Home Credit EDA

Business Problem Statement

The problem is that Home Credit currently serves clients who cannot be served by the traditional banking system or other traditional financial institutions. These clients cannot be served by traditional institutions because they are usually from underprivileged populations. This results in them having insufficient/non-existent credit history. It also makes it difficult for Home Credit to utilize traditional measures like FICO Scores to see if a client can repay their loan.

If Home Credit approves a loan for a client and they cannot pay the loan back, then it is a financial loss for Home Credit. However, if a client can pay back a loan but is denied, it represents a loss of potential revenue for Home Credit. Both scenarios ultimately affect Home Credit's ability to operate efficiently because Home Credit either loses money from bad loans or forgoes lending opportunities.

Load Libraries

```
# load libraries
pacman::p_load(tidyverse,skimr,janitor,knitr,caret,rminer,mice,dbscan)
```

Read in Datasets

```
# read in datasets
train_set <- read_csv("C:\\Users\\User\\Box Sync\\Business Analytics Degree\\Semesters\\Fall Seme
```

There are 7 extra files in addition to the train_set that can be utilized for prediction. They are all connected primarily via the SK_ID_CURR variable. The EDA will primarily focus on the train_set. It should be noted that other data sets will be incorporated to help improve the models if needed.

View Data

```
head(train_set) # get first 6 rows of dataset
```

```
# A tibble: 6 × 122
  SK_ID_CURR TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY
       <dbl>  <dbl> <chr>              <chr>       <chr>        <chr>
1     100002      1 Cash loans         M           N            Y
2     100003      0 Cash loans         F           N            N
3     100004      0 Revolving loans    M           Y            Y
4     100006      0 Cash loans         F           N            Y
```

```
5     100007      0 Cash loans         M           N            Y
6     100008      0 Cash loans         M           N            Y
# ℹ 116 more variables: CNT_CHILDREN <dbl>, AMT_INCOME_TOTAL <dbl>,
#   AMT_CREDIT <dbl>, AMT_ANNUITY <dbl>, AMT_GOODS_PRICE <dbl>,
#   NAME_TYPE_SUITE <chr>, NAME_INCOME_TYPE <chr>, NAME_EDUCATION_TYPE <chr>,
#   NAME_FAMILY_STATUS <chr>, NAME_HOUSING_TYPE <chr>,
#   REGION_POPULATION_RELATIVE <dbl>, DAYS_BIRTH <dbl>, DAYS_EMPLOYED <dbl>,
#   DAYS_REGISTRATION <dbl>, DAYS_ID_PUBLISH <dbl>, OWN_CAR_AGE <dbl>,
#   FLAG_MOBIL <dbl>, FLAG_EMP_PHONE <dbl>, FLAG_WORK_PHONE <dbl>, …
```

Exploratory Questions

There are several variables in this dataset that have interesting relationships. The following contains a list of questions:

- Is there a relationship between a client's contract type and default status?
- What is the relationship between a client's occupation and default status?
- What is the relationship between a client's education and default status?
- What is the relationship between a client's marital status and default status?
- What is the relationship between a client's income type and default status?
- Is there a relationship between the credit scores and a client's default status?
- What is the relationship between all the annuity amount, annuity credit, the cost of the item that has caused them to apply for a loan, etc.
- Is there a relationship between gender and a client's default status?
- Does the age of a client's car have any contribution, since cars are typically used for collateral in loans?
- Does the age of the client have any impact on their ability to get loans, credit etc? Could it impact their ability to purchase a car which could potentially be used as collateral? Consequently would this affect their ability to repay a loan?

Class Prevalence

```r
table(train_set$TARGET) # Get a count of the target, default and no default.
```

```

     0      1 
282686  24825 
```

```r
round(prop.table(table(train_set$TARGET)),4) * 100 # Multiply by 100 to get percentages
```

```
     0      1 
91.93  8.07 
```

Clients who have difficulties paying back loans are represented as a 1 in the dataset while 0 represents all the other cases. (0 presumably means that the client was able to pay back their loan but may have had some case not related to paying back loans.) The clients who paid back their loans is more prevalent at 91.93% while the clients who had payment difficulties appear at 8.07%. This means that there is a large class imbalance.

It should be noted that the objective is to predict clients who can pay back their loans. Hence, since 91.93% of the clients paid back their loan, the model has a lot of information to learn about successful repayment.

Build Majority Classifier

```r
# Define the counts
yes_count <- 282686
no_count <- 24825

# Calculate the total number of observations
total_count <- yes_count + no_count

# Majority class
majority_class <- ifelse(yes_count > no_count, 1, 0) # 1 if yes_count is greater than no count

# Convert Majority Class to factor
majority_class <- factor(majority_class)

#Generate the majority classifier predictions for all instances
predicted <- rep(majority_class, total_count) # Total count represents the total number of instan
# since majority class is zero, this will be repeated for every prediction or total number of ins

# Create metrics list for Majority Classifier
metrics_list = c("ACC","TPR","PRECISION", "F1", "CONF")

# Generate metrics
majority_output <- mmetric(factor(train_set$TARGET), predicted, metrics_list) #Set all the metric

majority_output_rounded <- lapply(majority_output, function(x) { # Utilize function to round off 
  if (is.numeric(x)) {
    return(round(x, 2))
  } else {
    return(x)
  }
})
print(majority_output_rounded)
```

```
$res
       ACC       TPR1       TPR2  PRECISION1  PRECISION2        F11        F12
```

```
     91.93     100.00       0.00       91.93       0.00       95.79       0.00

$conf
      pred
target      0      1
     0 282686      0
     1  24825      0
```

TPR stands for Recall

The majority classifier has an overall accuracy of 91.93%. This is however because the majority class (0 - pay back loan) comprises the dataset majority. (Majority Classifiers predict the majority class for everything which means that accuracy will be the majority class value.)

However since the objective is to predict the clients who can pay back their loans, the majority class accuracy can be used as the baseline accuracy. This also means that any future model will have to beat this majority class accuracy. (91.93%)

Additional metrics that may be desirable to be beat is `Precision 1` and `F1-1` . It should be noted that beating the recall for 1(pay back loan) will not be possible since it is 100%. This is because TPR1 represents the majority class and the majority classifier will make no false negative errors. Thus, everything will be classified as positive which divided by the total positive observations results in 100.

- True Positive represents correctly predicting that somebody will be able to pack a loan.
- True Negative represents correctly predicting that somebody will have difficulty paying back a loan.
- False Negative represents incorrectly predicting difficulty paying back the loan.
- False Positive represents incorrectly predicting that somebody will pay back the loan when they actually will have difficulty paying back the loan.

Build Random Classifier

```
class_labels = c(0,1) # Create a class label with 2 labels, 0 and 1
class_labels <- factor(class_labels) # Convert Class labels to a factor

set.seed(123) # Set seed for reproducibility
predictions <- sample(x = class_labels, size = total_count,replace = TRUE, prob = c(.5,.5)) # Samp

predictions <- factor(predictions) # Convert the predictions into a factor

random_output <- mmetric(factor(train_set$TARGET),predictions,metrics_list) # Generate metrics


random_output_rounded <- lapply(random_output, function(x) { # Utilize function to round off metr
```

```
  if (is.numeric(x)) {
    return(round(x, 0))
  } else {
    return(x)
  }
})
print(random_output_rounded) # Print out the rounded metrics
```

```
$res
       ACC       TPR1       TPR2 PRECISION1 PRECISION2        F11        F12 
        50         50         50         92          8         65         14 

$conf
      pred
target      0      1
     0 140944 141742
     1  12473  12352
```

The random classifier model has an overall accuracy of 50% due to randomly assigning an observation to either default or no default with 50% probability. Consequently, this has also affected recall which measures the following:

predicted class/total observations in the actual class

The objective of any model built will be to have an accuracy higher than 50% and to have a higher recall (greater than TPR1), higher precision1, and a higher F11 score as well.

Check Low Variance Columns

Create two datasets

```
train_clean <- train_set # Assign train_set to train_clean set
train_set_02 <- train_set # Assign train_set to train_set_02
```

train_clean will be used to officially implement any changes, data cleaning etc.

train_set_02 will be used to experiment on the dataset and try changes out before officially implementing it in the train_clean.

Additionally, it is important to retain the original dataset as a backup.

Define Function to Check for NA's

```
# Define function to check for NAs
find_na <- function(x) sum(is.na(x))

# Apply function to each column with map()
missing_values <- map(.x = train_set, .f = find_na) %>% # Assign changes to a new variable
  unlist() %>%
  data.frame()

missing_values <- missing_values %>% # Overwrite the missing_values with a new dataframe
  rownames_to_column(var = "Column") %>%
  rename(Missing_Values = 1) # Rename the unnamed column to "Missing_Values"
```

This code will create a dataframe that displays every variable and how many missing values per variable.

Cross Reference Low Variance Columns with no Missing Values

The missing values will be cross referenced with the Low Variance columns in order to see which Low Variance Columns have no missing values.

```
# Identify near zero variance predictors by name
near_zero_variance <- nearZeroVar(train_set, names = TRUE)


#Check the near zero variance columns that have no missing values
cols_with_zero_missing <- missing_values[missing_values$Missing_Values %in% near_zero_variance & 

# Calculate the variances for these columns
zero_missing_cols <- cols_with_zero_missing$Missing_Values
variances_zero_missing <- sapply(zero_missing_cols, function(col) var(train_set[[col]], na.rm = T

#Print the results
print(variances_zero_missing)
```

```
           DAYS_EMPLOYED               FLAG_MOBIL
            1.995884e+10             3.251916e-06
        FLAG_CONT_MOBILE  REG_REGION_NOT_LIVE_REGION
            1.863122e-03             1.491488e-02
LIVE_REGION_NOT_WORK_REGION          FLAG_DOCUMENT_2
            3.900570e-02             4.227326e-05
         FLAG_DOCUMENT_4          FLAG_DOCUMENT_5
            8.129156e-05             1.488649e-02
         FLAG_DOCUMENT_7          FLAG_DOCUMENT_9
            1.918269e-04             3.880631e-03
        FLAG_DOCUMENT_10         FLAG_DOCUMENT_11
```

```
       2.276297e-05        3.896764e-03
    FLAG_DOCUMENT_12    FLAG_DOCUMENT_13
       6.503811e-06        3.512662e-03
    FLAG_DOCUMENT_14    FLAG_DOCUMENT_15
       2.927867e-03        1.208253e-03
    FLAG_DOCUMENT_16    FLAG_DOCUMENT_17
       9.829565e-03        2.665869e-04
    FLAG_DOCUMENT_18    FLAG_DOCUMENT_19
       8.063723e-03        5.947485e-04
    FLAG_DOCUMENT_20    FLAG_DOCUMENT_21
       5.070432e-04        3.348363e-04
```

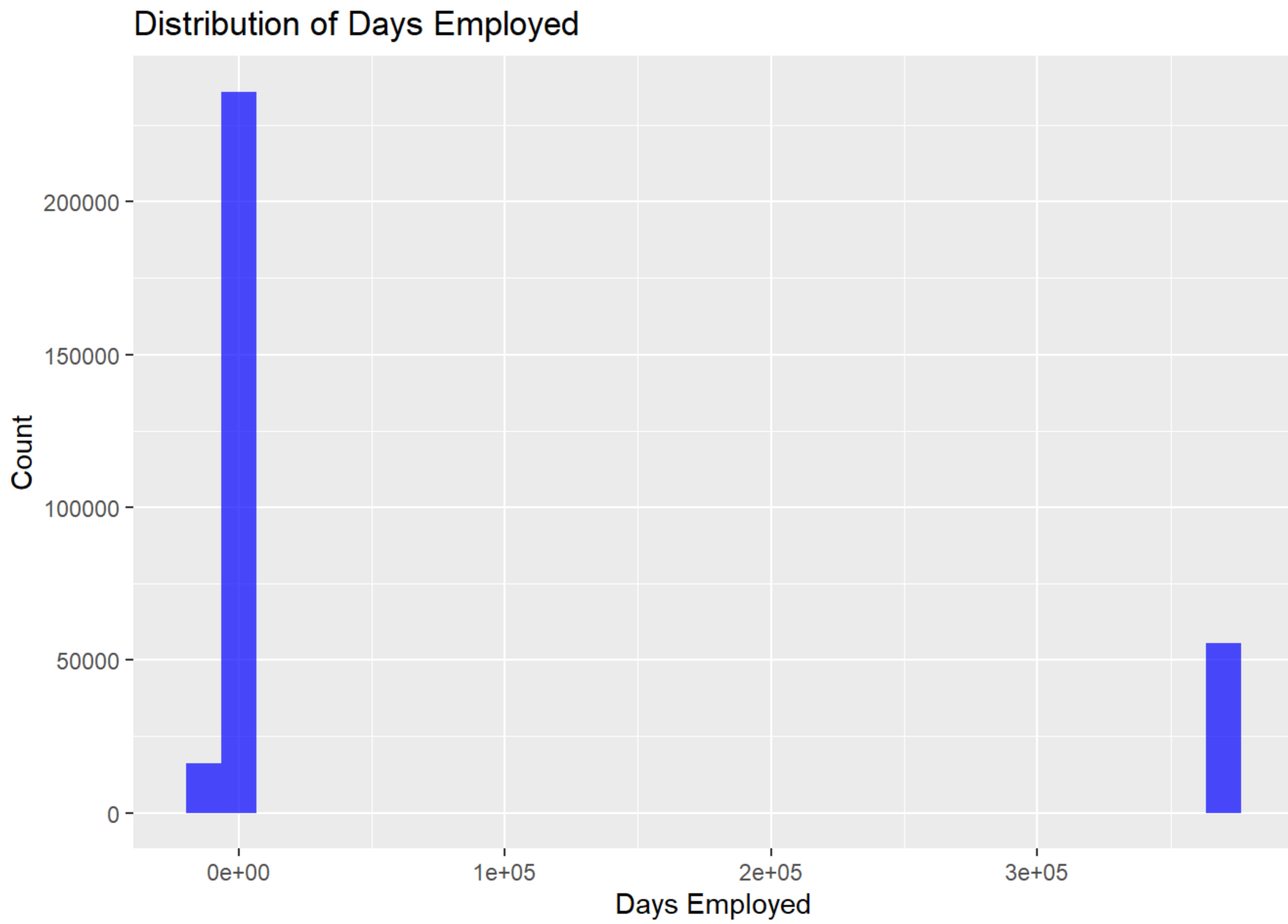
The variances for these columns are very small and incredibly close to zero. It is very unlikely that these predictors will contain any relevant information for prediction. It additionally may cause issues with cross validation, and slow down prediction algorithms since the dataset's dimensionality is large.

- Additionally, for the `Flag_Document_Columns`, it is unclear what Document 1 to 20 represents. The full context is unknown, which makes it difficult to glean any useful information from these columns. Thus, any `Flag_Document_Columns` that has low variance can be dropped.
- Moreover, for the `REG_REGION_NOT_LIVE_REGION` and `LIVE_REGION_NOT_WORK_REGION`, there are similar predictors that measure similar characteristics like `REG_REGION_NOT_WORK_REGION`, `REG_CITY_NOT_LIVE_CITY`, etc. Hence, these two columns can be dropped as well.

The `Days Employed` column however requires further exploration. This is because `Days Employed` has a very high variance at 19,958,840,000. Thus, further investigation on this discrepancy is required.

Plot Histogram for Days Employed

```
# Create a histogram to visualize the distribution
ggplot(train_set, aes(x = DAYS_EMPLOYED)) +
  geom_histogram(bins = 30, fill = "blue", alpha = 0.7) +
  labs(title = "Distribution of Days Employed", x = "Days Employed", y = "Count")
```



The histogram above shows that Days Employed is a bimodal distribution with one large peak and a smaller peak around 0 days. It also has a count that is greater than 250,000 when both peaks are combined. The dataset has 307,511 rows which means that 0 days comprises roughly 81.3% of the rows in the dataset.

This would also explain why the Near Zero Variance Function considered this to be low variance since most of the values were clustered around zero relative to the rest of the data distribution. There are a few things to note however:

- The high number of observations around zero seems unrealistic in the context of the Days Employed variable. The “Column Description” file states that this variable represents how many days a client started employment before the application. It seems very unlikely that 81% of the applicants started their employment and got a loan on the same day. The 0 days thus seems to be some sort of rubber-stamp procedure or some recording policy Home Credit has created.

- One could argue that Home Credit serves people who can’t qualify for traditional loans. Thus, there will be some unusual discrepancies/features when compared to a more traditional loan approval dataset. Although this may be true, it still seems very unlikely that clients would get a job and apply for a loan on the same day. Other predictors like credit card scores might be different for this dataset, however this predictor does not seem like that.

The other peak occurs around 3.5e+05 which represents 350,000 days or 958 years.

- This is a very unrealistic value.

Since the values for this histogram are very unrealistic, this predictor will be dropped from the dataset.

Drop near zero variance predictors (no missing values)

```
train_clean <- train_clean %>% #Drop all the selected predictors
  select(-DAYS_EMPLOYED,-FLAG_MOBIL, -FLAG_CONT_MOBILE,  -REG_REGION_NOT_LIVE_REGION, -LIVE_REGIO
- FLAG_DOCUMENT_20, -FLAG_DOCUMENT_21)
```

These columns are the low variance columns that have no missing values which will be dropped from the `train_clean` dataset.

Cross Reference Low Variance Columns with Missing Values

The rest of the low variance columns will be cross referenced with the missing values to see which columns have both low variance and missing values.

```
# Check the near zero variance columns that have missing values
cols_with_missing <- missing_values[missing_values$Missing_Values %in% near_zero_variance & missi


# Calculate the variances for these columns
missing_cols <- cols_with_missing$Missing_Values
variances_missing <- sapply(missing_cols, function(col) var(train_set[[col]], na.rm = TRUE))


missing_count <- sapply(missing_cols, function(col) sum(is.na(train_set[[col]])))

# Combine the variances and missing counts into a data frame
results_with_missing <- data.frame(
  Variable = names(variances_missing),
  Variance = variances_missing,
  Missing_Count = missing_count
)

# Print the results
print(results_with_missing)
```

```
                                Variable    Variance Missing_Count
BASEMENTAREA_AVG            BASEMENTAREA_AVG 0.006796050        179943
LANDAREA_AVG                    LANDAREA_AVG 0.006590784        182590
NONLIVINGAREA_AVG          NONLIVINGAREA_AVG 0.004833473        169682
BASEMENTAREA_MODE          BASEMENTAREA_MODE 0.007107700        179943
LANDAREA_MODE                  LANDAREA_MODE 0.006683108        182590
NONLIVINGAREA_MODE        NONLIVINGAREA_MODE 0.004935605        169682
```

```
BASEMENTAREA_MEDI                    BASEMENTAREA_MEDI 0.006753347        179943
LANDAREA_MEDI                            LANDAREA_MEDI 0.006751418        182590
NONLIVINGAREA_MEDI                  NONLIVINGAREA_MEDI 0.004923335        169682
HOUSETYPE_MODE                          HOUSETYPE_MODE          NA        154297
EMERGENCYSTATE_MODE                EMERGENCYSTATE_MODE          NA        145755
AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_HOUR 0.007030676         41519
AMT_REQ_CREDIT_BUREAU_DAY    AMT_REQ_CREDIT_BUREAU_DAY 0.012267203         41519
AMT_REQ_CREDIT_BUREAU_WEEK AMT_REQ_CREDIT_BUREAU_WEEK 0.041895898         41519
```

The following variables all describe different aspectes of the client's home which has been normalized per the Home Credit Columns Description File:

- `BASEMENTAREA_AVG`
- `LANDAREA_AVG`
- `NONLIVINGAREA_AVG`
- `BASEMENTAREA_MODE`
- `LANDAREA_MODE`
- `NONLIVINGAREA_MODE`
- `BASEMENTAREA_MEDI`
- `LANDAREA_MEDI`
- `NONLIVINGAREA_MEDI`

The next following set of variables describe similar metrics per the Home Credit Columns Description File:

- `AMT_REQ_CREDIT_BUREAU_HOUR`
- `AMT_REQ_CREDIT_BUREAU_DAY`
- `AMT_REQ_CREDIT_BUREAU_WEEK`

The last following set of variables that are similar and have low variance are:

- `HOUSETYPE_MODE`
- `EMERGENCYSTATE_MODE`

Thus, to increase the variance, some of these predictors will be combined into one predictor via Principal Component Analysis.

Build Housing Related Correlation Matrix for PCA

```
# Define the predictors to check for correlation
predictors_to_combine <- c("BASEMENTAREA_AVG", "LANDAREA_AVG", "NONLIVINGAREA_AVG","NONLIVINGAREA_

# Calculate the correlation matrix (ignoring missing values)
correlation_matrix <- cor(train_set_02[predictors_to_combine], use = "complete.obs")

# Display the correlation matrix
correlation_matrix
```

```
                   BASEMENTAREA_AVG LANDAREA_AVG NONLIVINGAREA_AVG
BASEMENTAREA_AVG          1.0000000    0.4632255         0.2669469
LANDAREA_AVG              0.4632255    1.0000000         0.1633207
NONLIVINGAREA_AVG         0.2669469    0.1633207         1.0000000
NONLIVINGAREA_AVG         0.2669469    0.1633207         1.0000000
BASEMENTAREA_MODE         0.9717283    0.4665704         0.2562097
LANDAREA_MODE             0.4573192    0.9713993         0.1568702
NONLIVINGAREA_MODE        0.2605044    0.1624556         0.9600620
BASEMENTAREA_MEDI         0.9939003    0.4648647         0.2666730
LANDAREA_MEDI             0.4656233    0.9908201         0.1631723
NONLIVINGAREA_MEDI        0.2660521    0.1648120         0.9880936
                   NONLIVINGAREA_AVG BASEMENTAREA_MODE LANDAREA_MODE
BASEMENTAREA_AVG           0.2669469         0.9717283     0.4573192
LANDAREA_AVG               0.1633207         0.4665704     0.9713993
NONLIVINGAREA_AVG          1.0000000         0.2562097     0.1568702
NONLIVINGAREA_AVG          1.0000000         0.2562097     0.1568702
BASEMENTAREA_MODE          0.2562097         1.0000000     0.4791006
LANDAREA_MODE              0.1568702         0.4791006     1.0000000
NONLIVINGAREA_MODE         0.9600620         0.2728346     0.1709925
BASEMENTAREA_MEDI          0.2666730         0.9764608     0.4606966
LANDAREA_MEDI              0.1631723         0.4707552     0.9791699
NONLIVINGAREA_MEDI         0.9880936         0.2614236     0.1614583
                   NONLIVINGAREA_MODE BASEMENTAREA_MEDI LANDAREA_MEDI
BASEMENTAREA_AVG            0.2605044         0.9939003     0.4656233
LANDAREA_AVG                0.1624556         0.4648647     0.9908201
NONLIVINGAREA_AVG           0.9600620         0.2666730     0.1631723
NONLIVINGAREA_AVG           0.9600620         0.2666730     0.1631723
BASEMENTAREA_MODE           0.2728346         0.9764608     0.4707552
LANDAREA_MODE               0.1709925         0.4606966     0.9791699
NONLIVINGAREA_MODE          1.0000000         0.2633458     0.1640328
BASEMENTAREA_MEDI           0.2633458         1.0000000     0.4677095
LANDAREA_MEDI               0.1640328         0.4677095     1.0000000
NONLIVINGAREA_MEDI          0.9723618         0.2682970     0.1661594
                   NONLIVINGAREA_MEDI
BASEMENTAREA_AVG            0.2660521
LANDAREA_AVG                0.1648120
NONLIVINGAREA_AVG           0.9880936
NONLIVINGAREA_AVG           0.9880936
BASEMENTAREA_MODE           0.2614236
LANDAREA_MODE               0.1614583
NONLIVINGAREA_MODE          0.9723618
BASEMENTAREA_MEDI           0.2682970
LANDAREA_MEDI               0.1661594
NONLIVINGAREA_MEDI          1.0000000
```

Some variables are highly correlated with each other like `LANDAREA_MEDI` and `LANDAREA_AVG` at .99. Other variables however are somewhat correlated like `BASEMENTAREA_MEDI` and `LANDAREA_MEDI` at 0.46.

