANIZATION_TYPE |
| **23** | Industry: type 11 | 0.879318 | ORGANIZATION_TYPE |