This indicates that Principal Component Analysis should be fairly appropriate for combining these columns into one predictor.

Combine Housing Related Predictors

```
for (col in predictors_to_combine) { # For-Loop for all the columns in the pred_to_combine vector
  train_clean[[col]] <- ifelse(is.na(train_clean[[col]]), # Get all the columns from pred_to_comb
  median(train_clean[[col]], na.rm = TRUE),train_clean[[col]]) # Impute those values with the med
}

pca_result <- prcomp(train_clean[predictors_to_combine], na.action = na.omit, scale. = TRUE) # As

summary(pca_result) # Get summary of PCA
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7
Standard deviation     2.2732 1.7605 1.2693 0.23396 0.16348 0.13662 0.09901
Proportion of Variance 0.5167 0.3099 0.1611 0.00547 0.00267 0.00187 0.00098
Cumulative Proportion  0.5167 0.8267 0.9878 0.99326 0.99593 0.99780 0.99878
                           PC8     PC9      PC10
Standard deviation     0.08398 0.07196 8.918e-13
Proportion of Variance 0.00071 0.00052 0.000e+00
Cumulative Proportion  0.99948 1.00000 1.000e+00
```

```
# Extract the first principal component
pca_scores <- pca_result$x[, 1]  # First principal component

train_clean$House_Attribute_Low_Variance <- pca_scores #Create a new col and assign pca scores to
var(train_clean$House_Attribute_Low_Variance) # Show the variance of the house_attribute column
```

```
[1] 5.167377
```

The Housing Related Predictors with low columns have first had any missing values imputed with the median. Afterwards, the predictors have been combined into one column using PCA.

Remove all the housing related low variance predictors

```
train_clean <- train_clean %>% #Remove the following predictors
  select(-BASEMENTAREA_AVG, -LANDAREA_AVG, -NONLIVINGAREA_AVG,-NONLIVINGAREA_AVG,-BASEMENTAREA_MOI
```

It should be noted that all the housing columns that have low variance have been combined into one column, which means that the rest of the housing columns can be removed.

Build Bureau Correlation Matrix for PCA

```r
# b = bureau
# Define the predictors you want to check for correlation
b_predictors_to_combine <- c("AMT_REQ_CREDIT_BUREAU_HOUR",
"AMT_REQ_CREDIT_BUREAU_DAY", "AMT_REQ_CREDIT_BUREAU_WEEK")

# Calculate the correlation matrix (ignoring missing values)
b_correlation_matrix <- cor(train_set_02[b_predictors_to_combine], use = "complete.obs")

#Display correlation matrix
b_correlation_matrix
```

```
                           AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY
AMT_REQ_CREDIT_BUREAU_HOUR                1.000000000                 0.2303743
AMT_REQ_CREDIT_BUREAU_DAY                 0.230374292                 1.0000000
AMT_REQ_CREDIT_BUREAU_WEEK                0.004705609                 0.2174118
                           AMT_REQ_CREDIT_BUREAU_WEEK
AMT_REQ_CREDIT_BUREAU_HOUR                0.004705609
AMT_REQ_CREDIT_BUREAU_DAY                 0.217411847
AMT_REQ_CREDIT_BUREAU_WEEK                1.000000000
```

The correlation is pretty weak, nothing very strong, which means that PCA would be inappropriate. The counts for the variable however should still be analyzed.

Count of Bureau Required Variables

```r
# Get a count of how many inquiries
table(train_set$AMT_REQ_CREDIT_BUREAU_HOUR)
```

```
     0      1      2      3      4 
264366   1560     56      9      1 
```

```r
table(train_set$AMT_REQ_CREDIT_BUREAU_DAY)
```

```
     0      1      2      3      4      5      6      8      9 
264503   1292    106     45     26      9      8      1      2 
```

```r
table(train_set$AMT_REQ_CREDIT_BUREAU_MON)
```

```
     0      1      2      3      4      5      6      7      8      9     10 
222233  33147   5386   1991   1076    602    343    298    185    206    132 
    11     12     13     14     15     16     17     18     19     22     23 
   119     77     72     40     35     23     14      6      3      1      1 
    24     27 
     1      1 
```

The number of inquires seems relatively limited with month having the highest amount of inquires at 27 for one observation.

Although these variables have low variance, they do seem like they could still be useful for predicting whether someone will pay back a loan or not.

However, NA values must be first addressed.

Address Missing Bureau Required Values

```
# AMT_REQ_CREDIT_BUREAU_HOUR
train_clean$AMT_REQ_CREDIT_BUREAU_HOUR[is.na(train_set$AMT_REQ_CREDIT_BUREAU_HOUR)] <- 0
sum(is.na(train_clean$AMT_REQ_CREDIT_BUREAU_HOUR))
```

```
[1] 0
```

```
# AMT_REQ_CREDIT_BUREAU_DAY
train_clean$AMT_REQ_CREDIT_BUREAU_DAY[is.na(train_set$AMT_REQ_CREDIT_BUREAU_DAY)] <- 0
sum(is.na(train_clean$AMT_REQ_CREDIT_BUREAU_DAY))
```

```
[1] 0
```

```
# AMT_REQ_CREDIT_BUREAU_MON
train_clean$AMT_REQ_CREDIT_BUREAU_MON[is.na(train_set$AMT_REQ_CREDIT_BUREAU_MON)] <- 0
sum(is.na(train_clean$AMT_REQ_CREDIT_BUREAU_MON))
```

```
[1] 0
```

The NA values have two possible meanings:

- There is genuinely no information on the inquiries made. This seems unlikely that Home Credit would forget to record how many inquires they make about clients.
- The NA values could be another way of representing zero inquiries or no information which could imply zero inquiries. Thus the NA values will be imputed as zero.

Check Counts of HOUSETYPE_MODE and EMERGENCYSTATE_MODE

```
# Get a count of the values
table(train_set_02$HOUSETYPE_MODE)
```

```

  block of flats specific housing   terraced house 
          150503             1499             1212 
```

```
# Get a count of the values
table(train_set_02$EMERGENCYSTATE_MODE)
```

```
    No    Yes 
159428   2328 
```

These seem like categorical variables that have distinct levels. The missing values therefore can be addressed with imputation.

Impute Missing Values for HOUSETYPE_MODE

```
# Define a function in R that finds the mode
find_mode <- function(x){
  table(x) %>%
    which.max() %>% #Find which category appears the most
    names() #Get the name of this category
}

housetype_mode <- find_mode(train_clean$HOUSETYPE_MODE) #Use this function to find mode and assig


train_clean$HOUSETYPE_MODE[is.na(train_clean$HOUSETYPE_MODE)] <- housetype_mode #Impute the train_
```

The missing values for `HOUSETYPE_MODE` have been imputed with the mode.

Convert HOUSETYPE_MODE to factor

```
#Convert HouseType_Mode to a factor
train_clean$HOUSETYPE_MODE <- as.factor(train_clean$HOUSETYPE_MODE)
```

The `Housetype Mode` column also consists of categorical values which have no meaningful numeric difference. Thus, this variable will be converted into a factor.

Impute Missing Values for EMERGENCYSTATE_MODE

```
# Get a count of the values
table(train_set_02$EMERGENCYSTATE_MODE)
```

```
    No    Yes 
159428   2328 
```

```
# See how many missing values are in the columns
sum(is.na(train_set_02$EMERGENCYSTATE_MODE))
```

```
[1] 145755
```

```
# Set variable with string "Unknown"
unknown <- "Unknown"

# Replace all NA values with this "Unkown" value
train_clean$EMERGENCYSTATE_MODE[is.na(train_clean$EMERGENCYSTATE_MODE)] <- unknown

# Convert Variable to a factor
train_clean$EMERGENCYSTATE_MODE <- as.factor(train_clean$EMERGENCYSTATE_MODE)
```

As shown previously, there are only 2 values Emergency State Mode. (Emergency State Mode seems to describe the state of the client's residence)

There are however missing values that must be addressed. The missing values could be due to the client not providing the information or Home Credit not insisting on the information, etc.

This however does not necessarily mean that the information is missing. Instead, the information could be unknown which in itself could be a helpful predictor. Thus, the `NA` values will be imputed with "unknown" which will also enable various algorithms to utilize the "unknown" when making predictions.

Check Missing Variables

How many missing columns

```
#tc = train_clean
missing_values_tc <- map(.x = train_clean, .f = find_na) %>%
  unlist() %>%
  data.frame() #Assign the df w/ all missing values to train_clean

missing_values_tc <- missing_values_tc %>% #override missing_values_tc with new changes
  rownames_to_column(var = "Column") %>%
  rename(Missing_Values = 1) # Rename the unnamed column to "Missing_Values"

missing_values_tc %>%
  filter( . != 0) %>% #Show only missing columns
  summarise(missing_col_cnt  = length(.), # compute missing values as percentages
            missing_col_pctg = (length(.)/122) * 100) %>% round(2)
```

```
  missing_col_cnt missing_col_pctg
1              53            43.44
```

Since the low variance columns and its missing values have been addressed, the rest of the dataset will be examined for any missing values. The train_clean dataset will be utilized since some of the low variance columns were dropped. Thus utilizing this dataset will give us a more accurate count.

Overall, there are 53 columns with missing values which comprise 43.44% of this dataset.

Names of missing columns

```
missing_values_tc %>%
  filter( . != 0) %>% # Show only non-zero columns
  arrange(desc(.)) # Arrange in descending order
```

```
                  Missing_Values      .
1                 COMMONAREA_AVG 214865
2                COMMONAREA_MODE 214865
3                COMMONAREA_MEDI 214865
4         NONLIVINGAPARTMENTS_AVG 213514
5        NONLIVINGAPARTMENTS_MODE 213514
6        NONLIVINGAPARTMENTS_MEDI 213514
7             FONDKAPREMONT_MODE 210295
8           LIVINGAPARTMENTS_AVG 210199
9          LIVINGAPARTMENTS_MODE 210199
10         LIVINGAPARTMENTS_MEDI 210199
11                 FLOORSMIN_AVG 208642
12                FLOORSMIN_MODE 208642
13                FLOORSMIN_MEDI 208642
14               YEARS_BUILD_AVG 204488
15              YEARS_BUILD_MODE 204488
16              YEARS_BUILD_MEDI 204488
17                   OWN_CAR_AGE 202929
18                  EXT_SOURCE_1 173378
19                 ELEVATORS_AVG 163891
20                ELEVATORS_MODE 163891
21                ELEVATORS_MEDI 163891
22            WALLSMATERIAL_MODE 156341
23                APARTMENTS_AVG 156061
24               APARTMENTS_MODE 156061
25               APARTMENTS_MEDI 156061
26                 ENTRANCES_AVG 154828
27                ENTRANCES_MODE 154828
28                ENTRANCES_MEDI 154828
29                LIVINGAREA_AVG 154350
30               LIVINGAREA_MODE 154350
31               LIVINGAREA_MEDI 154350
32                 FLOORSMAX_AVG 153020
33                FLOORSMAX_MODE 153020
34                FLOORSMAX_MEDI 153020
35   YEARS_BEGINEXPLUATATION_AVG 150007
36  YEARS_BEGINEXPLUATATION_MODE 150007
37  YEARS_BEGINEXPLUATATION_MEDI 150007
38                TOTALAREA_MODE 148431
39               OCCUPATION_TYPE  96391
40                  EXT_SOURCE_3  60965
41    AMT_REQ_CREDIT_BUREAU_WEEK  41519
42     AMT_REQ_CREDIT_BUREAU_QRT  41519
```

```
43   AMT_REQ_CREDIT_BUREAU_YEAR  41519
44                NAME_TYPE_SUITE   1292
45       OBS_30_CNT_SOCIAL_CIRCLE   1021
46       DEF_30_CNT_SOCIAL_CIRCLE   1021
47       OBS_60_CNT_SOCIAL_CIRCLE   1021
48       DEF_60_CNT_SOCIAL_CIRCLE   1021
49                   EXT_SOURCE_2    660
50                AMT_GOODS_PRICE    278
51                    AMT_ANNUITY     12
52                CNT_FAM_MEMBERS      2
53         DAYS_LAST_PHONE_CHANGE      1
```

These are the actual names of the columns that are missing values. Interestingly several of them seem to be related and have the same missing values. For instance:

- `NONLIVINGAPARTMENTS_AVG`
- `NONLIVINGAPARTMENTS_MODE`
- `NONLIVINGAPARTMENTS_MEDI`

These 3 variables are all missing 213,514 observations.

Columns greater than 50%

```
missing_values_tc %>%
  filter( . != 0) %>% #Show only non-zero columns
  mutate(col_pctg_missing_values = ./nrow(train_clean)) %>% # Get decimal by dividing by # of rows
  filter(col_pctg_missing_values >=.50) %>% # Filter values that are greater than 50%
  select(Missing_Values,col_pctg_missing_values) %>% # Select only missing values and the pctg.
  arrange(desc(col_pctg_missing_values)) # Arrange in descending order
```

```
             Missing_Values col_pctg_missing_values
1            COMMONAREA_AVG               0.6987230
2           COMMONAREA_MODE               0.6987230
3           COMMONAREA_MEDI               0.6987230
4    NONLIVINGAPARTMENTS_AVG              0.6943296
5   NONLIVINGAPARTMENTS_MODE              0.6943296
6   NONLIVINGAPARTMENTS_MEDI              0.6943296
7        FONDKAPREMONT_MODE               0.6838617
8      LIVINGAPARTMENTS_AVG               0.6835495
9     LIVINGAPARTMENTS_MODE               0.6835495
10    LIVINGAPARTMENTS_MEDI               0.6835495
11            FLOORSMIN_AVG               0.6784863
12           FLOORSMIN_MODE               0.6784863
13           FLOORSMIN_MEDI               0.6784863
14           YEARS_BUILD_AVG              0.6649778
15          YEARS_BUILD_MODE              0.6649778
16          YEARS_BUILD_MEDI              0.6649778
17               OWN_CAR_AGE              0.6599081
```

```
18             EXT_SOURCE_1             0.5638107
19            ELEVATORS_AVG             0.5329598
20           ELEVATORS_MODE             0.5329598
21           ELEVATORS_MEDI             0.5329598
22       WALLSMATERIAL_MODE             0.5084078
23           APARTMENTS_AVG             0.5074973
24          APARTMENTS_MODE             0.5074973
25          APARTMENTS_MEDI             0.5074973
26            ENTRANCES_AVG             0.5034877
27           ENTRANCES_MODE             0.5034877
28           ENTRANCES_MEDI             0.5034877
29           LIVINGAREA_AVG             0.5019333
30          LIVINGAREA_MODE             0.5019333
31          LIVINGAREA_MEDI             0.5019333
```

These are all the variables that have 50% or more of the values missing. It should be noted that the vast majority of these variables are all related. The variables are summary statistics describing various information about a client's residence.

For instance:

- `COMMONAREA_AVG`
- `COMMONAREA_MODE`
- `COMMONAREA_MEDI`

These three columns are describing the average size of a common area, the mode of the common area (how many common areas are in a client's residence) and the median size of a common area.

The other variables are also describing similar metrics for other parts of the client's residence like the living room, etc.

Create a Correlation Matrix for Housing Attribute Variables

```r
# Extract relevant columns for COMMONAREA
commonarea_cols <- train_clean %>%
  select(contains("COMMONAREA"))
# Run correlation for COMMONAREA columns
cor_commonarea <- cor(commonarea_cols, use = "complete.obs")

# Extract relevant columns for NONLIVINGAPARTMENTS
nonliving_cols <- train_clean %>%
  select(contains("NONLIVINGAPARTMENTS"))
# Run correlation for NONLIVINGAPARTMENTS columns
cor_nonliving <- cor(nonliving_cols, use = "complete.obs")

# Select only the columns related to 'LIVINGAPARTMENTS'
living_apartments_cols <- train_clean %>%
  select(starts_with("LIVINGAPARTMENTS"))
# Calculate the correlation matrix for these columns
cor_living <- cor(living_apartments_cols, use = "complete.obs")
```

```
# Select only the columns related to 'FLOORSMIN'
floorsmin_col <- train_clean %>%
  select(starts_with("FLOORSMIN"))
# Calculate the correlation matrix for these columns
cor_floorsmin <- cor(floorsmin_col, use = "complete.obs")

# Select only the columns related to 'YEARS_BUILD'
yearsbuild_col <- train_clean %>%
  select(starts_with("YEARS_BUILD"))
# Calculate the correlation matrix for these columns
cor_yearsbuild <- cor(yearsbuild_col, use = "complete.obs")

# Select only the columns related to 'ELEVATORS'
elevators_col <- train_clean %>%
  select(starts_with("ELEVATORS"))
# Calculate the correlation matrix for these columns
cor_elevators <- cor(elevators_col, use = "complete.obs")

# Select only the columns related to 'APARTMENTS'
apartments_col <- train_clean %>%
  select(starts_with("APARTMENTS"))
# Calculate the correlation matrix for these columns
cor_apartments <- cor(apartments_col, use = "complete.obs")

# Select only the columns related to 'ENTRANCES'
entrances_col <- train_clean %>%
  select(starts_with("ENTRANCES"))
# Calculate the correlation matrix for these columns
cor_entrances <- cor(entrances_col, use = "complete.obs")

# Select only the columns related to 'ENTRANCES'
Living_area_col <- train_clean %>%
  select(starts_with("LIVINGAREA"))
# Calculate the correlation matrix for these columns
cor_Living_Area <- cor(Living_area_col, use = "complete.obs")

# Print the results
print("Correlation for COMMONAREA variables:")
```

```
[1] "Correlation for COMMONAREA variables:"
```

```
print(cor_commonarea)
```

```
                COMMONAREA_AVG COMMONAREA_MODE COMMONAREA_MEDI
COMMONAREA_AVG       1.0000000       0.9771471       0.9959781
COMMONAREA_MODE      0.9771471       1.0000000       0.9798866
COMMONAREA_MEDI      0.9959781       0.9798866       1.0000000
```

```r
print("Correlation for NONLIVINGAPARTMENTS variables:")
```

```
[1] "Correlation for NONLIVINGAPARTMENTS variables:"
```

```r
print(cor_nonliving)
```

```
                         NONLIVINGAPARTMENTS_AVG NONLIVINGAPARTMENTS_MODE
NONLIVINGAPARTMENTS_AVG                1.0000000                0.9693698
NONLIVINGAPARTMENTS_MODE               0.9693698                1.0000000
NONLIVINGAPARTMENTS_MEDI               0.9907679                0.9785746
                         NONLIVINGAPARTMENTS_MEDI
NONLIVINGAPARTMENTS_AVG                 0.9907679
NONLIVINGAPARTMENTS_MODE                0.9785746
NONLIVINGAPARTMENTS_MEDI                1.0000000
```

```r
print("Correlation for LIVINGAPARTMENTS variables:")
```

```
[1] "Correlation for LIVINGAPARTMENTS variables:"
```

```r
print(cor_living)
```

```
                      LIVINGAPARTMENTS_AVG LIVINGAPARTMENTS_MODE
LIVINGAPARTMENTS_AVG             1.0000000             0.9701167
LIVINGAPARTMENTS_MODE            0.9701167             1.0000000
LIVINGAPARTMENTS_MEDI            0.9938255             0.9756053
                      LIVINGAPARTMENTS_MEDI
LIVINGAPARTMENTS_AVG              0.9938255
LIVINGAPARTMENTS_MODE             0.9756053
LIVINGAPARTMENTS_MEDI             1.0000000
```

```r
print("Correlation for FLOORSMIN variables:")
```

```
[1] "Correlation for FLOORSMIN variables:"
```

```r
print(cor_floorsmin)
```

```
               FLOORSMIN_AVG FLOORSMIN_MODE FLOORSMIN_MEDI
FLOORSMIN_AVG      1.0000000      0.9858751      0.9972410
FLOORSMIN_MODE     0.9858751      1.0000000      0.9884056
FLOORSMIN_MEDI     0.9972410      0.9884056      1.0000000
```

```r
print("Correlation for YEARS_BUILD variables:")
```

```
[1] "Correlation for YEARS_BUILD variables:"
```

```r
print(cor_yearsbuild)
```

```
                 YEARS_BUILD_AVG YEARS_BUILD_MODE YEARS_BUILD_MEDI
YEARS_BUILD_AVG        1.0000000        0.9894439        0.9984947
YEARS_BUILD_MODE       0.9894439        1.0000000        0.9894625
YEARS_BUILD_MEDI       0.9984947        0.9894625        1.0000000
```

```
print("Correlation for ELEVATORS variables:")
```

```
[1] "Correlation for ELEVATORS variables:"
```

```
print(cor_elevators)
```

```
               ELEVATORS_AVG ELEVATORS_MODE ELEVATORS_MEDI
ELEVATORS_AVG      1.0000000      0.9788373      0.9960994
ELEVATORS_MODE     0.9788373      1.0000000      0.9828279
ELEVATORS_MEDI     0.9960994      0.9828279      1.0000000
```

```
print("Correlation for APARTMENTS variables:")
```

```
[1] "Correlation for APARTMENTS variables:"
```

```
print(cor_apartments)
```

```
                APARTMENTS_AVG APARTMENTS_MODE APARTMENTS_MEDI
APARTMENTS_AVG       1.0000000       0.9732595       0.9950808
APARTMENTS_MODE      0.9732595       1.0000000       0.9771931
APARTMENTS_MEDI      0.9950808       0.9771931       1.0000000
```

```
print("Correlation for ENTRANCES variables:")
```

```
[1] "Correlation for ENTRANCES variables:"
```

```
print(cor_entrances)
```

```
               ENTRANCES_AVG ENTRANCES_MODE ENTRANCES_MEDI
ENTRANCES_AVG      1.0000000      0.9777426      0.9968864
ENTRANCES_MODE     0.9777426      1.0000000      0.9806772
ENTRANCES_MEDI     0.9968864      0.9806772      1.0000000
```

```
print("Correlation for LIVINGAREA variables:")
```

```
[1] "Correlation for LIVINGAREA variables:"
```

```
print(cor_Living_Area)
```

```
                LIVINGAREA_AVG LIVINGAREA_MODE LIVINGAREA_MEDI
LIVINGAREA_AVG       1.0000000       0.9720498       0.9955959
```

```
LIVINGAREA_MODE      0.9720498       1.0000000       0.9747434
LIVINGAREA_MEDI      0.9955959       0.9747434       1.0000000
```

The following variables are all very highly correlated with each other and are missing approximately 70% of their column values.

Additionally, only one of the variables is required since the variables are measuring different aspects of the same attribute as previously mentioned (i.e median of common area, mean of common area, etc.) Thus, only of one of the columns is required which will be the median. The median is a more reliable estimate compared to the other 2 metrics.

Moreover for the mode, it's very unlikely that someone will have a multiple common areas, living areas, etc. Thus, the mode can also be removed and it's also highly correlated with the median columns which means that very little information will be lost. It'll also help with creating a more parsimonious model and reducing noise.

Impute Missing Values with MICE

The mice package will be utilized to impute the missing values

```
# Create a vector of columns to be imputed
columns_to_impute <- c("COMMONAREA_MEDI", "LIVINGAPARTMENTS_MEDI",
"NONLIVINGAPARTMENTS_MEDI",
"FLOORSMIN_MEDI",
"YEARS_BUILD_MEDI",
"ELEVATORS_MEDI",
"APARTMENTS_MEDI"    ,
"ENTRANCES_MEDI",
"LIVINGAREA_MEDI")

# Create variables that contain similar columns
common_area <- c('COMMONAREA_AVG','COMMONAREA_MODE','COMMONAREA_MEDI')
living_apartments <- c("LIVINGAPARTMENTS_AVG", "LIVINGAPARTMENTS_MODE", "LIVINGAPARTMENTS_MEDI")
non_living_apartments <- c("NONLIVINGAPARTMENTS_AVG", "NONLIVINGAPARTMENTS_MODE",
floorsmin <- c("FLOORSMIN_AVG", "FLOORSMIN_MODE", "FLOORSMIN_MEDI")
years_build <- c("YEARS_BUILD_AVG", "YEARS_BUILD_MODE", "YEARS_BUILD_MEDI")
elevators <- c("ELEVATORS_AVG", "ELEVATORS_MODE", "ELEVATORS_MEDI")
apartments <- c("APARTMENTS_AVG" ,"APARTMENTS_MODE", "APARTMENTS_MEDI")
entrances <- c("ENTRANCES_AVG" ,"ENTRANCES_MODE", "ENTRANCES_MEDI")
living_area <- c("LIVINGAREA_AVG", "LIVINGAREA_MODE", "LIVINGAREA_MEDI")


# Create a dataset with only the relevant columns
common_area_subset <- train_clean[, common_area]
# Use mice to impute missing values
common_area_subset_imputed <- mice(common_area_subset, m = 1, method = 'pmm', maxit = 5, seed = 1
```

```
 iter imp variable
  1   1  COMMONAREA_AVG  COMMONAREA_MODE  COMMONAREA_MEDI
```

```
 2   1  COMMONAREA_AVG  COMMONAREA_MODE  COMMONAREA_MEDI
 3   1  COMMONAREA_AVG  COMMONAREA_MODE  COMMONAREA_MEDI
 4   1  COMMONAREA_AVG  COMMONAREA_MODE  COMMONAREA_MEDI
 5   1  COMMONAREA_AVG  COMMONAREA_MODE  COMMONAREA_MEDI
```

```r
# Get the completed dataset
finished_data_common_area_subset <- complete(common_area_subset_imputed)


# Create a dataset with only the relevant columns
living_apartments_subset <- train_clean[, living_apartments]
# Use mice to impute missing values
living_apartments_imputed <- mice(living_apartments_subset, m = 1, method = 'pmm', maxit = 5, see
```

```
iter imp variable
 1   1  LIVINGAPARTMENTS_AVG  LIVINGAPARTMENTS_MODE  LIVINGAPARTMENTS_MEDI
 2   1  LIVINGAPARTMENTS_AVG  LIVINGAPARTMENTS_MODE  LIVINGAPARTMENTS_MEDI
 3   1  LIVINGAPARTMENTS_AVG  LIVINGAPARTMENTS_MODE  LIVINGAPARTMENTS_MEDI
 4   1  LIVINGAPARTMENTS_AVG  LIVINGAPARTMENTS_MODE  LIVINGAPARTMENTS_MEDI
 5   1  LIVINGAPARTMENTS_AVG  LIVINGAPARTMENTS_MODE  LIVINGAPARTMENTS_MEDI
```

```r
# Get the completed dataset
finished_data_living_apartments_subset <- complete(living_apartments_imputed)


# Create a dataset with only the relevant columns
non_living_apartments_subset <- train_clean[, non_living_apartments]
# Use mice to impute missing values
non_living_apartments_imputed <- mice(non_living_apartments_subset, m = 1, method = 'pmm', maxit 
```

```
iter imp variable
 1   1  NONLIVINGAPARTMENTS_AVG  NONLIVINGAPARTMENTS_MODE  NONLIVINGAPARTMENTS_MEDI
 2   1  NONLIVINGAPARTMENTS_AVG  NONLIVINGAPARTMENTS_MODE  NONLIVINGAPARTMENTS_MEDI
 3   1  NONLIVINGAPARTMENTS_AVG  NONLIVINGAPARTMENTS_MODE  NONLIVINGAPARTMENTS_MEDI
 4   1  NONLIVINGAPARTMENTS_AVG  NONLIVINGAPARTMENTS_MODE  NONLIVINGAPARTMENTS_MEDI
 5   1  NONLIVINGAPARTMENTS_AVG  NONLIVINGAPARTMENTS_MODE  NONLIVINGAPARTMENTS_MEDI
```

```r
# Get the completed dataset
finished_data_non_living_apartments <- complete(non_living_apartments_imputed)


# Create a dataset with only the relevant columns
floorsmin_subset <- train_clean[, floorsmin]
```

```r
# Use mice to impute missing values
floorsmin_subset_imputed <- mice(floorsmin_subset, m = 1, method = 'pmm', maxit = 5, seed = 123)
```

```
 iter imp variable
  1   1  FLOORSMIN_AVG  FLOORSMIN_MODE  FLOORSMIN_MEDI
  2   1  FLOORSMIN_AVG  FLOORSMIN_MODE  FLOORSMIN_MEDI
  3   1  FLOORSMIN_AVG  FLOORSMIN_MODE  FLOORSMIN_MEDI
  4   1  FLOORSMIN_AVG  FLOORSMIN_MODE  FLOORSMIN_MEDI
  5   1  FLOORSMIN_AVG  FLOORSMIN_MODE  FLOORSMIN_MEDI
```

```r
# Get the completed dataset
finished_data_floorsmin_subset <- complete(floorsmin_subset_imputed)


# Create a dataset with only the relevant columns
years_build_subset <- train_clean[, years_build]
# Use mice to impute missing values
years_build_imputed <- mice(years_build_subset, m = 1, method = 'pmm', maxit = 5, seed = 123)
```

```
 iter imp variable
  1   1  YEARS_BUILD_AVG  YEARS_BUILD_MODE  YEARS_BUILD_MEDI
  2   1  YEARS_BUILD_AVG  YEARS_BUILD_MODE  YEARS_BUILD_MEDI
  3   1  YEARS_BUILD_AVG  YEARS_BUILD_MODE  YEARS_BUILD_MEDI
  4   1  YEARS_BUILD_AVG  YEARS_BUILD_MODE  YEARS_BUILD_MEDI
  5   1  YEARS_BUILD_AVG  YEARS_BUILD_MODE  YEARS_BUILD_MEDI
```

```r
# Get the completed dataset
finished_data_years_build_subset <- complete(years_build_imputed)


# Create a dataset with only the relevant columns
elevators_subset <- train_clean[, elevators]
# Use mice to impute missing values
elevators_imputed <- mice(elevators_subset, m = 1, method = 'pmm', maxit = 5, seed = 123)
```

```
 iter imp variable
  1   1  ELEVATORS_AVG  ELEVATORS_MODE  ELEVATORS_MEDI
  2   1  ELEVATORS_AVG  ELEVATORS_MODE  ELEVATORS_MEDI
  3   1  ELEVATORS_AVG  ELEVATORS_MODE  ELEVATORS_MEDI
  4   1  ELEVATORS_AVG  ELEVATORS_MODE  ELEVATORS_MEDI
  5   1  ELEVATORS_AVG  ELEVATORS_MODE  ELEVATORS_MEDI
```

```r
# Get the completed dataset
finished_data_elevators_subset <- complete(elevators_imputed)
```

```r
# Create a dataset with only the relevant columns
apartments_subset <- train_clean[, apartments]
# Use mice to impute missing values
apartments_imputed <- mice(apartments_subset, m = 1, method = 'pmm', maxit = 5, seed = 123)
```

```
 iter imp variable
  1   1  APARTMENTS_AVG  APARTMENTS_MODE  APARTMENTS_MEDI
  2   1  APARTMENTS_AVG  APARTMENTS_MODE  APARTMENTS_MEDI
  3   1  APARTMENTS_AVG  APARTMENTS_MODE  APARTMENTS_MEDI
  4   1  APARTMENTS_AVG  APARTMENTS_MODE  APARTMENTS_MEDI
  5   1  APARTMENTS_AVG  APARTMENTS_MODE  APARTMENTS_MEDI
```

```r
# Get the completed dataset
finished_data_apartments_subset <- complete(apartments_imputed)


# Create a dataset with only the relevant columns
entrances_subset <- train_clean[, entrances]
# Use mice to impute missing values
entrances_imputed <- mice(entrances_subset, m = 1, method = 'pmm', maxit = 5, seed = 123)
```

```
 iter imp variable
  1   1  ENTRANCES_AVG  ENTRANCES_MODE  ENTRANCES_MEDI
  2   1  ENTRANCES_AVG  ENTRANCES_MODE  ENTRANCES_MEDI
  3   1  ENTRANCES_AVG  ENTRANCES_MODE  ENTRANCES_MEDI
  4   1  ENTRANCES_AVG  ENTRANCES_MODE  ENTRANCES_MEDI
  5   1  ENTRANCES_AVG  ENTRANCES_MODE  ENTRANCES_MEDI
```

```r
# Get the completed dataset
finished_data_entrances_subset <- complete(entrances_imputed)


# Create a dataset with only the relevant columns
living_area_subset <- train_clean[, living_area]
# Use mice to impute missing values
living_area_imputed <- mice(living_area_subset, m = 1, method = 'pmm', maxit = 5, seed = 123)
```

```
 iter imp variable
  1   1  LIVINGAREA_AVG  LIVINGAREA_MODE  LIVINGAREA_MEDI
  2   1  LIVINGAREA_AVG  LIVINGAREA_MODE  LIVINGAREA_MEDI
  3   1  LIVINGAREA_AVG  LIVINGAREA_MODE  LIVINGAREA_MEDI
  4   1  LIVINGAREA_AVG  LIVINGAREA_MODE  LIVINGAREA_MEDI
  5   1  LIVINGAREA_AVG  LIVINGAREA_MODE  LIVINGAREA_MEDI
```

```r
# Get the completed dataset
finished_data_living_area_subset <- complete(living_area_imputed)
```

```
# Replace the column with the imputed column values from the MICE Package
train_clean$COMMONAREA_MEDI <- finished_data_common_area_subset$COMMONAREA_MEDI
train_clean$LIVINGAPARTMENTS_MEDI <- finished_data_living_apartments_subset$LIVINGAPARTMENTS_MEDI
train_clean$NONLIVINGAPARTMENTS_MEDI <- finished_data_non_living_apartments$NONLIVINGAPARTMENTS_MI
train_clean$FLOORSMIN_MEDI <- finished_data_floorsmin_subset$FLOORSMIN_MEDI
train_clean$YEARS_BUILD_MEDI <- finished_data_years_build_subset$YEARS_BUILD_MEDI
train_clean$ELEVATORS_MEDI <- finished_data_elevators_subset$ELEVATORS_MEDI
train_clean$APARTMENTS_MEDI <- finished_data_apartments_subset$APARTMENTS_MEDI
train_clean$ENTRANCES_MEDI <- finished_data_entrances_subset$ENTRANCES_MEDI
train_clean$LIVINGAREA_MEDI <- finished_data_living_area_subset$LIVINGAREA_MEDI
```

The MICE function from the MICE library has been implemented to combine information from the correlated predictors and impute the missing values for the median columns.

It does this by iteratively modelling the column values by using the selected variables as predictors. For instance, to impute the median of the “Common Area” variable, the MICE package will utilize the `COMMONAREA_AVG` and `COMMONAREA_MODE` variables to impute missing values for the `COMMONAREA_MEDI`. It will do this by creating a predictive model utilizing the `COMMONAREA_AVG` and `COMMONAREA_MODE` predictors to make predictions about the missing values.

Since these columns are highly correlated, the MICE algorithm will be able to utilize these values to come up with plausible values for the `COMMONAREA_MEDI` column.

The method utilized is “pmm” which is Predictive Mean Matching. (PMM) This is utilized for continuous data to ensure that imputed values are plausible. (This is done by selecting an actual variable from a similar case which would be `COMMONAREA_AVG` and `COMMONAREA_MODE` for this scenario.)

This process is then repeated for all of the other variables. This method is more robust then imputation with the median because it can take the other columns into account and their effect on the column being imputed.

Drop the Non-Median Columns

```
train_clean <- train_clean %>%
  select(-ends_with("_AVG")) %>% # Drop any column that ends with "AVG"
  select(-ends_with("_MODE")) # Drop any column that ends with "MODE"
```

The missing values in the `MEDI` columns have been successfully imputed. This means that any column that ended with `AVG` or `MODE` which were our `APARTMENTS_AVG`, `ELEVATORS_AVG`, `ELEVATORS_MODE`, etc. should be dropped. (The above code does this.)

It should be noted that `FONDKAPREMONT_MODE` and `WALLSMATERIAL_MODE` were also dropped as a result. These variables however had no other associated columns and were missing 68.38% and 50.84% values respectively. A lot of information will not be lost if these columns are dropped from the dataset. It’ll also help make any future models more parsimonious.

Remaining Missing Values Greater Than 50%

```
#tc = train_clean
missing_values_tc1 <- map(.x = train_clean, .f = find_na) %>%
  unlist() %>%
  data.frame() #Assign dataframe to a new variable called missing values tc1

missing_values_tc1 <- missing_values_tc1 %>%
  rownames_to_column(var = "Column") %>%
  rename(Missing_Values = 1) # Rename the unnamed column to "Missing_Values"

missing_values_tc1 %>%
  filter( . != 0) %>% # Filter to only include rows that are not zero
  summarise(missing_col_cnt  = length(.), # See which columns are missing values
            missing_col_pctg = (length(.)/122) * 100) %>% round(2) # Convert this to a pctg.
```

```
  missing_col_cnt missing_col_pctg
1              19            15.57
```

```
missing_values_tc1 %>%
  filter( . != 0) %>% # Filter to only include rows that are not zero
  mutate(col_pctg_missing_values = ./nrow(train_clean)) %>% # Get this a percentage
  filter(col_pctg_missing_values >=.50) %>% #Filter to only include rows greater than zero
  select(Missing_Values,col_pctg_missing_values) %>% # Select relevant columns
  arrange(desc(col_pctg_missing_values)) # Arrange in descending order
```

```
  Missing_Values col_pctg_missing_values
1    OWN_CAR_AGE               0.6599081
2   EXT_SOURCE_1               0.5638107
```

The only two variables left that are greater than 50% are OWN_CAR_AGE and EXT_SOURCE_1.

Own Car Age Missing Values

```
train_clean %>%
  select(FLAG_OWN_CAR,OWN_CAR_AGE) %>% # Select the following two variables
  filter(FLAG_OWN_CAR == "N") %>% # Only select rows where client marked "N".
  head() # Display the first 6 observations
```

```
# A tibble: 6 × 2
  FLAG_OWN_CAR OWN_CAR_AGE
  <chr>              <dbl>
1 N                     NA
2 N                     NA
3 N                     NA
4 N                     NA
5 N                     NA
6 N                     NA
```

The `OWN_CAR_AGE` variable is closely associated with the `FLAG_OWN_CAR` variable. The `FLAG_OWN_CAR` variable indicates whether someone owns a car or not. This however means that if someone does not own a car they are marked with "N" which should be a zero in the `OWN_CAR_AGE` variable.

This however is not the case. Instead anyone that does not own a car is marked as NA which is inaccurate because they very likely don't own a car. Thus, any NA_Values in the `OWN_CAR_AGE` variable will be imputed as zero.

The above data frame displays this where anybody who marked "no" for owning a car shows an NA value in the `OWN_CAR_AGE` variable. First 6 rows have been showed for demonstration purposes.

```
# Replace missing values in the specific column
train_clean$OWN_CAR_AGE[is.na(train_clean$OWN_CAR_AGE)] <- 0

# Check that the imputation is correct
sum(is.na(train_clean$OWN_CAR_AGE))
```

```
[1] 0
```

Any NA Value in the car data set has now been imputed with zero. The code output also reflect that this was successful, since it shows zero which means that there are no missing values in the NA_Column.

EXT_SOURCE_1 Missing Values

```
# b = bureau
# Define the predictors you want to check for correlation
ext_predictors_to_combine <- c("EXT_SOURCE_1",
"EXT_SOURCE_2", "EXT_SOURCE_3","TARGET")

# Calculate the correlation matrix (ignoring missing values)
ext_predictors_cor_matrix <- cor(train_set_02[ext_predictors_to_combine], use = "complete.obs")

ext_predictors_cor_matrix
```

```
             EXT_SOURCE_1 EXT_SOURCE_2 EXT_SOURCE_3     TARGET
EXT_SOURCE_1    1.0000000    0.2066255    0.1867595 -0.1547572
EXT_SOURCE_2    0.2066255    1.0000000    0.1152402 -0.1448242
EXT_SOURCE_3    0.1867595    0.1152402    1.0000000 -0.1734574
TARGET         -0.1547572   -0.1448242   -0.1734574  1.0000000
```

```
# EXT_Source_1 missing values
round(sum(is.na(train_clean$EXT_SOURCE_1))/nrow(train_clean),2)
```

```
[1] 0.56
```

```
# EXT_Source_2 missing values
round(sum(is.na(train_clean$EXT_SOURCE_2))/nrow(train_clean),4)
```

```
[1] 0.0021
```

```
# EXT_Source_3 missing values
round(sum(is.na(train_clean$EXT_SOURCE_3))/nrow(train_clean),2)
```

```
[1] 0.2
```

It should be noted that `EXT_SOURCE_1` is the variable with 56.38% of its values missing. However, there are 2 other variables that correspond to `EXT_SOURCE_1` which are `EXT_SOURCE_2` and `EXT_SOURCE_3`.

These 3 variables all report the same metric which are the normalized external credit scores.

The above code chunk however shows that these variables are not highly correlated. However with the target variable, all the variables have a somewhat weak negative association.

A possible reason is that the external data sources represent different credit score agencies. The credit agency represented by `EXT_SOURCE_1` may have stricter policies which is why it give less credit scores to clients.

Consequently, this means that the NA's in `EXT_SOURCE_1` could be that particular credit agency refusing to give credit scores. It would also explain why `EXT_SOURCE_1` is missing 56% while `EXT_SOURCE_2` and `EXT_SOURCE_3` are missing far less values. (`EXT_SOURCE_2` $<$ 1%) and (`EXT_SOURCE_3` $<$ 20%).

Moreover, Home Credit's customers are people who cannot utilize traditional finance routes. This is due to having insufficient financial history, etc. This causes some credit-scoring companies like the company represented in `EXT_SOURCE_1` to not give the client's any credit scores.

Thus, for `EXT_SOURCE_1`, any missing values will be imputed with zero.

```
train_clean$EXT_SOURCE_1[is.na(train_set$EXT_SOURCE_1)] <- 0 #Impute all NA's with zero

sum(is.na(train_clean$EXT_SOURCE_1)) # Check that imputation was sucessful. There should be no mi
```

```
[1] 0
```

EXT_SOURCE_2 & 3 Missing Values

Since `EXT_SOURCE_1` has already been addressed, it is logical to also address `EXT_SOURCE_2` and `EXT_SOURCE_3` as well.

`EXT_SOURCE_2` and `3` seem to represent credit card scoring companies who are more lenient with clients. `EXT_SOURCE_2` seems to be very generous since less than 1% of its values are missing.

Thus, the information from EXT_SOURCE_2 and 3 will be used to impute the missing values in both columns.

```
# Create a dataset with only the relevant columns
data_subset <- train_clean[, c("EXT_SOURCE_2", "EXT_SOURCE_3")]
```

```
# Use mice to impute missing values
imputed_data <- mice(data_subset, m = 1, method = 'pmm', maxit = 5, seed = 123)
```

```
 iter imp variable
  1   1  EXT_SOURCE_2  EXT_SOURCE_3
  2   1  EXT_SOURCE_2  EXT_SOURCE_3
  3   1  EXT_SOURCE_2  EXT_SOURCE_3
  4   1  EXT_SOURCE_2  EXT_SOURCE_3
  5   1  EXT_SOURCE_2  EXT_SOURCE_3
```

```
# Get the completed dataset
completed_data <- complete(imputed_data)

# Check that the imputation was sucessful and that there are no missing values
sum(is.na(completed_data$EXT_SOURCE_2))
```

```
[1] 0
```

```
sum(is.na(completed_data$EXT_SOURCE_3))
```

```
[1] 0
```

```
# Add these imputed columns to the dataset
train_clean$EXT_SOURCE_2 <- completed_data$EXT_SOURCE_2
train_clean$EXT_SOURCE_3 <- completed_data$EXT_SOURCE_3

# Check that the columns were added successfully and that there are no missing values.
sum(is.na(train_clean$EXT_SOURCE_2))
```

```
[1] 0
```

```
sum(is.na(train_clean$EXT_SOURCE_3))
```

```
[1] 0
```

The MICE function from the MICE (Multivariate Imputation by Chained Equations) library has been implemented to combine information from both datasets and impute the missing values for both columns.

It does this by iteratively modelling the `EXT_SOURCE_2` and `EXT_SOURCE_3` values by using the selected variables as predictors. (It will use `EXT_SOURCE_2` and `EXT_SOURCE_3 as predictors`)

Then it will create a predictive model utilizing these predictors to make predictions about the missing values.

The method utilized is pmm which is Predictive Mean Matching (PMM) which is utilized for continuous data to ensure that imputed values are plausible. (This is done by selecting an actual variable from a similar case which would be EXT_Source_2 and 3 for this scenario.)

Columns less than 50%

```
#tc = train_clean
missing_values_tc01 <- map(.x = train_clean, .f = find_na) %>% # stores result in a variable
  unlist() %>%
  data.frame() # Assign all the missing values from train_clean to a dataframe

missing_values_tc01 <- missing_values_tc01 %>%
  rownames_to_column(var = "Column") %>%
  rename(Missing_Values = 1) # Rename the unnamed column to "Missing_Values"

missing_values_tc01 %>%
  filter( . != 0) %>% #Filter to include non_zero rows
  mutate(col_pctg_missing_values = ./nrow(train_clean)) %>% #Calculate percentage which are the N
  filter(col_pctg_missing_values <.50) %>% #Filter to only include values less than .50
  select(Missing_Values,col_pctg_missing_values) %>% # Select only relevant columns
  arrange(desc(col_pctg_missing_values)) # Arrange these values in descending order
```

```
                 Missing_Values col_pctg_missing_values
1                FLOORSMAX_MEDI            4.976082e-01
2  YEARS_BEGINEXPLUATATION_MEDI            4.878102e-01
3               OCCUPATION_TYPE            3.134555e-01
4    AMT_REQ_CREDIT_BUREAU_WEEK            1.350163e-01
5     AMT_REQ_CREDIT_BUREAU_QRT            1.350163e-01
6    AMT_REQ_CREDIT_BUREAU_YEAR            1.350163e-01
7               NAME_TYPE_SUITE            4.201476e-03
8      OBS_30_CNT_SOCIAL_CIRCLE            3.320206e-03
9      DEF_30_CNT_SOCIAL_CIRCLE            3.320206e-03
10     OBS_60_CNT_SOCIAL_CIRCLE            3.320206e-03
11     DEF_60_CNT_SOCIAL_CIRCLE            3.320206e-03
12              AMT_GOODS_PRICE            9.040327e-04
13                  AMT_ANNUITY            3.902299e-05
14              CNT_FAM_MEMBERS            6.503832e-06
15       DAYS_LAST_PHONE_CHANGE            3.251916e-06
```

The above code displays which columns are missing less than 50% of their values.

The first 3 columns are still missing several values however:

- `FLOORSMAX_MEDI`
- `YEARS_BEGINEXPLUATATION_MEDI`
- `OCCUPATION_TYPE`

`FLOORSMAX_MEDI` and `YEARS_BEGINEXPLUATATION_MEDI` are very similar to other variables that were missing more than 50% of their values. All of these columns describe aspects of a client's house.

FLOORSMAX_MEDI Missing Values

```
# Get all columns that end with _MEDI
MEDI_COL <- train_clean %>%
  select(ends_with("_MEDI")) %>%
  colnames()

# Define the predictors to check for correlation
floor_combine <- c("FLOORSMAX_MEDI","FLOORSMIN_MEDI")
overall_combine <- c(floor_combine,MEDI_COL)
# Calculate the correlation matrix (ignoring missing values)
fl_correlation_matrix <- cor(train_clean[overall_combine], use = "complete.obs")

# Extract only the correlations for FLOORSMAX_MEDI
fl_max_correlations <- fl_correlation_matrix["FLOORSMAX_MEDI", ]

# Display the correlations
fl_max_correlations
```

```
             FLOORSMAX_MEDI              FLOORSMIN_MEDI
                 1.00000000                  0.49308937
            APARTMENTS_MEDI YEARS_BEGINEXPLUATATION_MEDI
                 0.60225515                  0.12555662
           YEARS_BUILD_MEDI             COMMONAREA_MEDI
                 0.38313310                  0.26007138
             ELEVATORS_MEDI              ENTRANCES_MEDI
                 0.63749643                  0.08710470
             FLOORSMAX_MEDI              FLOORSMIN_MEDI
                 1.00000000                  0.49308937
       LIVINGAPARTMENTS_MEDI             LIVINGAREA_MEDI
                 0.39101233                  0.61045589
    NONLIVINGAPARTMENTS_MEDI
                 0.07177439
```

Most of the following variable's correlations with `Floorsmax` are somewhat strongly associated except for the following:

- `NONLIVINGAPARTMENTS_MEDI`
- `YEARS_BEGINEXPLUATATION_MEDI`
- `ENTRANCES_MEDI`

Thus, these variables will be excluded when performing imputation.

Impute FLOORSMAX_MEDI missing values

```
# Names to exclude
exclude_names <- c("NONLIVINGAPARTMENTS_MEDI", "YEARS_BEGINEXPLUATATION_MEDI", "ENTRANCES_MEDI")

# Extract names of variables from the correlations, excluding the ones in 'exclude_names'
filtered_names <- setdiff(names(fl_max_correlations), exclude_names)
```

```
# Print the filtered names
filtered_names
```

```
[1] "FLOORSMAX_MEDI"        "FLOORSMIN_MEDI"        "APARTMENTS_MEDI"
[4] "YEARS_BUILD_MEDI"      "COMMONAREA_MEDI"       "ELEVATORS_MEDI"
[7] "LIVINGAPARTMENTS_MEDI" "LIVINGAREA_MEDI"
```

```
# Create a dataset with only the relevant columns
fl_data_subset <-train_clean[,filtered_names]

# Use mice to impute missing values
fl_imputed_data <- mice(fl_data_subset, m = 1, method = 'pmm', maxit = 5, seed = 123)
```

```
 iter imp variable
  1   1  FLOORSMAX_MEDI
  2   1  FLOORSMAX_MEDI
  3   1  FLOORSMAX_MEDI
  4   1  FLOORSMAX_MEDI
  5   1  FLOORSMAX_MEDI
```

```
# Get the completed dataset
fl_completed_data <- complete(fl_imputed_data)

# Insert new imputed column back into train dataset.
train_clean$FLOORSMAX_MEDI <- fl_completed_data$FLOORSMAX_MEDI

# Check that imputation was sucessful.
sum(is.na(train_clean$FLOORSMAX_MEDI))
```

```
[1] 0
```

The MICE function from the MICE package can be used again to impute the missing values. MICE will iteratively model the FLOORS_MAX_MEDI value by using the selected variables as predictors.

Then it will create a predictive model utilizing these predictors to make predictions about the missing values.

YEARS_BEGINEXPLUATATION_MEDI

```
# Define the predictors to check for correlation
yr_beg_exp <- c("YEARS_BEGINEXPLUATATION_MEDI")
yr_overall_combine <- c(yr_beg_exp,MEDI_COL)
# Calculate the correlation matrix (ignoring missing values)
yr_correlation_matrix <- cor(train_clean[MEDI_COL], use = "complete.obs")

# Extract only the correlations for FLOORSMAX_MEDI
yr_correlations <- yr_correlation_matrix["YEARS_BEGINEXPLUATATION_MEDI", ]
```

```
# Display yr_correlations
yr_correlations
```

```
             APARTMENTS_MEDI YEARS_BEGINEXPLUATATION_MEDI 
                  0.09774174                   1.00000000 
            YEARS_BUILD_MEDI              COMMONAREA_MEDI 
                  0.11382762                   0.01764346 
              ELEVATORS_MEDI               ENTRANCES_MEDI 
                  0.06818354                   0.04454885 
              FLOORSMAX_MEDI               FLOORSMIN_MEDI 
                  0.09814022                   0.03249784 
       LIVINGAPARTMENTS_MEDI              LIVINGAREA_MEDI 
                  0.03175953                   0.06110845 
    NONLIVINGAPARTMENTS_MEDI 
                  0.01069508 
```

The `YEARS_BEGINEXPLUTATION_MEDI` will be checked with the other `MEDI` columns to see if there is any significant correlation. The `MODE` and `AVG` columns have already been dropped. See the "Drop the Non-Median Columns" Header. Moreover, the `MEDI` columns will be the most similar to the `YEARS_BEGINEXPLUTATION_MEDI` column due to the variables measuring the median of different aspects of the house.

The correlation between this `YEARS_BEGINEXPLUATATION_MEDI` and the other variables is fairly weak. Additionally, this variable is missing a lot of information at 48.78%. Thus, this variable may not contain much information. There are additionally several other columns which describe various aspects of a client's house. Due to these reasons, this variable will be dropped from the dataset.

Drop YEARS_BEGINEXPLUATATION_MEDI

```
#Drop the Year_Beginexpluation variable from the dataframe
train_clean <- train_clean %>%
  select(-YEARS_BEGINEXPLUATATION_MEDI)
```

`YEARS_BEGINEXPLUATATION_MEDI` has been dropped

OCCUPATION_TYPE Missing Values

```
# Show the unique values in the occupation type variable
unique(train_clean$OCCUPATION_TYPE)
```

```
 [1] "Laborers"              "Core staff"            "Accountants"          
 [4] "Managers"              NA                      "Drivers"              
 [7] "Sales staff"           "Cleaning staff"        "Cooking staff"        
[10] "Private service staff" "Medicine staff"        "Security staff"       
[13] "High skill tech staff" "Waiters/barmen staff"  "Low-skill Laborers"   
[16] "Realty agents"         "Secretaries"           "IT staff"             
[19] "HR staff"             
```

The above code displays all the values for `OCCUPATION_TYPE`. NA however appears if the client's occupation is unknown. This however could be inaccurate because Home Credit serves clients who can't get financing through the regular finance system. Thus, NA could instead represent clients who are currently not employed. Hence, NA will be changed to "unemployed" in the dataset.

Convert NA to Unemployed in Occupation Type

```
# Convert any NA Values to Unemployed in Occupation Type
train_clean$OCCUPATION_TYPE[is.na(train_set$OCCUPATION_TYPE)] <- "Unemployed"

# Check to see that this was successful.
sum(is.na(train_clean$OCCUPATION_TYPE))
```

```
[1] 0
```

```
# Convert variable to a factor
train_clean$OCCUPATION_TYPE <- as.factor(train_clean$OCCUPATION_TYPE)
```

The missing values have been successfully imputed with Unemployed and the `OCCUPATION_TYPE` variable has been converted into a factor.

Remaining Missing Values

```
#tc = train_clean
missing_values_tc02 <- map(.x = train_clean, .f = find_na) %>% # Assign all the reamining missing
  unlist() %>%
  data.frame() #Convert this to a dataframe

missing_values_tc02 <- missing_values_tc02 %>% # Override df with new changes
  rownames_to_column(var = "Column") %>%
  rename(Missing_Values = 1) # Rename the unnamed column to "Missing_Values"

missing_values_tc03 <- missing_values_tc02 %>%
  filter( . != 0) %>% #Filter to only include non-zero values
  mutate(col_pctg_missing_values = ./nrow(train_clean)) %>% # Calculate the pctg of missing value
  filter(col_pctg_missing_values <.50) %>% #Filter the values to anything less than .50
  select(Missing_Values,col_pctg_missing_values) %>% # Select relevant columns
  arrange(desc(col_pctg_missing_values)) # Arrange in descending order

missing_values_tc03 # Display the new dataframe
```

```
              Missing_Values col_pctg_missing_values
1 AMT_REQ_CREDIT_BUREAU_WEEK            1.350163e-01
2  AMT_REQ_CREDIT_BUREAU_QRT            1.350163e-01
3 AMT_REQ_CREDIT_BUREAU_YEAR            1.350163e-01
4            NAME_TYPE_SUITE            4.201476e-03
```

```
5     OBS_30_CNT_SOCIAL_CIRCLE            3.320206e-03
6     DEF_30_CNT_SOCIAL_CIRCLE            3.320206e-03
7     OBS_60_CNT_SOCIAL_CIRCLE            3.320206e-03
8     DEF_60_CNT_SOCIAL_CIRCLE            3.320206e-03
9              AMT_GOODS_PRICE            9.040327e-04
10                 AMT_ANNUITY            3.902299e-05
11             CNT_FAM_MEMBERS            6.503832e-06
12      DAYS_LAST_PHONE_CHANGE            3.251916e-06
```

All the other columns are missing less than 1% of their values. This is really small which means that imputation with the median will suffice.

```
# Step 1: Extract column names with missing values from missing_values_tc03
columns_to_impute <- missing_values_tc03$Missing_Values

# Step 2: Perform median imputation on these columns in the main dataset
train_clean[columns_to_impute] <- lapply(train_clean[columns_to_impute], function(x) {
  # Replace missing values with the median of each column
  ifelse(is.na(x), median(x, na.rm = TRUE), x)
})

# Step 3: Check if the missing values are imputed
sum(is.na(train_clean[columns_to_impute]))  # This should return 0 if imputation was successful
```

```
[1] 0
```

All the remaining columns that had less than 1% of the missing data have been imputed with the median.

Check Overall Missing Values

```
#tc = train_clean
missing_values_tc03 <- map(.x = train_clean, .f = find_na) %>%
  unlist() %>%
  data.frame()

# Display the new dataframe which should have zero missing values
missing_values_tc03
```

```
                                .
SK_ID_CURR                      0
TARGET                          0
NAME_CONTRACT_TYPE              0
CODE_GENDER                     0
FLAG_OWN_CAR                    0
FLAG_OWN_REALTY                 0
CNT_CHILDREN                    0
AMT_INCOME_TOTAL                0
AMT_CREDIT                      0
AMT_ANNUITY                     0
```

```
AMT_GOODS_PRICE              0
NAME_TYPE_SUITE              0
NAME_INCOME_TYPE             0
NAME_EDUCATION_TYPE          0
NAME_FAMILY_STATUS           0
NAME_HOUSING_TYPE            0
REGION_POPULATION_RELATIVE   0
DAYS_BIRTH                   0
DAYS_REGISTRATION            0
DAYS_ID_PUBLISH              0
OWN_CAR_AGE                  0
FLAG_EMP_PHONE               0
FLAG_WORK_PHONE              0
FLAG_PHONE                   0
FLAG_EMAIL                   0
OCCUPATION_TYPE              0
CNT_FAM_MEMBERS              0
REGION_RATING_CLIENT         0
REGION_RATING_CLIENT_W_CITY  0
WEEKDAY_APPR_PROCESS_START   0
HOUR_APPR_PROCESS_START      0
REG_REGION_NOT_WORK_REGION   0
REG_CITY_NOT_LIVE_CITY       0
REG_CITY_NOT_WORK_CITY       0
LIVE_CITY_NOT_WORK_CITY      0
ORGANIZATION_TYPE            0
EXT_SOURCE_1                 0
EXT_SOURCE_2                 0
EXT_SOURCE_3                 0
APARTMENTS_MEDI              0
YEARS_BUILD_MEDI             0
COMMONAREA_MEDI              0
ELEVATORS_MEDI               0
ENTRANCES_MEDI               0
FLOORSMAX_MEDI               0
FLOORSMIN_MEDI               0
LIVINGAPARTMENTS_MEDI        0
LIVINGAREA_MEDI              0
NONLIVINGAPARTMENTS_MEDI     0
OBS_30_CNT_SOCIAL_CIRCLE     0
DEF_30_CNT_SOCIAL_CIRCLE     0
OBS_60_CNT_SOCIAL_CIRCLE     0
DEF_60_CNT_SOCIAL_CIRCLE     0
DAYS_LAST_PHONE_CHANGE       0
FLAG_DOCUMENT_3              0
FLAG_DOCUMENT_6              0
FLAG_DOCUMENT_8              0
AMT_REQ_CREDIT_BUREAU_HOUR   0
AMT_REQ_CREDIT_BUREAU_DAY    0
AMT_REQ_CREDIT_BUREAU_WEEK   0
AMT_REQ_CREDIT_BUREAU_MON    0
```

```
AMT_REQ_CREDIT_BUREAU_QRT     0
AMT_REQ_CREDIT_BUREAU_YEAR    0
House_Attribute_Low_Variance 0
```

The cleaning was successful, there are no missing values in any of the columns now.

Outliers

```
# Mutate any character variables into factors
train_clean <- train_clean %>% mutate_if(is.character, as.factor)

# Numeric Variables which should be factors stored as a vector
cat_values <- c("TARGET","FLAG_EMP_PHONE","FLAG_WORK_PHONE","FLAG_EMAIL","FLAG_PHONE",'REG_REGION_

# Utilize mutate to transform all the columns in the cat_values vector to factors.
train_clean <- train_clean %>%
  mutate(across(all_of(cat_values), as.factor))

# Display the summary of train_clean
summary(train_clean)
```

```
   SK_ID_CURR      TARGET           NAME_CONTRACT_TYPE CODE_GENDER  FLAG_OWN_CAR
 Min.   :100002   0:282686   Cash loans     :278232   F   :202448   N:202924
 1st Qu.:189146   1: 24825   Revolving loans: 29279   M   :105059   Y:104587
 Median :278202                                       XNA:      4
 Mean   :278181
 3rd Qu.:367143
 Max.   :456255

 FLAG_OWN_REALTY  CNT_CHILDREN      AMT_INCOME_TOTAL       AMT_CREDIT
 N: 94199        Min.   : 0.0000   Min.   :     25650   Min.   :  45000
 Y:213312        1st Qu.: 0.0000   1st Qu.:    112500   1st Qu.: 270000
                 Median : 0.0000   Median :    147150   Median : 513531
                 Mean   : 0.4171   Mean   :    168798   Mean   : 599026
                 3rd Qu.: 1.0000   3rd Qu.:    202500   3rd Qu.: 808650
                 Max.   :19.0000   Max.   :117000000   Max.   :4050000

  AMT_ANNUITY     AMT_GOODS_PRICE           NAME_TYPE_SUITE
 Min.   :  1616   Min.   :  40500   Children       :  3267
 1st Qu.: 16524   1st Qu.: 238500   Family         : 40149
 Median : 24903   Median : 450000   Group of people:   271
 Mean   : 27108   Mean   : 538316   Other_A        :   866
 3rd Qu.: 34596   3rd Qu.: 679500   Other_B        :  1770
 Max.   :258026   Max.   :4050000   Spouse, partner: 11370
                                    Unaccompanied  :249818
            NAME_INCOME_TYPE                            NAME_EDUCATION_TYPE
 Working             :158774   Academic degree                :    164
 Commercial associate: 71617   Higher education               : 74863
```

```
Pensioner           : 55362   Incomplete higher            : 10277
State servant       : 21703   Lower secondary              :  3816
Unemployed          :    22   Secondary / secondary special:218391
Student             :    18
(Other)             :    15
           NAME_FAMILY_STATUS           NAME_HOUSING_TYPE
Civil marriage      : 29775   Co-op apartment    :  1122
Married             :196432   House / apartment  :272868
Separated           : 19770   Municipal apartment: 11183
Single / not married: 45444   Office apartment   :  2617
Unknown             :     2   Rented apartment   :  4881
Widow               : 16088   With parents       : 14840

REGION_POPULATION_RELATIVE   DAYS_BIRTH     DAYS_REGISTRATION DAYS_ID_PUBLISH
Min.   :0.00029            Min.   :-25229   Min.   :-24672    Min.   :-7197
1st Qu.:0.01001            1st Qu.:-19682   1st Qu.: -7480    1st Qu.:-4299
Median :0.01885            Median :-15750   Median : -4504    Median :-3254
Mean   :0.02087            Mean   :-16037   Mean   : -4986    Mean   :-2994
3rd Qu.:0.02866            3rd Qu.:-12413   3rd Qu.: -2010    3rd Qu.:-1720
Max.   :0.07251            Max.   : -7489   Max.   :     0    Max.   :    0

 OWN_CAR_AGE     FLAG_EMP_PHONE FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL
Min.   : 0.000   0: 55386       0:246203        0:221080   0:290069
1st Qu.: 0.000   1:252125       1: 61308        1: 86431   1: 17442
Median : 0.000
Mean   : 4.102
3rd Qu.: 5.000
Max.   :91.000

   OCCUPATION_TYPE  CNT_FAM_MEMBERS  REGION_RATING_CLIENT
Unemployed :96391   Min.   : 1.000   1: 32197
Laborers   :55186   1st Qu.: 2.000   2:226984
Sales staff:32102   Median : 2.000   3: 48330
Core staff :27570   Mean   : 2.153
Managers   :21371   3rd Qu.: 3.000
Drivers    :18603   Max.   :20.000
(Other)    :56288
REGION_RATING_CLIENT_W_CITY WEEKDAY_APPR_PROCESS_START HOUR_APPR_PROCESS_START
1: 34167                    FRIDAY   :50338            Min.   : 0.00
2:229484                    MONDAY   :50714            1st Qu.:10.00
3: 43860                    SATURDAY :33852            Median :12.00
                            SUNDAY   :16181            Mean   :12.06
                            THURSDAY :50591            3rd Qu.:14.00
                            TUESDAY  :53901            Max.   :23.00
                            WEDNESDAY:51934
REG_REGION_NOT_WORK_REGION REG_CITY_NOT_LIVE_CITY REG_CITY_NOT_WORK_CITY
0:291899                   0:283472               0:236644
1: 15612                   1: 24039               1: 70867
```

```
LIVE_CITY_NOT_WORK_CITY              ORGANIZATION_TYPE   EXT_SOURCE_1
0:252296                Business Entity Type 3: 67992   Min.   :0.0000
1: 55215                XNA                   : 55374   1st Qu.:0.0000
                        Self-employed         : 38412   Median :0.0000
                        Other                 : 16683   Mean   :0.2190
                        Medicine              : 11193   3rd Qu.:0.4563
                        Business Entity Type 2: 10553   Max.   :0.9627
                        (Other)               :107304
 EXT_SOURCE_2        EXT_SOURCE_3       APARTMENTS_MEDI   YEARS_BUILD_MEDI
Min.   :0.0000001   Min.   :0.0005273   Min.   :0.00000   Min.   :0.0000
1st Qu.:0.3924313   1st Qu.:0.3689687   1st Qu.:0.03440   1st Qu.:0.6578
Median :0.5659668   Median :0.5352763   Median :0.07290   Median :0.7048
Mean   :0.5143837   Mean   :0.5096131   Mean   :0.09987   Mean   :0.7182
3rd Qu.:0.6636269   3rd Qu.:0.6674577   3rd Qu.:0.12280   3rd Qu.:0.7920
Max.   :0.8549997   Max.   :0.8960095   Max.   :1.00000   Max.   :1.0000

COMMONAREA_MEDI  ELEVATORS_MEDI    ENTRANCES_MEDI   FLOORSMAX_MEDI
Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.0059   1st Qu.:0.00000   1st Qu.:0.0345   1st Qu.:0.1667
Median :0.0163   Median :0.00000   Median :0.1034   Median :0.1667
Mean   :0.0399   Mean   :0.07417   Mean   :0.1209   Mean   :0.2156
3rd Qu.:0.0461   3rd Qu.:0.12000   3rd Qu.:0.1379   3rd Qu.:0.3333
Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000

FLOORSMIN_MEDI   LIVINGAPARTMENTS_MEDI LIVINGAREA_MEDI
Min.   :0.0000   Min.   :0.00000       Min.   :0.0000
1st Qu.:0.0833   1st Qu.:0.02740       1st Qu.:0.0391
Median :0.2083   Median :0.06070       Median :0.0680
Mean   :0.2220   Mean   :0.08486       Mean   :0.1015
3rd Qu.:0.3750   3rd Qu.:0.10180       3rd Qu.:0.1225
Max.   :1.0000   Max.   :1.00000       Max.   :1.0000

NONLIVINGAPARTMENTS_MEDI OBS_30_CNT_SOCIAL_CIRCLE DEF_30_CNT_SOCIAL_CIRCLE
Min.   :0.000000         Min.   :  0.000          Min.   : 0.0000
1st Qu.:0.000000         1st Qu.:  0.000          1st Qu.: 0.0000
Median :0.000000         Median :  0.000          Median : 0.0000
Mean   :0.006421         Mean   :  1.417          Mean   : 0.1429
3rd Qu.:0.000000         3rd Qu.:  2.000          3rd Qu.: 0.0000
Max.   :1.000000         Max.   :348.000          Max.   :34.0000

OBS_60_CNT_SOCIAL_CIRCLE DEF_60_CNT_SOCIAL_CIRCLE DAYS_LAST_PHONE_CHANGE
Min.   :  0.000          Min.   : 0.00000         Min.   :-4292.0
1st Qu.:  0.000          1st Qu.: 0.00000         1st Qu.:-1570.0
Median :  0.000          Median : 0.00000         Median : -757.0
Mean   :  1.401          Mean   : 0.09972         Mean   : -962.9
3rd Qu.:  2.000          3rd Qu.: 0.00000         3rd Qu.: -274.0
Max.   :344.000          Max.   :24.00000         Max.   :    0.0

FLAG_DOCUMENT_3 FLAG_DOCUMENT_6 FLAG_DOCUMENT_8 AMT_REQ_CREDIT_BUREAU_HOUR
```

```
 0: 89171        0:280433        0:282487        Min.   :0.000000
 1:218340        1: 27078        1: 25024        1st Qu.:0.000000
                                                 Median :0.000000
                                                 Mean   :0.005538
                                                 3rd Qu.:0.000000
                                                 Max.   :4.000000

 AMT_REQ_CREDIT_BUREAU_DAY AMT_REQ_CREDIT_BUREAU_WEEK AMT_REQ_CREDIT_BUREAU_MON
 Min.   :0.000000          Min.   :0.00000            Min.   : 0.0000
 1st Qu.:0.000000          1st Qu.:0.00000            1st Qu.: 0.0000
 Median :0.000000          Median :0.00000            Median : 0.0000
 Mean   :0.006055          Mean   :0.02972            Mean   : 0.2313
 3rd Qu.:0.000000          3rd Qu.:0.00000            3rd Qu.: 0.0000
 Max.   :9.000000          Max.   :8.00000            Max.   :27.0000

 AMT_REQ_CREDIT_BUREAU_QRT AMT_REQ_CREDIT_BUREAU_YEAR
 Min.   :  0.0000          Min.   : 0.000
 1st Qu.:  0.0000          1st Qu.: 1.000
 Median :  0.0000          Median : 1.000
 Mean   :  0.2296          Mean   : 1.778
 3rd Qu.:  0.0000          3rd Qu.: 3.000
 Max.   :261.0000          Max.   :25.000

 House_Attribute_Low_Variance
 Min.   :-2.7716
 1st Qu.:-0.5217
 Median :-0.5217
 Mean   : 0.0000
 3rd Qu.:-0.3499
 Max.   :53.1529
```

Utilize the summary command to see all the variables and see if any potential outliers exist.

Following variables have values that are considered strange. All these columns have values that are excessively large. For instance, `CNT_CHILDREN` has a maximum value of 19 while `OWN_CAR_AGE` has a max value of 91. Each variable and its values will be discussed in its respective section. This is just meant to provide an overview of the variables.

- `CNT_CHILDREN`
- `OWN_CAR_AGE`
- `CNT_FAM_MEMBERS`
- `AMT_INCOME_TOTAL`
- `AMT_CREDIT`
- `AMT_GOODS_PRICE`
- `OBS_60_CNT_SOCIAL_CIRCLE`
- `OBS_30_CNT_SOCIAL_CIRCLE`
- `DEF_60_CNT_SOCIAL_CIRCLE`
- `DEF_30_CNT_SOCIAL_CIRCLE`

Things to Address:

- `AMT_REQ_CREDIT_BUREAU_QRT`
- `AMT_REQ_CREDIT_BUREAU_YEAR`
- `AMT_REQ_CREDIT_BUREAU_MON`
- `DAYS_BIRTH`
- `DAYS_REGISTRATION`
- `DAYS_ID_PUBLISH`
- `Gender: XNA`

Before manually removing the rows, the Local Outlier Factor(LOF) algorithm will be utilized from the Library Dbscan. The LOF algorithm measures the local density deviation of a data point relative to its neighbors(cluster). The number of data points are used to determine a cluster or neighborhood. The number of datapoints itself is controlled by the minPts parameter. (If minPts is set to 3, then the 3 nearest neighbors will be utilized to determine if an observation is an outlier. A higher minPt will search more of the dataset while a lower minPt will do a smaller, more localized search.)

A point is considered an outlier if its density(LOF Score) is significantly lower than its neighbors. This is determined by selecting a threshold score. If the LOF Score is a higher than the threshold score, then the observation is marked as an outlier.

The threshold is an arbitrary number determined by us, but there are some guidelines:

LOF Values Around 1:

- For most data points that are not outliers, the LOF score will be close to 1.
- LOF scores of 1 indicate that the point is in a region of similar density as its neighbors, so it's not considered an outlier.

LOF Values $>$ 1:

- LOF scores greater than 1 indicate potential outliers. The larger the score, the more likely the point is an outlier.
- Common thresholds for identifying outliers are LOF scores between 1.5 to 2 and higher.

Typical Thresholds:

- Threshold between 1.5 and 2: This range often works well for flagging moderate outliers.

This also means that a lower threshold will result in the LOF algorithm classifying more observations as outliers. (Observation's LOF Score must be greater than the outlier.) This also means that a higher threshold will result in the LOF algorithm classifying less observations as outliers.

LOF Alogrithm

```
threshold <- 1.5 #Determined by us
```

```
numeric_columns <- sapply(train_clean, is.numeric) # Get all numeric columns
train_clean_numeric <- train_clean[, numeric_columns] # Use Nuemric Columns to subset train_clean


lof_scores <- lof(train_clean_numeric, minPts = 6) # Set up LOF Algorithm
outliers <- train_clean[lof_scores > threshold, ] # If the lof_score is greater than the threshol

train_clean_filtered <- train_clean[lof_scores <= threshold, ] # Filter train_clean filtered to o

# Get the row indices of the filtered dataset
filtered_indices <- rownames(train_clean_filtered)

# Use these indices to filter the original dataset
train_clean <- train_clean[filtered_indices, ]

# Calculate how many rows have been taken away from the train_clean dataset.
row_reduction <- ((307511 - nrow(train_clean))/307511) * 100

# Round this output to 2
round(row_reduction,2)
```

```
[1] 2.66
```

```
# Check the output for train_clean again via summary()
summary(train_clean)
```

```
   SK_ID_CURR      TARGET          NAME_CONTRACT_TYPE CODE_GENDER   FLAG_OWN_CAR
 Min.   :100002   0:275141   Cash loans     :270835   F   :197104   N:197544
 1st Qu.:186789   1: 24190   Revolving loans: 28496   M   :102223   Y:101787
 Median :273518                                       XNA:      4
 Mean   :273443
 3rd Qu.:360017
 Max.   :446775

 FLAG_OWN_REALTY  CNT_CHILDREN     AMT_INCOME_TOTAL       AMT_CREDIT
 N: 91738        Min.   : 0.0000   Min.   :     25650   Min.   :  45000
 Y:207593        1st Qu.: 0.0000   1st Qu.:    112500   1st Qu.: 270000
                 Median : 0.0000   Median :    146700   Median : 513531
                 Mean   : 0.4169   Mean   :    168839   Mean   : 599157
                 3rd Qu.: 1.0000   3rd Qu.:    202500   3rd Qu.: 808650
                 Max.   :19.0000   Max.   :117000000   Max.   :4050000

  AMT_ANNUITY      AMT_GOODS_PRICE           NAME_TYPE_SUITE
 Min.   :  1616   Min.   :  40500   Children        :  3182
 1st Qu.: 16524   1st Qu.: 238500   Family          : 39086
 Median : 24903   Median : 450000   Group of people:   260
 Mean   : 27112   Mean   : 538429   Other_A         :   845
 3rd Qu.: 34596   3rd Qu.: 679500   Other_B         :  1725
```

```
Max.   :258026   Max.   :4050000   Spouse, partner: 11086
                                   Unaccompanied  :243147
            NAME_INCOME_TYPE                    NAME_EDUCATION_TYPE
Working             :154576   Academic degree              :   160
Commercial associate: 69742   Higher education             : 72887
Pensioner           : 53825   Incomplete higher            : 10012
State servant       : 21135   Lower secondary              :  3709
Unemployed          :    21   Secondary / secondary special:212563
Student             :    17
(Other)             :    15
           NAME_FAMILY_STATUS           NAME_HOUSING_TYPE
Civil marriage      : 28964   Co-op apartment    :  1096
Married             :191263   House / apartment  :265611
Separated           : 19193   Municipal apartment: 10886
Single / not married: 44275   Office apartment   :  2547
Unknown             :     2   Rented apartment   :  4749
Widow               : 15634   With parents       : 14442

REGION_POPULATION_RELATIVE   DAYS_BIRTH     DAYS_REGISTRATION DAYS_ID_PUBLISH
Min.   :0.00029            Min.   :-25229   Min.   :-24672    Min.   :-7197
1st Qu.:0.01001            1st Qu.:-19678   1st Qu.: -7480    1st Qu.:-4299
Median :0.01885            Median :-15748   Median : -4504    Median :-3254
Mean   :0.02087            Mean   :-16035   Mean   : -4987    Mean   :-2994
3rd Qu.:0.02866            3rd Qu.:-12411   3rd Qu.: -2010    3rd Qu.:-1719
Max.   :0.07251            Max.   : -7489   Max.   :     0    Max.   :    0

 OWN_CAR_AGE     FLAG_EMP_PHONE FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL
Min.   : 0.000   0: 53847       0:239640        0:215197   0:282334
1st Qu.: 0.000   1:245484       1: 59691        1: 84134   1: 16997
Median : 0.000
Mean   : 4.099
3rd Qu.: 5.000
Max.   :91.000

   OCCUPATION_TYPE  CNT_FAM_MEMBERS  REGION_RATING_CLIENT
Unemployed :93726   Min.   : 1.000   1: 31347
Laborers   :53759   1st Qu.: 2.000   2:220922
Sales staff:31261   Median : 2.000   3: 47062
Core staff :26863   Mean   : 2.153
Managers   :20797   3rd Qu.: 3.000
Drivers    :18084   Max.   :20.000
(Other)    :54841
REGION_RATING_CLIENT_W_CITY WEEKDAY_APPR_PROCESS_START HOUR_APPR_PROCESS_START
1: 33263                    FRIDAY   :49001            Min.   : 0.00
2:223363                    MONDAY   :49321            1st Qu.:10.00
3: 42705                    SATURDAY :32983            Median :12.00
                            SUNDAY   :15739            Mean   :12.06
                            THURSDAY :49235            3rd Qu.:14.00
                            TUESDAY  :52523            Max.   :23.00
                            WEDNESDAY:50529
REG_REGION_NOT_WORK_REGION REG_CITY_NOT_LIVE_CITY REG_CITY_NOT_WORK_CITY
```

```
0:284142                   0:275921               0:230329
1: 15189                   1: 23410               1: 69002




LIVE_CITY_NOT_WORK_CITY              ORGANIZATION_TYPE   EXT_SOURCE_1
0:245563                Business Entity Type 3: 66202   Min.   :0.0000
1: 53768                XNA                   : 53836   1st Qu.:0.0000
                        Self-employed         : 37380   Median :0.0000
                        Other                 : 16258   Mean   :0.2191
                        Medicine              : 10921   3rd Qu.:0.4565
                        Business Entity Type 2: 10291   Max.   :0.9627
                        (Other)               :104443
 EXT_SOURCE_2        EXT_SOURCE_3       APARTMENTS_MEDI   YEARS_BUILD_MEDI
Min.   :0.0000001   Min.   :0.0005273   Min.   :0.00000   Min.   :0.0000
1st Qu.:0.3926921   1st Qu.:0.3689687   1st Qu.:0.03440   1st Qu.:0.6578
Median :0.5660151   Median :0.5352763   Median :0.07290   Median :0.7048
Mean   :0.5145087   Mean   :0.5096313   Mean   :0.09989   Mean   :0.7182
3rd Qu.:0.6637001   3rd Qu.:0.6674577   3rd Qu.:0.12280   3rd Qu.:0.7920
Max.   :0.8549997   Max.   :0.8960095   Max.   :1.00000   Max.   :1.0000

COMMONAREA_MEDI  ELEVATORS_MEDI    ENTRANCES_MEDI   FLOORSMAX_MEDI
Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.0059   1st Qu.:0.00000   1st Qu.:0.0345   1st Qu.:0.1667
Median :0.0163   Median :0.00000   Median :0.1034   Median :0.1667
Mean   :0.0399   Mean   :0.07414   Mean   :0.1210   Mean   :0.2155
3rd Qu.:0.0461   3rd Qu.:0.12000   3rd Qu.:0.1379   3rd Qu.:0.3333
Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000

FLOORSMIN_MEDI   LIVINGAPARTMENTS_MEDI LIVINGAREA_MEDI
Min.   :0.0000   Min.   :0.00000       Min.   :0.0000
1st Qu.:0.0833   1st Qu.:0.02740       1st Qu.:0.0391
Median :0.2083   Median :0.06070       Median :0.0680
Mean   :0.2220   Mean   :0.08485       Mean   :0.1014
3rd Qu.:0.3750   3rd Qu.:0.10180       3rd Qu.:0.1224
Max.   :1.0000   Max.   :1.00000       Max.   :1.0000

NONLIVINGAPARTMENTS_MEDI OBS_30_CNT_SOCIAL_CIRCLE DEF_30_CNT_SOCIAL_CIRCLE
Min.   :0.000000         Min.   :  0.000          Min.   : 0.0000
1st Qu.:0.000000         1st Qu.:  0.000          1st Qu.: 0.0000
Median :0.000000         Median :  0.000          Median : 0.0000
Mean   :0.006418         Mean   :  1.418          Mean   : 0.1431
3rd Qu.:0.000000         3rd Qu.:  2.000          3rd Qu.: 0.0000
Max.   :1.000000         Max.   :348.000          Max.   :34.0000

OBS_60_CNT_SOCIAL_CIRCLE DEF_60_CNT_SOCIAL_CIRCLE DAYS_LAST_PHONE_CHANGE
Min.   :  0.000          Min.   : 0.00000         Min.   :-4292.0
1st Qu.:  0.000          1st Qu.: 0.00000         1st Qu.:-1570.0
Median :  0.000          Median : 0.00000         Median : -758.0
```

```
 Mean   :  1.401          Mean   : 0.09976         Mean   : -963.1
 3rd Qu.:  2.000          3rd Qu.: 0.00000         3rd Qu.: -274.0
 Max.   :344.000          Max.   :24.00000         Max.   :    0.0

 FLAG_DOCUMENT_3 FLAG_DOCUMENT_6 FLAG_DOCUMENT_8 AMT_REQ_CREDIT_BUREAU_HOUR
 0: 86712        0:273027        0:275019        Min.   :0.000000
 1:212619        1: 26304        1: 24312        1st Qu.:0.000000
                                                 Median :0.000000
                                                 Mean   :0.005549
                                                 3rd Qu.:0.000000
                                                 Max.   :4.000000

 AMT_REQ_CREDIT_BUREAU_DAY AMT_REQ_CREDIT_BUREAU_WEEK AMT_REQ_CREDIT_BUREAU_MON
 Min.   :0.00000           Min.   :0.00000            Min.   : 0.0000
 1st Qu.:0.00000           1st Qu.:0.00000            1st Qu.: 0.0000
 Median :0.00000           Median :0.00000            Median : 0.0000
 Mean   :0.00609           Mean   :0.02971            Mean   : 0.2313
 3rd Qu.:0.00000           3rd Qu.:0.00000            3rd Qu.: 0.0000
 Max.   :9.00000           Max.   :8.00000            Max.   :27.0000

 AMT_REQ_CREDIT_BUREAU_QRT AMT_REQ_CREDIT_BUREAU_YEAR
 Min.   :  0.0000          Min.   : 0.000
 1st Qu.:  0.0000          1st Qu.: 1.000
 Median :  0.0000          Median : 1.000
 Mean   :  0.2297          Mean   : 1.779
 3rd Qu.:  0.0000          3rd Qu.: 3.000
 Max.   :261.0000          Max.   :25.000

 House_Attribute_Low_Variance
 Min.   :-2.77162
 1st Qu.:-0.52169
 Median :-0.52169
 Mean   : 0.00013
 3rd Qu.:-0.34935
 Max.   :53.15290
```

I was relatively conservative with how many rows were eliminated from this dataset since I picked a moderate threshold of 1.5 for determining outliers. The amount of neighbors (minPts) used to determine each algorithm was also relatively small at 6. This consequently resulted in only 2.6% of the rows being eliminated. (The dataset has gone from 307,511 rows to 299,311 rows.)

Several of the extreme values are still present which will require manual cleaning such as `OWN_CAR_AGE` which shows 91 years.

CNT_CHILDREN

`CNT_CHILDREN` has a max value of 19. This is a very unrealistic value. Thus, it'll be filtered.

```
train_clean <- train_clean %>% #Overwrite dataset with the changes
  filter(CNT_CHILDREN <= 7) #7 is a fairly large number that takes into account how many children

summary(train_clean$CNT_CHILDREN) # Call summary to check the rest of hte values and the cnt_chil
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.4163  1.0000  7.0000
```

This has removed some rows. The train_clean data set has gone from 299,331 rows to 299,317 rows.

OWN_CAR_AGE

The oldest car in the dataset is 91 years old which seems very implausible. Thus, this value will be filtered out based on the average age of a car which is 12.43 yrs per this article:

https://www.caranddriver.com/news/a60882953/average-age-us-cars-trucks-suvs-rises/

```
train_clean <- train_clean %>% # OVerride df with new changes
  filter(OWN_CAR_AGE < 12.43) # Filter to only include cars with an average age less than 12.43 y
```

The number of rows has gone from 299,317 rows to 262,483 rows.

CNT_FAM_MEMBERS

Some people in this dataset have been recorded to have 20 family members. This however seems very unlikely, since the median is 2. This also indicates that most clients may have only put their immediate family such as their spouse and family. Thus, the max value will be reduced to 5 to indicate for any extra immediate family members like extra children.

```
train_clean <- train_clean %>% # OVerride df with new changes
  filter(CNT_FAM_MEMBERS < 5) # Filter to only inclue families with less than 5 people
```

The number of rows has decreased from 262,483 to 259,293 rows.

AMT_INCOME_TOTAL

117,000,000 was listed as the maximum income for someone in this dataset. This value however seems very unlikely in the context of this case. Home Credit is lending to people who typically can’t afford conventional financing. A client with an income of 117,000,000 would probably not need to utilize Home Credit’s services. It additionally is very far away from the 3rd quartile which is only 202,500 dollars. Thus, to address this issue, a value relatively close to the 3rd quartile will be selected to filter the data.

```r
train_clean <- train_clean %>% # Override df with new changes
  filter(AMT_INCOME_TOTAL < 300000) # Filter the income to be less than 300k
```

The number of rows has reduced from 259,293 rows to 239,985 rows.

```r
# Count how many values in 'AMT_CREDIT' are greater than or equal to 1,000,000
sum(train_clean$AMT_CREDIT >= 1000000)
```

```
[1] 33624
```

```r
# Count how many values in 'AMT_CREDIT' are less than 1,000,000
sum(train_clean$AMT_CREDIT < 1000000)
```

```
[1] 206361
```

```r
# Count how many values in 'AMT_CREDIT' are greater than or equal to 1,000,000
sum(train_clean$AMT_GOODS_PRICE >= 1000000)
```

```
[1] 22279
```

```r
# Count how many values in 'AMT_CREDIT' are less than 1,000,000
sum(train_clean$AMT_GOODS_PRICE < 1000000)
```

```
[1] 217706
```

Upon further inspection, it has been decided to not remove `AMT_CREDIT` and `AMT_GOODS_PRICE`. This is because `AMT_CREDIT` has values that are greater than 1,000,000 dollars which comprise 14% of the dataset. If these rows are removed, then too much data will be lost. Additionally, the high volume of rows suggests that these are legitimate observations. Moreover, `AMT_GOODS_PRICE` is closely associated with `AMT_CREDIT`, which means that a closer inspection of these two variables will be required.

AMT_CREDIT and AMT_GOODS_PRICE

The maximum value for both of these variable is 4050000 which indicates that some people may have gotten a loan for 400,000 dollars (AMT_CREDIT) and then purchased something for 405,000 dollars. This relationship however seems to have been altered by removing some of the rows from AMT_INCOME_TOTAL.

```r
summary(train_clean$AMT_CREDIT) # Get a summary of just AMT_Credit
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  45000  270000  491031  567455  781880 3860019 
```

```r
summary(train_clean$AMT_GOODS_PRICE) # Get a summary of just AMT_Goods_Price
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  40500  229500  450000  509194  675000 3555000 
```

```
credit_col <- c("AMT_CREDIT", "AMT_GOODS_PRICE") # Combine these two columns into a vector

# Calculate the correlation matrix (ignoring missing values)
correlation_matrix_credit <- cor(train_clean[credit_col], use = "complete.obs")

# Display correlation matrix
correlation_matrix_credit
```

```
                AMT_CREDIT AMT_GOODS_PRICE
AMT_CREDIT       1.0000000       0.9858512
AMT_GOODS_PRICE  0.9858512       1.0000000
```

The maximum value for `AMT_CREDIT` is now 3,860,019 while the goods price is 3,555,000. These values are still relatively high, which may require removal. However too many rows should not be removed. Hence, a more generous threshold will be utilized.

Additionally both of these columns are highly correlated with each other per the correlation matrix. This means that these values are highly associated and removing values from one column will affect the other column. Hence, due to this reason, `AMT_GOODS_PRICE` will also not have any values removed.

OBS_60_CNT_SOCIAL_CIRCLE, DEF_60_CNT_SOCIAL_CIRCLE, OBS_30_CNT_SOCIAL_CIRCLE, DEF_30_CNT_SOCIAL_CIRCLE

These variables shows how many people in the Client's social circle had late payments or defaulted. The maximum value for `OBS_60_CNT_SOCIAL_CIRCLE` is 344.000 which indicates that one client had 344 people in their social circle who made late payments.

```
train_clean %>%
  filter(OBS_60_CNT_SOCIAL_CIRCLE == 344) # Filter to only include the maximum values from OBS_60
```

```
# A tibble: 1 × 64
  SK_ID_CURR TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY
       <dbl> <fct>  <fct>              <fct>       <fct>        <fct>          
1     272071 0      Revolving loans    M           N            Y              
# ℹ 58 more variables: CNT_CHILDREN <dbl>, AMT_INCOME_TOTAL <dbl>,
#   AMT_CREDIT <dbl>, AMT_ANNUITY <dbl>, AMT_GOODS_PRICE <dbl>,
#   NAME_TYPE_SUITE <fct>, NAME_INCOME_TYPE <fct>, NAME_EDUCATION_TYPE <fct>,
#   NAME_FAMILY_STATUS <fct>, NAME_HOUSING_TYPE <fct>,
#   REGION_POPULATION_RELATIVE <dbl>, DAYS_BIRTH <dbl>,
#   DAYS_REGISTRATION <dbl>, DAYS_ID_PUBLISH <dbl>, OWN_CAR_AGE <dbl>,
#   FLAG_EMP_PHONE <fct>, FLAG_WORK_PHONE <fct>, FLAG_PHONE <fct>, …
```

Upon further inspection, it appears that the maximum values for the following variables are also associated with this client:

- DEF_60_CNT_SOCIAL_CIRCLE (24)
- DEF_30_CNT_SOCIAL_CIRCLE (34)
- OBS_30_CNT_SOCIAL_CIRCLE (348)
- OBS_60_CNT_SOCIAL_CIRCLE (344)

These values overall suggest that all of the extreme values are associated with only one client. This means that this client is a very extreme observation. Interestingly, the client was able to pay back their loan. However, regardless of their default status, this client is still an extreme observation that only appears once in the dataset. Thus, this row will be removed which will also remove the extreme values for the other columns as well:

- OBS_30_CNT_SOCIAL_CIRCLE
- DEF_30_CNT_SOCIAL_CIRCLE
- DEF_60_CNT_SOCIAL_CIRCLE

```
train_clean <- train_clean %>%
  filter(OBS_60_CNT_SOCIAL_CIRCLE != 344) # Filter to include all rows that do not equal 344
```

This has removed a row from the train_clean dataset. The overall number of rows is 239,984.

Check OBS_60_CNT_SOCIAL_CIRCLE, DEF_60_CNT_SOCIAL_CIRCLE, OBS_30_CNT_SOCIAL_CIRCLE, DEF_30_CNT_SOCIAL_CIRCLE

```
# Get summaries for just the selected variables by utilizing dollar sign notation.
summary(train_clean$OBS_30_CNT_SOCIAL_CIRCLE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   0.000   0.000   1.432   2.000  47.000 
```

```
summary(train_clean$DEF_60_CNT_SOCIAL_CIRCLE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.1021  0.0000  7.0000 
```

```
summary(train_clean$DEF_30_CNT_SOCIAL_CIRCLE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.1462  0.0000  8.0000 
```

```
summary(train_clean$OBS_60_CNT_SOCIAL_CIRCLE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   0.000   0.000   1.415   2.000  47.000 
```

```
train_clean %>%
  filter(OBS_60_CNT_SOCIAL_CIRCLE == 47,
```

```
        OBS_30_CNT_SOCIAL_CIRCLE == 47) # Filter to only include 47 in both OBS_60 and OBS_30
```

```
# A tibble: 1 × 64
  SK_ID_CURR TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY
       <dbl> <fct>  <fct>              <fct>       <fct>        <fct>
1     189856 0      Cash loans         M           Y            Y
# ℹ 58 more variables: CNT_CHILDREN <dbl>, AMT_INCOME_TOTAL <dbl>,
#   AMT_CREDIT <dbl>, AMT_ANNUITY <dbl>, AMT_GOODS_PRICE <dbl>,
#   NAME_TYPE_SUITE <fct>, NAME_INCOME_TYPE <fct>, NAME_EDUCATION_TYPE <fct>,
#   NAME_FAMILY_STATUS <fct>, NAME_HOUSING_TYPE <fct>,
#   REGION_POPULATION_RELATIVE <dbl>, DAYS_BIRTH <dbl>,
#   DAYS_REGISTRATION <dbl>, DAYS_ID_PUBLISH <dbl>, OWN_CAR_AGE <dbl>,
#   FLAG_EMP_PHONE <fct>, FLAG_WORK_PHONE <fct>, FLAG_PHONE <fct>, …
```

There are still some extreme values in the `OBS_30_CNT_SOCIAL_CIRCLE` and the `OBS_60_CNT_SOCIAL_CIRCLE` variables. This indicates that a stricter threshold is required for removing the remaining outlier values.

```
train_clean %>%
  filter(OBS_30_CNT_SOCIAL_CIRCLE <= 10,
        OBS_60_CNT_SOCIAL_CIRCLE <= 10) # Filter to only include 10 in both OBS_30_CNT and OBS_6
```

```
# A tibble: 237,776 × 64
   SK_ID_CURR TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY
        <dbl> <fct>  <fct>              <fct>       <fct>        <fct>
 1     100002 1      Cash loans         M           N            Y
 2     100003 0      Cash loans         F           N            N
 3     100006 0      Cash loans         F           N            Y
 4     100007 0      Cash loans         M           N            Y
 5     100008 0      Cash loans         M           N            Y
 6     100011 0      Cash loans         F           N            Y
 7     100012 0      Revolving loans    M           N            Y
 8     100014 0      Cash loans         F           N            Y
 9     100015 0      Cash loans         F           N            Y
10     100016 0      Cash loans         F           N            Y
# ℹ 237,766 more rows
# ℹ 58 more variables: CNT_CHILDREN <dbl>, AMT_INCOME_TOTAL <dbl>,
#   AMT_CREDIT <dbl>, AMT_ANNUITY <dbl>, AMT_GOODS_PRICE <dbl>,
#   NAME_TYPE_SUITE <fct>, NAME_INCOME_TYPE <fct>, NAME_EDUCATION_TYPE <fct>,
#   NAME_FAMILY_STATUS <fct>, NAME_HOUSING_TYPE <fct>,
#   REGION_POPULATION_RELATIVE <dbl>, DAYS_BIRTH <dbl>,
#   DAYS_REGISTRATION <dbl>, DAYS_ID_PUBLISH <dbl>, OWN_CAR_AGE <dbl>, …
```

```
train_clean <- train_clean %>% # Same Code but the changes are now being officially piped to the
  filter(OBS_30_CNT_SOCIAL_CIRCLE <= 10,
        OBS_60_CNT_SOCIAL_CIRCLE <= 10)
```

```
# Utilize the summary command to check the distributions for the OBS_30 and OBS_60 Variables.
summary(train_clean$OBS_30_CNT_SOCIAL_CIRCLE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   0.000   0.000   1.322   2.000  10.000 
```

```
summary(train_clean$OBS_60_CNT_SOCIAL_CIRCLE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   0.000   0.000   1.306   2.000  10.000 
```

```
summary(train_clean$DEF_60_CNT_SOCIAL_CIRCLE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.1005  0.0000  6.0000 
```

```
summary(train_clean$DEF_30_CNT_SOCIAL_CIRCLE)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.1432  0.0000  6.0000 
```

This distributions for Client's Social Circle Variable look more reasonable. It also appears that the same extreme observation occurred in `OBS_30` and `OBS_60`. 207 rows were removed due to this procedure which is less than 1% of the train_clean dataset. (The train_clean dataset had 239,983 rows prior to removal and now has 237,776 rows.)

Check Summary Values Again

```
# Check all of the variables
summary(train_clean)
```

```
   SK_ID_CURR      TARGET           NAME_CONTRACT_TYPE CODE_GENDER  FLAG_OWN_CAR
 Min.   :100002   0:218531   Cash loans     :215096   F  :168779   N:184043    
 1st Qu.:186729   1: 19245   Revolving loans: 22680   M  : 68994   Y: 53733    
 Median :273516                                       XNA:     3               
 Mean   :273444                                                                
 3rd Qu.:359980                                                                
 Max.   :446774                                                                

 FLAG_OWN_REALTY  CNT_CHILDREN     AMT_INCOME_TOTAL    AMT_CREDIT     
 N: 72789        Min.   :0.0000   Min.   : 25650   Min.   :  45000  
 Y:164987        1st Qu.:0.0000   1st Qu.:108000   1st Qu.: 270000  
                 Median :0.0000   Median :135000   Median : 491031  
                 Mean   :0.3625   Mean   :148333   Mean   : 567430  
                 3rd Qu.:1.0000   3rd Qu.:180000   3rd Qu.: 781695  
                 Max.   :3.0000   Max.   :299700   Max.   :3860019  
```

```
  AMT_ANNUITY     AMT_GOODS_PRICE          NAME_TYPE_SUITE
 Min.   :  1616   Min.   :  40500   Children       :  2713
 1st Qu.: 16006   1st Qu.: 229500   Family         : 31084
 Median : 23837   Median : 450000   Group of people:   203
 Mean   : 25678   Mean   : 509126   Other_A        :   682
 3rd Qu.: 32603   3rd Qu.: 675000   Other_B        :  1408
 Max.   :225000   Max.   :3555000   Spouse, partner:  8462
                                    Unaccompanied  :193224
            NAME_INCOME_TYPE                    NAME_EDUCATION_TYPE
 Working             :121870   Academic degree              :    99
 Commercial associate: 51983   Higher education             : 53967
 Pensioner           : 47417   Incomplete higher            :  7885
 State servant       : 16464   Lower secondary              :  3062
 Unemployed          :    18   Secondary / secondary special:172763
 Student             :    16
 (Other)             :     8
           NAME_FAMILY_STATUS           NAME_HOUSING_TYPE
 Civil marriage      : 23341   Co-op apartment    :   745
 Married             :147215   House / apartment  :210886
 Separated           : 16124   Municipal apartment:  8903
 Single / not married: 36790   Office apartment   :  1884
 Unknown             :     1   Rented apartment   :  3778
 Widow               : 14305   With parents       : 11580

 REGION_POPULATION_RELATIVE   DAYS_BIRTH     DAYS_REGISTRATION DAYS_ID_PUBLISH
 Min.   :0.00029            Min.   :-25201   Min.   :-24672    Min.   :-7197
 1st Qu.:0.01001            1st Qu.:-19961   1st Qu.: -7649    1st Qu.:-4299
 Median :0.01885            Median :-15968   Median : -4601    Median :-3262
 Mean   :0.02044            Mean   :-16187   Mean   : -5097    Mean   :-2996
 3rd Qu.:0.02639            3rd Qu.:-12439   3rd Qu.: -2108    3rd Qu.:-1721
 Max.   :0.07251            Max.   : -7673   Max.   :     0    Max.   :    0

  OWN_CAR_AGE   FLAG_EMP_PHONE FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL
 Min.   : 0.0   0: 47432       0:190061        0:170246   0:225462
 1st Qu.: 0.0   1:190344       1: 47715        1: 67530   1: 12314
 Median : 0.0
 Mean   : 1.4
 3rd Qu.: 0.0
 Max.   :12.0

    OCCUPATION_TYPE  CNT_FAM_MEMBERS REGION_RATING_CLIENT
 Unemployed :78776   Min.   :1.00    1: 21787
 Laborers   :41576   1st Qu.:2.00    2:179447
 Sales staff:26320   Median :2.00    3: 36542
 Core staff :21645   Mean   :2.08
 Managers   :12915   3rd Qu.:2.00
 Drivers    :11595   Max.   :4.00
 (Other)    :44949
 REGION_RATING_CLIENT_W_CITY WEEKDAY_APPR_PROCESS_START HOUR_APPR_PROCESS_START
 1: 23152                    FRIDAY   :38758            Min.   : 0.00
```

```
2:181331                     MONDAY   :39153          1st Qu.:10.00
3: 33293                     SATURDAY :26248          Median :12.00
                             SUNDAY   :12519          Mean   :12.09
                             THURSDAY :39084          3rd Qu.:14.00
                             TUESDAY  :41640          Max.   :23.00
                             WEDNESDAY:40374
REG_REGION_NOT_WORK_REGION REG_CITY_NOT_LIVE_CITY REG_CITY_NOT_WORK_CITY
0:227280                   0:219288               0:185117
1: 10496                   1: 18488               1: 52659




LIVE_CITY_NOT_WORK_CITY              ORGANIZATION_TYPE  EXT_SOURCE_1
0:197449                Business Entity Type 3:49932   Min.   :0.0000
1: 40327                XNA                   :47426   1st Qu.:0.0000
                        Self-employed         :29110   Median :0.0000
                        Other                 :12695   Mean   :0.2163
                        Medicine              : 9179   3rd Qu.:0.4528
                        Government            : 8079   Max.   :0.9516
                        (Other)               :81355
 EXT_SOURCE_2        EXT_SOURCE_3       APARTMENTS_MEDI   YEARS_BUILD_MEDI
Min.   :0.0000001   Min.   :0.0005273   Min.   :0.00000   Min.   :0.0000
1st Qu.:0.3859750   1st Qu.:0.3706496   1st Qu.:0.03440   1st Qu.:0.6578
Median :0.5627356   Median :0.5388627   Median :0.07290   Median :0.7048
Mean   :0.5111018   Mean   :0.5118501   Mean   :0.09919   Mean   :0.7177
3rd Qu.:0.6612173   3rd Qu.:0.6690567   3rd Qu.:0.12280   3rd Qu.:0.7920
Max.   :0.8549997   Max.   :0.8960095   Max.   :1.00000   Max.   :1.0000

COMMONAREA_MEDI   ELEVATORS_MEDI    ENTRANCES_MEDI   FLOORSMAX_MEDI
Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.00590   1st Qu.:0.00000   1st Qu.:0.0345   1st Qu.:0.1667
Median :0.01620   Median :0.00000   Median :0.1034   Median :0.1667
Mean   :0.03946   Mean   :0.07234   Mean   :0.1213   Mean   :0.2130
3rd Qu.:0.04580   3rd Qu.:0.12000   3rd Qu.:0.1379   3rd Qu.:0.3333
Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000

FLOORSMIN_MEDI   LIVINGAPARTMENTS_MEDI LIVINGAREA_MEDI
Min.   :0.0000   Min.   :0.00000       Min.   :0.0000
1st Qu.:0.0833   1st Qu.:0.02740       1st Qu.:0.0388
Median :0.2083   Median :0.06070       Median :0.0677
Mean   :0.2207   Mean   :0.08435       Mean   :0.1003
3rd Qu.:0.3750   3rd Qu.:0.10180       3rd Qu.:0.1209
Max.   :1.0000   Max.   :1.00000       Max.   :1.0000

NONLIVINGAPARTMENTS_MEDI OBS_30_CNT_SOCIAL_CIRCLE DEF_30_CNT_SOCIAL_CIRCLE
Min.   :0.00000          Min.   : 0.000           Min.   :0.0000
1st Qu.:0.00000          1st Qu.: 0.000           1st Qu.:0.0000
Median :0.00000          Median : 0.000           Median :0.0000
Mean   :0.00639          Mean   : 1.322           Mean   :0.1432
```

```
 3rd Qu.:0.00000          3rd Qu.: 2.000          3rd Qu.:0.0000  
 Max.   :1.00000          Max.   :10.000          Max.   :6.0000  

 OBS_60_CNT_SOCIAL_CIRCLE DEF_60_CNT_SOCIAL_CIRCLE DAYS_LAST_PHONE_CHANGE
 Min.   : 0.000           Min.   :0.0000           Min.   :-4292.0       
 1st Qu.: 0.000           1st Qu.:0.0000           1st Qu.:-1554.0       
 Median : 0.000           Median :0.0000           Median : -741.0       
 Mean   : 1.306           Mean   :0.1005           Mean   : -949.7       
 3rd Qu.: 2.000           3rd Qu.:0.0000           3rd Qu.: -266.0       
 Max.   :10.000           Max.   :6.0000           Max.   :    0.0       

 FLAG_DOCUMENT_3 FLAG_DOCUMENT_6 FLAG_DOCUMENT_8 AMT_REQ_CREDIT_BUREAU_HOUR
 0: 66846        0:214506        0:222941        Min.   :0.000000          
 1:170930        1: 23270        1: 14835        1st Qu.:0.000000          
                                                 Median :0.000000          
                                                 Mean   :0.005451          
                                                 3rd Qu.:0.000000          
                                                 Max.   :4.000000          

 AMT_REQ_CREDIT_BUREAU_DAY AMT_REQ_CREDIT_BUREAU_WEEK AMT_REQ_CREDIT_BUREAU_MON
 Min.   :0.000000          Min.   :0.00000            Min.   : 0.0000          
 1st Qu.:0.000000          1st Qu.:0.00000            1st Qu.: 0.0000          
 Median :0.000000          Median :0.00000            Median : 0.0000          
 Mean   :0.006157          Mean   :0.02947            Mean   : 0.2221          
 3rd Qu.:0.000000          3rd Qu.:0.00000            3rd Qu.: 0.0000          
 Max.   :9.000000          Max.   :8.00000            Max.   :24.0000          

 AMT_REQ_CREDIT_BUREAU_QRT AMT_REQ_CREDIT_BUREAU_YEAR
 Min.   : 0.00             Min.   : 0.00             
 1st Qu.: 0.00             1st Qu.: 1.00             
 Median : 0.00             Median : 1.00             
 Mean   : 0.23             Mean   : 1.78             
 3rd Qu.: 0.00             3rd Qu.: 3.00             
 Max.   :19.00             Max.   :25.00             

 House_Attribute_Low_Variance
 Min.   :-2.77162            
 1st Qu.:-0.52169            
 Median :-0.52169            
 Mean   :-0.00629            
 3rd Qu.:-0.34387            
 Max.   :53.15290            
```

The cleaning has been successful, there are no observable extreme values. Additionally, other variables that may have had some extreme values have also been indirectly addressed such as:

- `AMT_REQ_CREDIT_BUREAU_QRT`, old: 261, new: 19

Although this value is still high, it's much closer to the maximum of the other variables like `AMT_REQ_CREDIT_BUREAU_YEAR` (25) and `AMT_REQ_CREDIT_BUREAU_WEEK` (24). Thus, further removal of outliers does not seem required for the `AMT_CREDIT_BUREAU_QRT` column.

It should be noted that while `BUREAU_QRT` is close to the max of the other BUREAU Columns, the values are still high when compared to the median, 3rd quarter, etc. For instance, the median for `AMT_REQ_CREDIT_BUREAU_YEAR` is 3 inquires while the max is 25 inquires. `AMT_REQ_CREDIT_BUREAU_MON` also displays a similar trend with the max being 24 inquiries while the median is 1 inquiry.

These are however plausible values which could reflect Home Credit's concern about particular clients. In other words those clients may have had concerning characteristics that warranted more inquires then usual. The exception however would the be the old `AMT_REQ_CREDIT_BUREAU_QRT` which was 261 inquiries. This seems like an extreme value which warranted removal.

Other Variable Issues

The following 4 variables have the distribution of their variables reversed:

- DAYS_BIRTH
- DAYS_REGISTRATION
- DAYS_ID_PUBLISH
- DAYS_LAST_PHONE_CHANGE

For instance, the below code for DAYS_BIRTH displays this issue.

```
# Show the summary of Days_Birth.
summary(train_clean$DAYS_BIRTH)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -25201  -19961  -15968  -16187  -12439   -7673
```

```
round(25201/365,2) # Calculate the age of the oldest client in this dataframe.
```

```
[1] 69.04
```

```
round(7673/265,2)  # Calculate the age of the youngest client in this dataframe.
```

```
[1] 28.95
```

Days birth measures how many days it has been since the client was born. This is done by subtracting the birth date measured as 0 from the current client's age in day. So if a client is 69.04 years, their birthdate is calculated as follows:

0-25201 = -2501 days or -69.04 years. This however is an issue if the client is 69 years old, because the summary command has marked the client as the minimum value. The 69 year old client however should be the maximum while the -2501 client who is 29.85 years should be marked as the youngest.

Thus, to fix this issue, the variable distribution will be reordered.

```
# Convert DAYS_BIRTH to positive values with absolute value (to represent age in days)
train_clean$DAYS_BIRTH <- abs(train_clean$DAYS_BIRTH)
summary(train_clean$DAYS_BIRTH) # Show summary of Days_Birth.
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7673   12439   15968   16187   19961   25201
```

The other variables have the same issue:

- DAYS_REGISTRATION
- DAYS_ID_PUBLISH
- DAYS_LAST_PHONE_CHANGE

They all represent how many days it's has been since the client did something prior to the application.

DAYS_REGISTRATION

For instance, DAYS_REGISTRATION measures how many days prior to the application did client change his registration. However the datset has measured 0 days as the max which is incorrect. 24,672 days should be the max which means that the client changed his registration 24,672 days before the application. 0 days would likewise mean that the client did not change his registration prior to the application.

```
# Show summary of just Days_registration
summary(train_clean$DAYS_REGISTRATION)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -24672   -7649   -4601   -5097   -2108       0
```

```
train_clean$DAYS_REGISTRATION <- abs(train_clean$DAYS_REGISTRATION) # Convert values to positive 
summary(train_clean$DAYS_REGISTRATION) # Show updated summary of just Days_Registration
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0    2108    4601    5097    7649   24672
```

Days_Registration has been successfully reordered with 24672 days reflecting the maximum number of days a client changed their registration prior to their loan application.

DAYS_ID_PUBLISH

DAYS_ID_PUBLISH is how many days is how many days before the application did the client change the identity document that he used to apply for the loan. The issue however is that maximum number of days before a client changed their document is 0.

This is incorrect because 0 should be the minimum which means that the minimum number of days before a client changed their document is 0. If minimum is zero though, this means that the client did not change their document at all.

```
summary(train_clean$DAYS_ID_PUBLISH) # Show summary of just DAYS_ID_PUBLISH
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  -7197   -4299   -3262   -2996   -1721       0 
```

```
train_clean$DAYS_ID_PUBLISH <- abs(train_clean$DAYS_ID_PUBLISH) # Convert values to reverse distr
summary(train_clean$DAYS_ID_PUBLISH) #Show updated summary of just DAYS_ID_PUBLISH
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      0    1721    3262    2996    4299    7197 
```

Days_ID_Publish has been successfully reordered with 7197 days reflecting the maximum number of days a client changed the ID Document that they used to apply for the loan.

DAYS_LAST_PHONE_CHANGE

`DAYS_LAST_PHONE_CHANGE` represents how many days before the application did the client change their phone. The problem is that smallest amount of days and the largest amount of days are reversed. For instance, the min is 4,292 which means that the client changed their phone 4,292 days before the application. This however should be the max which would be the maximum number of days before an application. Thus, the column's values need to be reordered.

```
summary(train_clean$DAYS_LAST_PHONE_CHANGE) # Show summary of just DAYS_LAST_PHONE_CHANGE
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-4292.0 -1554.0  -741.0  -949.7  -266.0     0.0 
```

```
train_clean$DAYS_LAST_PHONE_CHANGE <- abs(train_clean$DAYS_LAST_PHONE_CHANGE) #Convert values to 
summary(train_clean$DAYS_LAST_PHONE_CHANGE) #Show updated summary of just DAYS_LAST_PHONE_CHANGE
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    0.0   266.0   741.0   949.7  1554.0  4292.0 
```

The columns values have been successfully reordered with the maximum number of days being 4,292 days and the minimum number of days being 0 days. (This means if someone changed their phone 4,292 days before the application, it will be recorded as the maximum.)

Check Overall Values Again

```
summary(train_clean) # Check all the overall values again to ensure that the observations look rea
```

```
   SK_ID_CURR      TARGET           NAME_CONTRACT_TYPE CODE_GENDER  FLAG_OWN_CAR
Min.   :100002   0:218531   Cash loans     :215096   F  :168779   N:184043
1st Qu.:186729   1: 19245   Revolving loans: 22680   M  : 68994   Y: 53733
Median :273516                                       XNA:     3
Mean   :273444
3rd Qu.:359980
Max.   :446774

FLAG_OWN_REALTY  CNT_CHILDREN     AMT_INCOME_TOTAL    AMT_CREDIT
N: 72789        Min.   :0.0000   Min.   : 25650   Min.   :  45000
Y:164987        1st Qu.:0.0000   1st Qu.:108000   1st Qu.: 270000
                Median :0.0000   Median :135000   Median : 491031
                Mean   :0.3625   Mean   :148333   Mean   : 567430
                3rd Qu.:1.0000   3rd Qu.:180000   3rd Qu.: 781695
                Max.   :3.0000   Max.   :299700   Max.   :3860019

  AMT_ANNUITY      AMT_GOODS_PRICE          NAME_TYPE_SUITE
Min.   :  1616   Min.   :  40500   Children       :  2713
1st Qu.: 16006   1st Qu.: 229500   Family         : 31084
Median : 23837   Median : 450000   Group of people:   203
Mean   : 25678   Mean   : 509126   Other_A        :   682
3rd Qu.: 32603   3rd Qu.: 675000   Other_B        :  1408
Max.   :225000   Max.   :3555000   Spouse, partner:  8462
                                   Unaccompanied  :193224
            NAME_INCOME_TYPE                         NAME_EDUCATION_TYPE
Working             :121870   Academic degree              :    99
Commercial associate: 51983   Higher education             : 53967
Pensioner           : 47417   Incomplete higher            :  7885
State servant       : 16464   Lower secondary              :  3062
Unemployed          :    18   Secondary / secondary special:172763
Student             :    16
(Other)             :     8
           NAME_FAMILY_STATUS            NAME_HOUSING_TYPE
Civil marriage      : 23341   Co-op apartment    :   745
Married             :147215   House / apartment  :210886
Separated           : 16124   Municipal apartment:  8903
Single / not married: 36790   Office apartment   :  1884
Unknown             :     1   Rented apartment   :  3778
Widow               : 14305   With parents       : 11580

REGION_POPULATION_RELATIVE   DAYS_BIRTH     DAYS_REGISTRATION DAYS_ID_PUBLISH
Min.   :0.00029            Min.   : 7673   Min.   :    0     Min.   :   0
1st Qu.:0.01001            1st Qu.:12439   1st Qu.: 2108     1st Qu.:1721
Median :0.01885            Median :15968   Median : 4601     Median :3262
```

```
Mean   :0.02044           Mean   :16187   Mean   : 5097     Mean   :2996
3rd Qu.:0.02639           3rd Qu.:19961   3rd Qu.: 7649     3rd Qu.:4299
Max.   :0.07251           Max.   :25201   Max.   :24672     Max.   :7197

 OWN_CAR_AGE   FLAG_EMP_PHONE FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL
Min.   : 0.0   0: 47432       0:190061        0:170246   0:225462
1st Qu.: 0.0   1:190344       1: 47715        1: 67530   1: 12314
Median : 0.0
Mean   : 1.4
3rd Qu.: 0.0
Max.   :12.0

   OCCUPATION_TYPE  CNT_FAM_MEMBERS REGION_RATING_CLIENT
Unemployed :78776   Min.   :1.00    1: 21787
Laborers   :41576   1st Qu.:2.00    2:179447
Sales staff:26320   Median :2.00    3: 36542
Core staff :21645   Mean   :2.08
Managers   :12915   3rd Qu.:2.00
Drivers    :11595   Max.   :4.00
(Other)    :44949
REGION_RATING_CLIENT_W_CITY WEEKDAY_APPR_PROCESS_START HOUR_APPR_PROCESS_START
1: 23152                    FRIDAY   :38758            Min.   : 0.00
2:181331                    MONDAY   :39153            1st Qu.:10.00
3: 33293                    SATURDAY :26248            Median :12.00
                            SUNDAY   :12519            Mean   :12.09
                            THURSDAY :39084            3rd Qu.:14.00
                            TUESDAY  :41640            Max.   :23.00
                            WEDNESDAY:40374
REG_REGION_NOT_WORK_REGION REG_CITY_NOT_LIVE_CITY REG_CITY_NOT_WORK_CITY
0:227280                   0:219288               0:185117
1: 10496                   1: 18488               1: 52659




LIVE_CITY_NOT_WORK_CITY              ORGANIZATION_TYPE  EXT_SOURCE_1
0:197449                Business Entity Type 3:49932   Min.   :0.0000
1: 40327                XNA                   :47426   1st Qu.:0.0000
                        Self-employed         :29110   Median :0.0000
                        Other                 :12695   Mean   :0.2163
                        Medicine              : 9179   3rd Qu.:0.4528
                        Government            : 8079   Max.   :0.9516
                        (Other)               :81355
 EXT_SOURCE_2        EXT_SOURCE_3       APARTMENTS_MEDI   YEARS_BUILD_MEDI
Min.   :0.0000001   Min.   :0.0005273   Min.   :0.00000   Min.   :0.0000
1st Qu.:0.3859750   1st Qu.:0.3706496   1st Qu.:0.03440   1st Qu.:0.6578
Median :0.5627356   Median :0.5388627   Median :0.07290   Median :0.7048
Mean   :0.5111018   Mean   :0.5118501   Mean   :0.09919   Mean   :0.7177
3rd Qu.:0.6612173   3rd Qu.:0.6690567   3rd Qu.:0.12280   3rd Qu.:0.7920
Max.   :0.8549997   Max.   :0.8960095   Max.   :1.00000   Max.   :1.0000
```

```
COMMONAREA_MEDI   ELEVATORS_MEDI    ENTRANCES_MEDI   FLOORSMAX_MEDI
Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.00590   1st Qu.:0.00000   1st Qu.:0.0345   1st Qu.:0.1667
Median :0.01620   Median :0.00000   Median :0.1034   Median :0.1667
Mean   :0.03946   Mean   :0.07234   Mean   :0.1213   Mean   :0.2130
3rd Qu.:0.04580   3rd Qu.:0.12000   3rd Qu.:0.1379   3rd Qu.:0.3333
Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000

FLOORSMIN_MEDI   LIVINGAPARTMENTS_MEDI LIVINGAREA_MEDI
Min.   :0.0000   Min.   :0.00000       Min.   :0.0000
1st Qu.:0.0833   1st Qu.:0.02740       1st Qu.:0.0388
Median :0.2083   Median :0.06070       Median :0.0677
Mean   :0.2207   Mean   :0.08435       Mean   :0.1003
3rd Qu.:0.3750   3rd Qu.:0.10180       3rd Qu.:0.1209
Max.   :1.0000   Max.   :1.00000       Max.   :1.0000

NONLIVINGAPARTMENTS_MEDI OBS_30_CNT_SOCIAL_CIRCLE DEF_30_CNT_SOCIAL_CIRCLE
Min.   :0.00000          Min.   : 0.000           Min.   :0.0000
1st Qu.:0.00000          1st Qu.: 0.000           1st Qu.:0.0000
Median :0.00000          Median : 0.000           Median :0.0000
Mean   :0.00639          Mean   : 1.322           Mean   :0.1432
3rd Qu.:0.00000          3rd Qu.: 2.000           3rd Qu.:0.0000
Max.   :1.00000          Max.   :10.000           Max.   :6.0000

OBS_60_CNT_SOCIAL_CIRCLE DEF_60_CNT_SOCIAL_CIRCLE DAYS_LAST_PHONE_CHANGE
Min.   : 0.000           Min.   :0.0000           Min.   :   0.0
1st Qu.: 0.000           1st Qu.:0.0000           1st Qu.: 266.0
Median : 0.000           Median :0.0000           Median : 741.0
Mean   : 1.306           Mean   :0.1005           Mean   : 949.7
3rd Qu.: 2.000           3rd Qu.:0.0000           3rd Qu.:1554.0
Max.   :10.000           Max.   :6.0000           Max.   :4292.0

FLAG_DOCUMENT_3 FLAG_DOCUMENT_6 FLAG_DOCUMENT_8 AMT_REQ_CREDIT_BUREAU_HOUR
0: 66846        0:214506        0:222941        Min.   :0.000000
1:170930        1: 23270        1: 14835        1st Qu.:0.000000
                                                Median :0.000000
                                                Mean   :0.005451
                                                3rd Qu.:0.000000
                                                Max.   :4.000000

AMT_REQ_CREDIT_BUREAU_DAY AMT_REQ_CREDIT_BUREAU_WEEK AMT_REQ_CREDIT_BUREAU_MON
Min.   :0.000000          Min.   :0.00000            Min.   : 0.0000
1st Qu.:0.000000          1st Qu.:0.00000            1st Qu.: 0.0000
Median :0.000000          Median :0.00000            Median : 0.0000
Mean   :0.006157          Mean   :0.02947            Mean   : 0.2221
3rd Qu.:0.000000          3rd Qu.:0.00000            3rd Qu.: 0.0000
Max.   :9.000000          Max.   :8.00000            Max.   :24.0000

AMT_REQ_CREDIT_BUREAU_QRT AMT_REQ_CREDIT_BUREAU_YEAR
Min.   : 0.00             Min.   : 0.00
```

```
 1st Qu.: 0.00            1st Qu.: 1.00
 Median : 0.00            Median : 1.00
 Mean   : 0.23            Mean   : 1.78
 3rd Qu.: 0.00            3rd Qu.: 3.00
 Max.   :19.00            Max.   :25.00

 House_Attribute_Low_Variance
 Min.   :-2.77162
 1st Qu.:-0.52169
 Median :-0.52169
 Mean   :-0.00629
 3rd Qu.:-0.34387
 Max.   :53.15290
```

The overall values look good, however there are 2 columns that have the same issue:

- `ORGANIZATION_TYPE`
- `CODE_GENDER`

Both of these values have a category called XNA. "XNA" however means different things in the context of both variables. Since these columns had no missing values (NA), R did not classify these columns as containing missing values. Hence, this must also be fixed.

ORGANIZATION_TYPE

`ORGANIZATION_TYPE` captures which industry the client works in. There is a column called "XNA" however. XNA could represent two things:

- The client genuinely did not provide the industry that they worked in. Thus, XNA represents the missing industry.

- XNA still represents the missing value but the missing value means something different. There are 58 unique levels in the `ORGANIZATION_TYPE` variable, including an "other" category. This means that even if the client's industry was not listed, they could have selected "other". Additionally, there is no "unemployed" category. XNA also represents 20% of the missing values in the `ORGANIZATION_TYPE` which does not seem Missing Completely at Random. Thus, XNA will be converted to "unemployment".

```
unique(train_clean$ORGANIZATION_TYPE) # See the unique values in Organization_Type
```

```
 [1] Business Entity Type 3 School                 Religion
 [4] Other                  XNA                    Electricity
 [7] Medicine               Business Entity Type 2 Transport: type 2
[10] Government             Construction           Housing
[13] Kindergarten           Self-employed          Industry: type 11
[16] Military               Services               Security Ministries
[19] Transport: type 4      Industry: type 1       Security
[22] Trade: type 2          University             Transport: type 3
[25] Police                 Business Entity Type 1 Postal
[28] Trade: type 7          Agriculture            Restaurant
```

```
[31] Culture                Hotel                  Industry: type 7
[34] Trade: type 3          Industry: type 3       Bank
[37] Industry: type 9       Insurance              Trade: type 6
[40] Transport: type 1      Emergency              Industry: type 12
[43] Industry: type 4       Industry: type 2       Mobile
[46] Trade: type 1          Industry: type 5       Industry: type 10
[49] Legal Services         Trade: type 5          Cleaning
[52] Industry: type 13      Trade: type 4          Telecom
[55] Realtor                Advertising            Industry: type 6
[58] Industry: type 8
58 Levels: Advertising Agriculture Bank ... XNA
```

```
round(sum(train_clean$ORGANIZATION_TYPE == 'XNA')/nrow(train_clean),2) * 100 # Calculate what per
```

```
[1] 20
```

```
# Replace the XNA Column with Unemployed
levels(train_clean$ORGANIZATION_TYPE)[levels(train_clean$ORGANIZATION_TYPE) == "XNA"] <- "Unemplo

# Check that this worked sucessfully
levels(train_clean$ORGANIZATION_TYPE)
```

```
 [1] "Advertising"            "Agriculture"            "Bank"
 [4] "Business Entity Type 1" "Business Entity Type 2" "Business Entity Type 3"
 [7] "Cleaning"               "Construction"           "Culture"
[10] "Electricity"            "Emergency"              "Government"
[13] "Hotel"                  "Housing"                "Industry: type 1"
[16] "Industry: type 10"      "Industry: type 11"      "Industry: type 12"
[19] "Industry: type 13"      "Industry: type 2"       "Industry: type 3"
[22] "Industry: type 4"       "Industry: type 5"       "Industry: type 6"
[25] "Industry: type 7"       "Industry: type 8"       "Industry: type 9"
[28] "Insurance"              "Kindergarten"           "Legal Services"
[31] "Medicine"               "Military"               "Mobile"
[34] "Other"                  "Police"                 "Postal"
[37] "Realtor"                "Religion"               "Restaurant"
[40] "School"                 "Security"               "Security Ministries"
[43] "Self-employed"          "Services"               "Telecom"
[46] "Trade: type 1"          "Trade: type 2"          "Trade: type 3"
[49] "Trade: type 4"          "Trade: type 5"          "Trade: type 6"
[52] "Trade: type 7"          "Transport: type 1"      "Transport: type 2"
[55] "Transport: type 3"      "Transport: type 4"      "University"
[58] "Unemployed"
```

The conversion was successful and XNA has now been changed to Unemployed which can be seen from the new levels for the `ORGANIZATION_TYPE` variable.

CODE GENDER

The `Gender` Variable has a valued called XNA. This could represent 2 things:

- XNA are clients who did not provide their gender to Home Credit or Home Credit was unable to obtain their gender through other means. In other words, these are genuinely missing values.
- The client may not be a "female" or "male". They may identify as some other gender. Thus, they decided to put anything for gender since "other" may not have been an option.

```
unique(train_clean$CODE_GENDER) # See the unique values in CODE_GENDER
```

```
[1] M   F   XNA
Levels: F M XNA
```

```
round(sum(train_clean$CODE_GENDER == 'XNA')/nrow(train_clean) * 100,4) # Calculate what percentag
```

```
[1] 0.0013
```

```
# Show the actual count of missing values
sum(train_clean$CODE_GENDER == 'XNA')
```

```
[1] 3
```

Upon closer inspection, XNA represents less than 1% of the values in the Gender_Count Column. (There are only 3 XNA values overall). Thus for simplicity, this column can be imputed with the mode. The XNA values could be converted to "other" but they comprise a small portion of the column. This means that their predictive power will be limited in comparison to the "Male" and "Female" genders.

```
gender_var <- find_mode(as.character(train_clean$CODE_GENDER)) # Use mode function to calculate wl

levels(train_clean$CODE_GENDER)[levels(train_clean$CODE_GENDER) == "XNA"] <- gender_var # Impute 

levels(train_clean$CODE_GENDER) # Check that the conversion worked with levels command
```

```
[1] "F" "M"
```

The conversion was successful, gender only has 2 values, Male and Female.

Visualizations

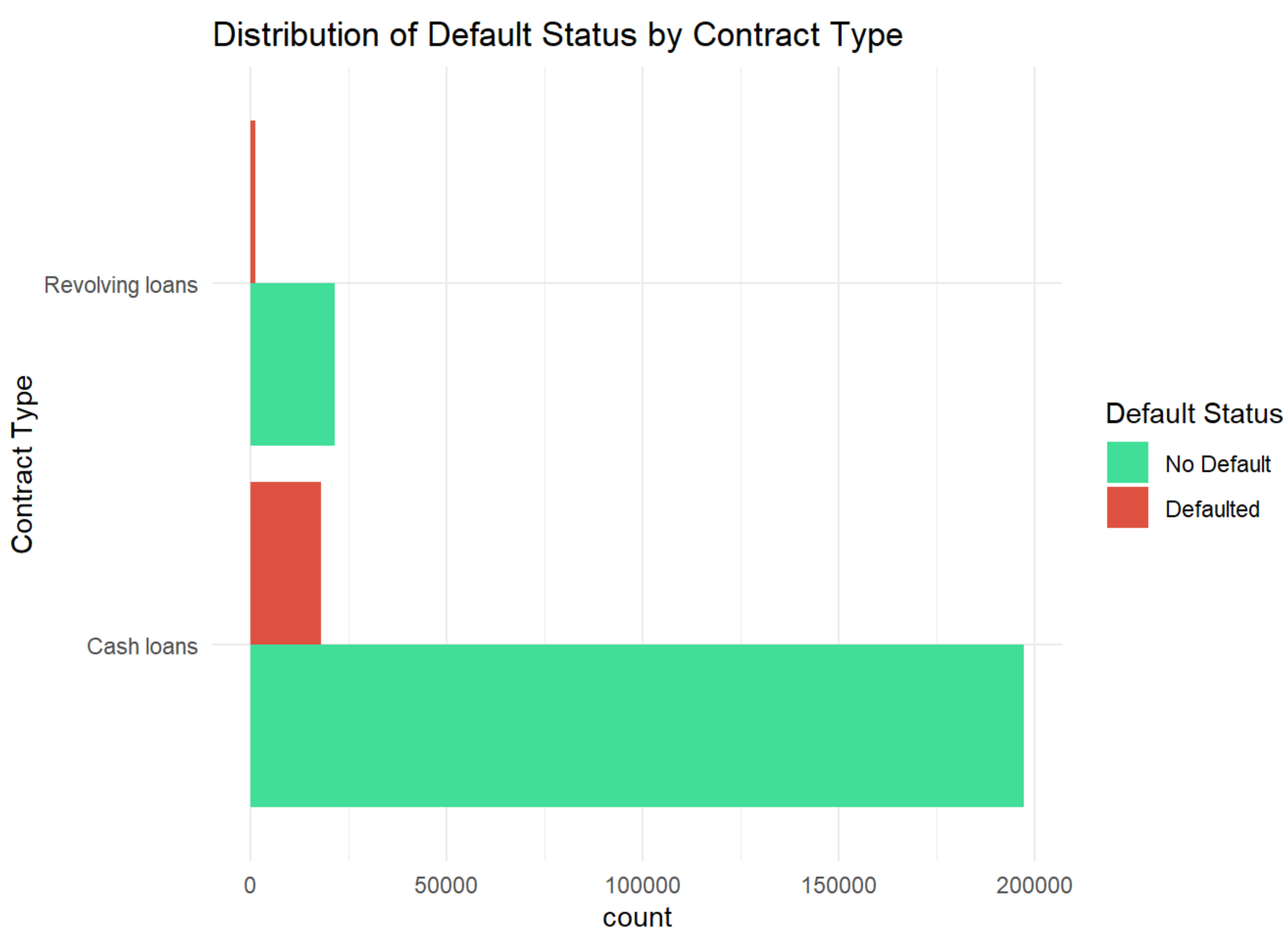
Visualizations have been created to help answer the exploratory quesetions. The visualizations will also help wtih enabling exploration of interesting variables.

Bar Plots

Distribution of Default Status by Contract Type

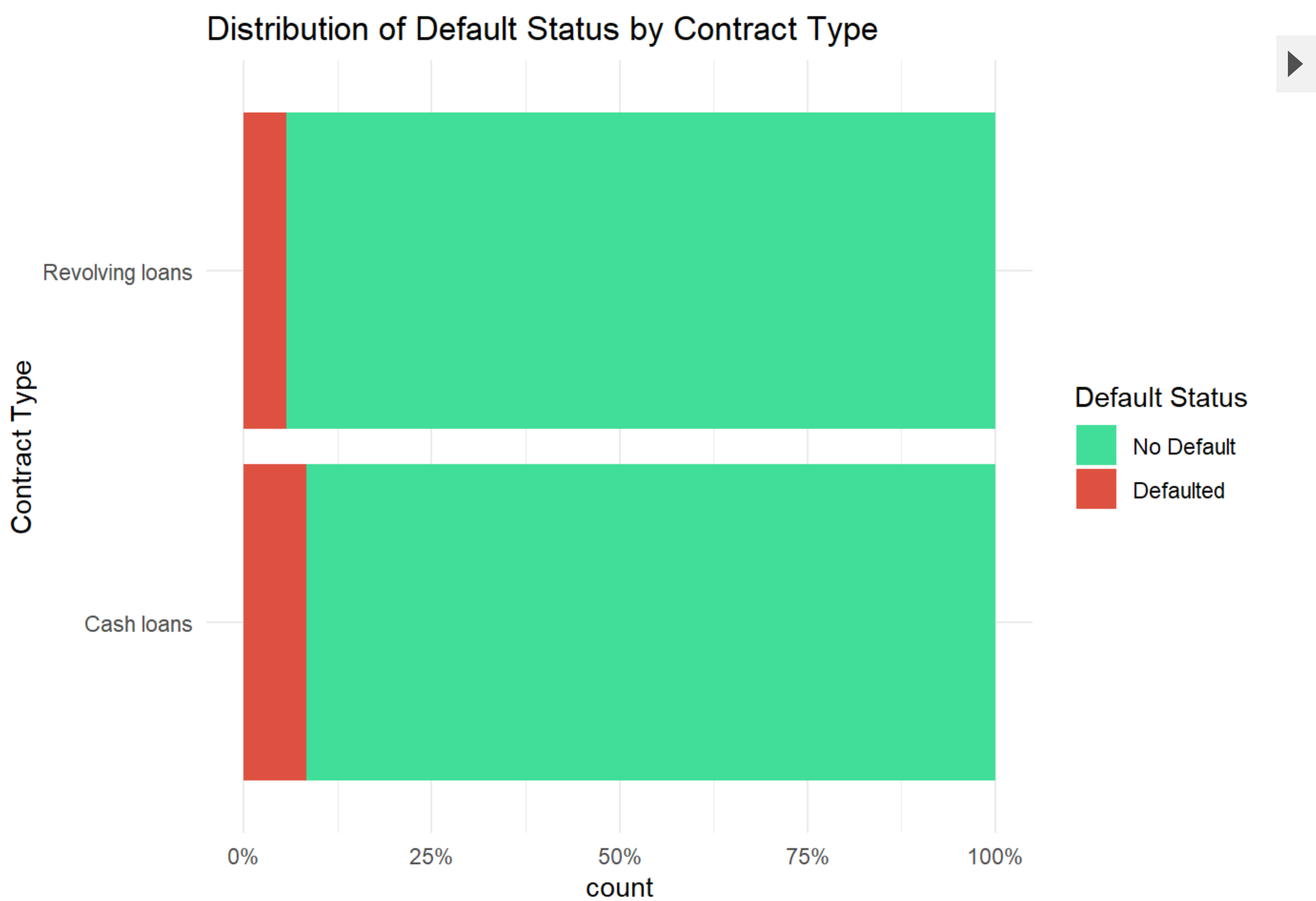
Is there a relationship between a client's contract type and default status? Perhaps certain Contract Types make it harder to pay the loan back.

```
# Count Visualization
ggplot(data = train_clean, aes(x = NAME_CONTRACT_TYPE, fill = as.factor(TARGET))) +
  geom_bar(position = 'dodge') + # plot type
  labs(title = "Distribution of Default Status by Contract Type",
       x = "Contract Type", # x-axis
       fill = "Default Status") +  # Change the legend title here
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +          # Custom labels for the leg
  theme_minimal() + # Specify the theme
  coord_flip() # Flip the graph
```




```
# Percentage Visualization
ggplot(data = train_clean, aes(x = NAME_CONTRACT_TYPE, fill = as.factor(TARGET))) + #Plot Name_Co
```

```
  geom_bar(position = 'fill') + #bars filled in with percentages
  labs(title = "Distribution of Default Status by Contract Type", #custom plot title
       x = "Contract Type", # custom x-axis title
       fill = "Default Status") +  # Change the legend title here
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),  # Assign custom colors
                    labels = c("No Default", "Defaulted")) +        # Custom labels for the legend
  theme_minimal() + # Specify the theme
  coord_flip() + # Flip the graph
  scale_y_continuous(labels = scales::percent_format()) # Convert counts to pctg.
```



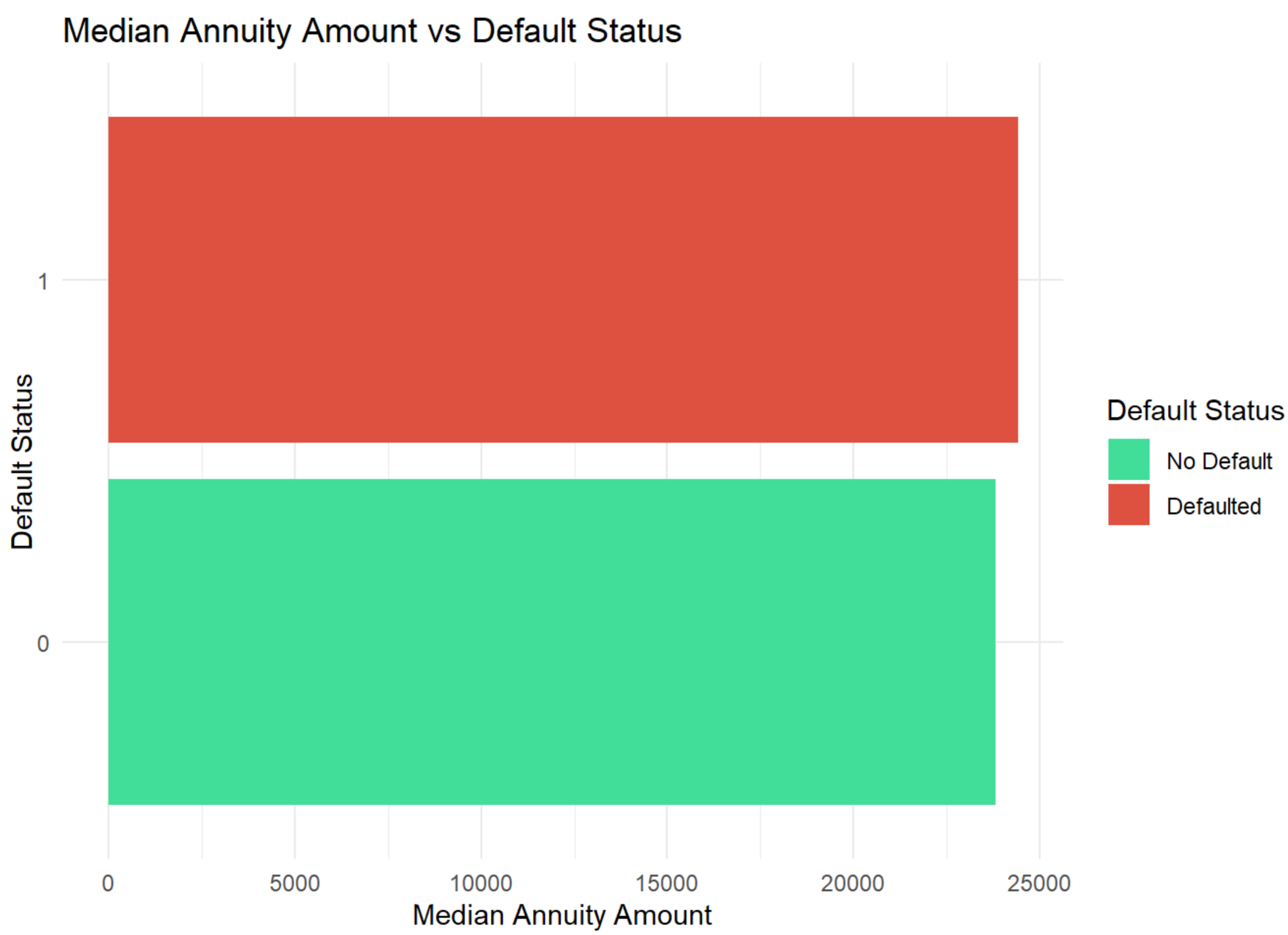
The above plot shows the distribution of Default Status by Contract Type. The first plot shows the count of default status for both revolving and cash loans.

The second plot shows the percentage of default status for each of the two groups which shows a few things. This means that out of all the Revolving Loans issued, the percentage of customers who defaulted is approximately 5% while the customers who did not default is approximately 95%. Likewise out of all the the Cash Loans issued, the percentage of customers who defaulted is approximately 11%, while the customers who did not default is approximately 89%. This means that that overall percentage of default is lower for the revolving loans (5%) when compared to the cash loans (11%).

Median Annuity Amount vs Default Status

Is there a relationship between Median Annuity Amount and Default Status? It could be the case that a lower Median Annuity Amount is easier to pay off. Thus, lower Median Annuity Amount may be lower for those who did not default on their loans.

```
train_clean %>%
  group_by(TARGET) %>% #Group the variables by target
  summarise(median_amt_annuity = median(AMT_ANNUITY)) %>% #Calculate median amt annuity for each 
  ggplot(mapping = aes(x = TARGET, y = median_amt_annuity, fill = TARGET)) +
  geom_col(position = 'dodge') + # Specify plot type
  labs(title = "Median Annuity Amount vs Default Status", # specify chart title
       x = "Default Status", # x-axis title
       y = "Median Annuity Amount", # y-axis title
       fill = 'Default Status') +
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +
  theme_minimal() + # Specify plot theme
  theme(axis.title.y = element_text(margin = margin(t = 50))) + # Adjust top margin of y-axis lab
  coord_flip() # Flip graph
```

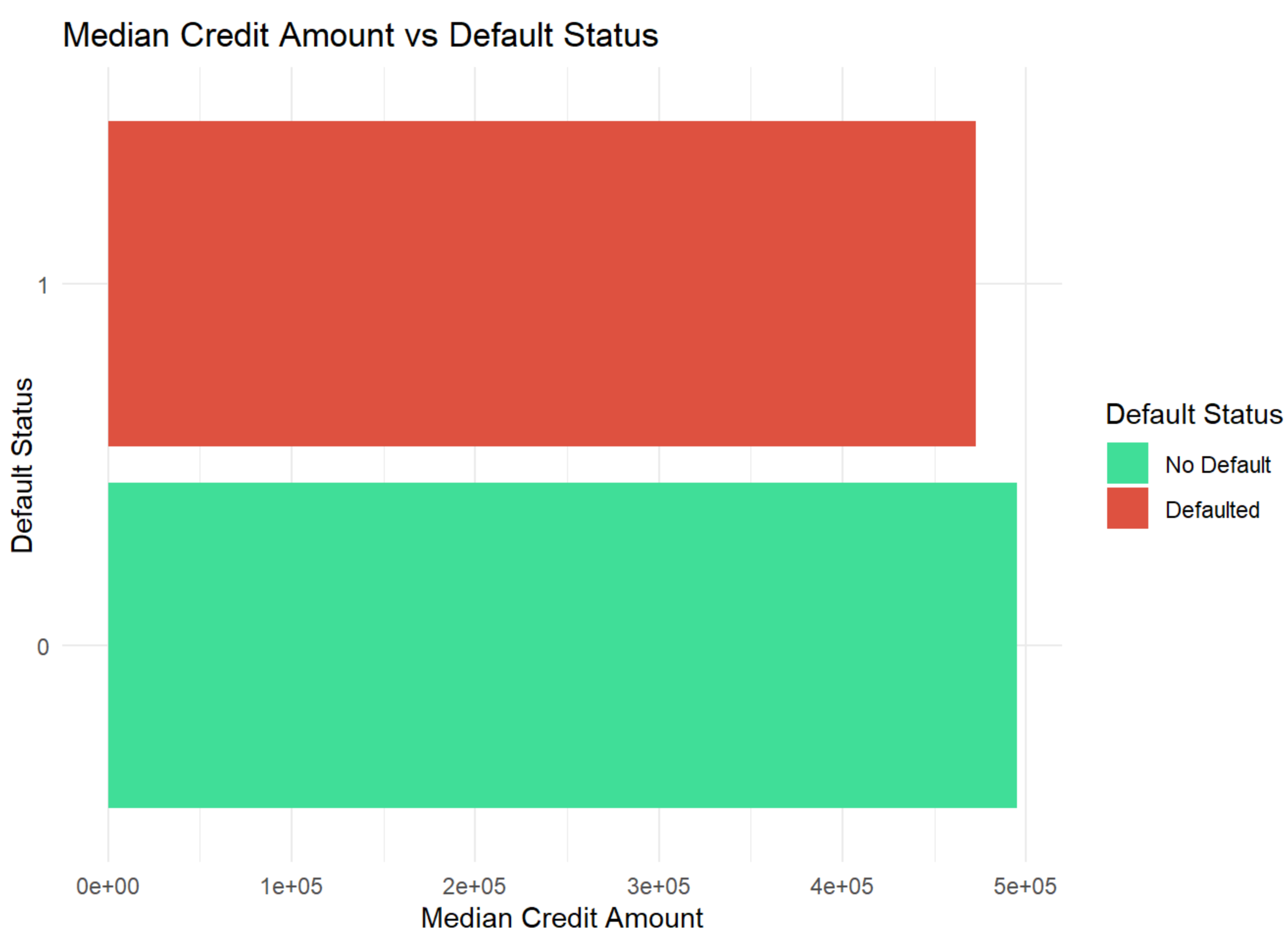


There does not appear to be a strong relationship between Annuity Amount and Default Status. The default and no default bars are very close to each other. The Median Annuity Amount for Defaulted is slightly larger than No Default but not by much.

Median Credit Amount vs Default Status

Is there a relationship between a client's Median Credit Amount and Default Status? It could be possible that clients with higher median credit have lower default rates. This is compared to clients who have lower Median Credit.

```
train_clean %>%
  group_by(TARGET) %>% #Group the variables by target
  summarise(median_credit_amt = median(AMT_CREDIT)) %>% #Calculate median amt annuity for each le
  ggplot(mapping = aes(x = TARGET, y = median_credit_amt, fill = TARGET)) +
  geom_col(position = 'dodge') + #Specify plot type
  labs(title = "Median Credit Amount vs Default Status", # specify title
       x = "Default Status", # x-axis title
       y = "Median Credit Amount", # y-axis title
       fill = 'Default Status') +
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +
  theme_minimal() + # Specify plot theme
  theme(axis.title.y = element_text(margin = margin(t = 50))) + # Adjust top margin of y-axis labe
  coord_flip() # Flip graph
```

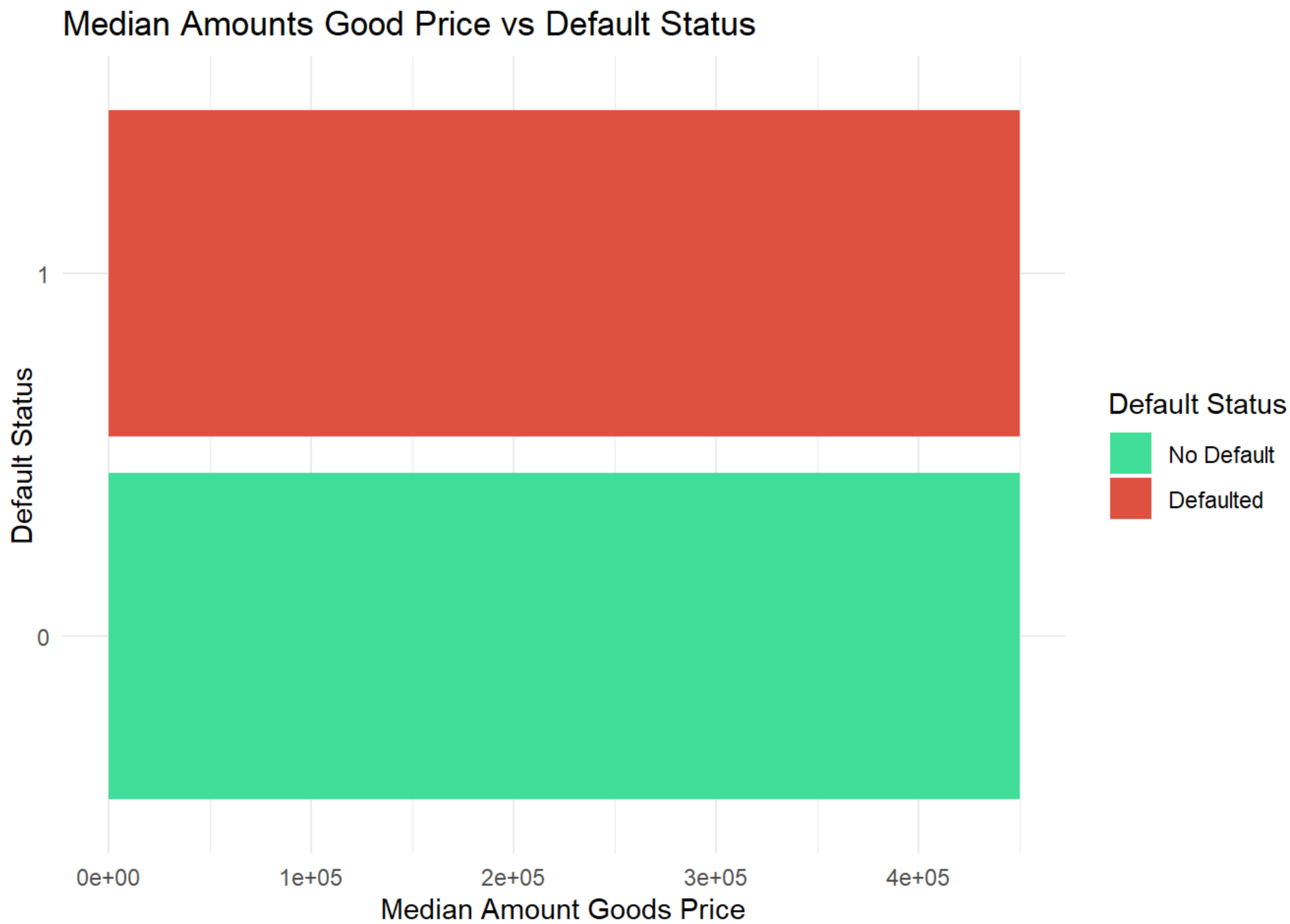


The relationship between Credit Amount and Default Status is also very similar. Interestingly, the Medan Credit Amount for “No Default” is slightly higher than the Credit Amount for “Defaulted”.

Median Amount Goods Price vs Default Status

Is there a relationship between a client’s median amount goods price and default status? It could be that a lower median amount goods price results in less default since the client would need a smaller loan to cover the price of the good.

```
train_clean %>%
  group_by(TARGET) %>% #Group the variables by target
  summarise(median_amt_goods_price = median(AMT_GOODS_PRICE)) %>% #Calculate median amt goods pri
  ggplot(mapping = aes(x = TARGET, y = median_amt_goods_price, fill = TARGET)) +
  geom_col(position = 'dodge') + # Specify plot type
  labs(title = "Median Amounts Good Price vs Default Status", # specify title
       x = "Default Status", # x-axis title
       y = "Median Amount Goods Price", # y-axis title
       fill = 'Default Status') +
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +
  theme_minimal() + # Specify plot theme
  theme(axis.title.y = element_text(margin = margin(t = 50))) +  # Adjust top margin of y-axis lal
  coord_flip() # Flip graph
```

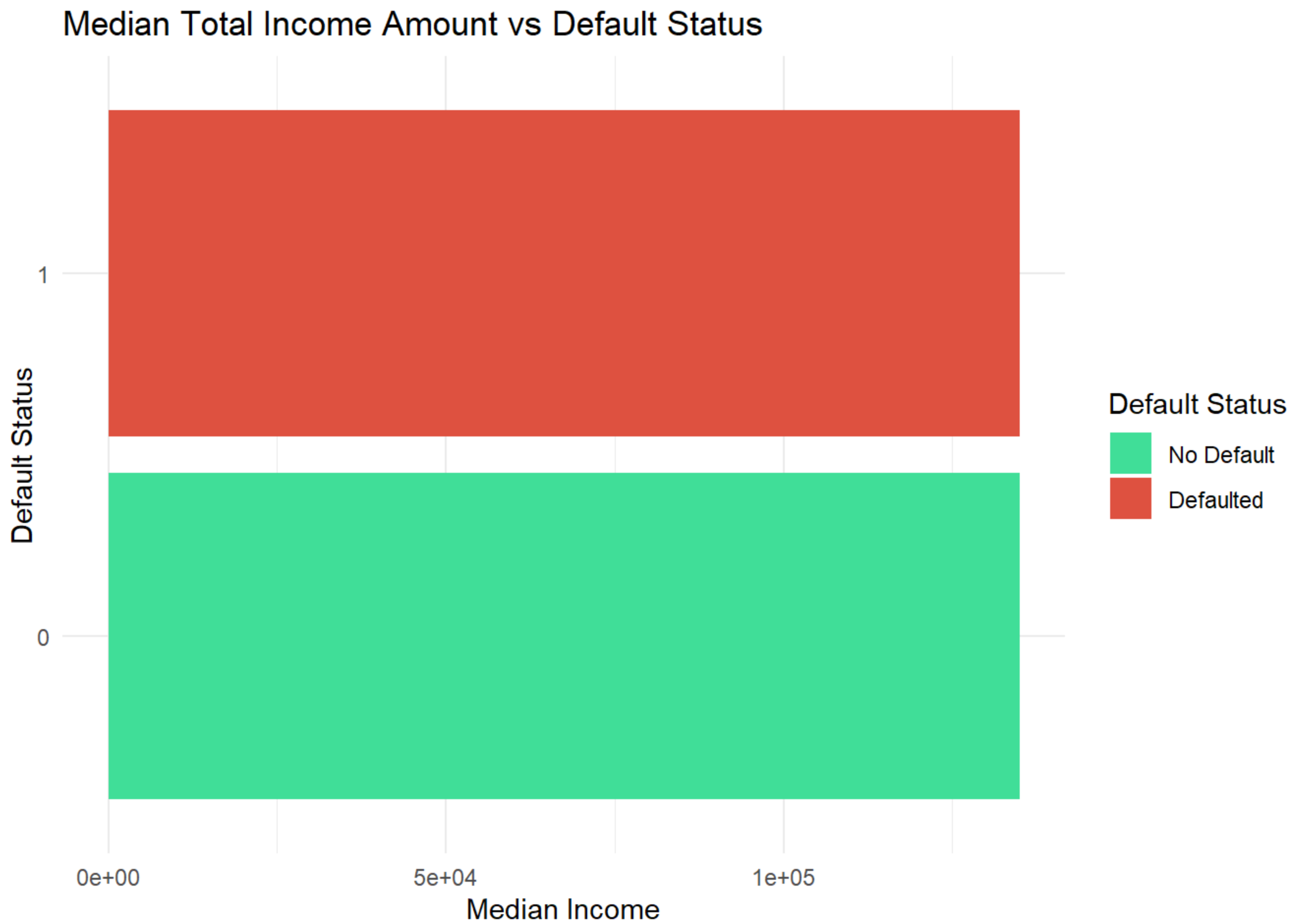


The Median Amount Goods Price is the same for both “No Default” and “Defaulted”. This indicates that there is no relationship between Median Amounts Good Price and Default Status.

Median Total Income vs Default Status

Is there a relationship between a client’s median income for default vs no default? It could be that a higher Median Income results in a lower default rate.

```
train_clean %>%
  group_by(TARGET) %>% #Group the variables by target
  summarise(median_income = median(AMT_INCOME_TOTAL)) %>% # Calculate median amt goods price for 
  ggplot(mapping = aes(x = TARGET, y = median_income, fill = TARGET)) +
  geom_col(position = 'dodge') + # Specify plot type
  labs(title = "Median Total Income Amount vs Default Status",
       x = "Default Status", # x-axis title
       y = "Median Income", # y-axis title
       fill = 'Default Status') +
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +
  theme_minimal() + # Specify plot theme
  theme(axis.title.y = element_text(margin = margin(t = 50))) + # Adjust top margin of y-axis lab
  coord_flip() # Flip graph
```

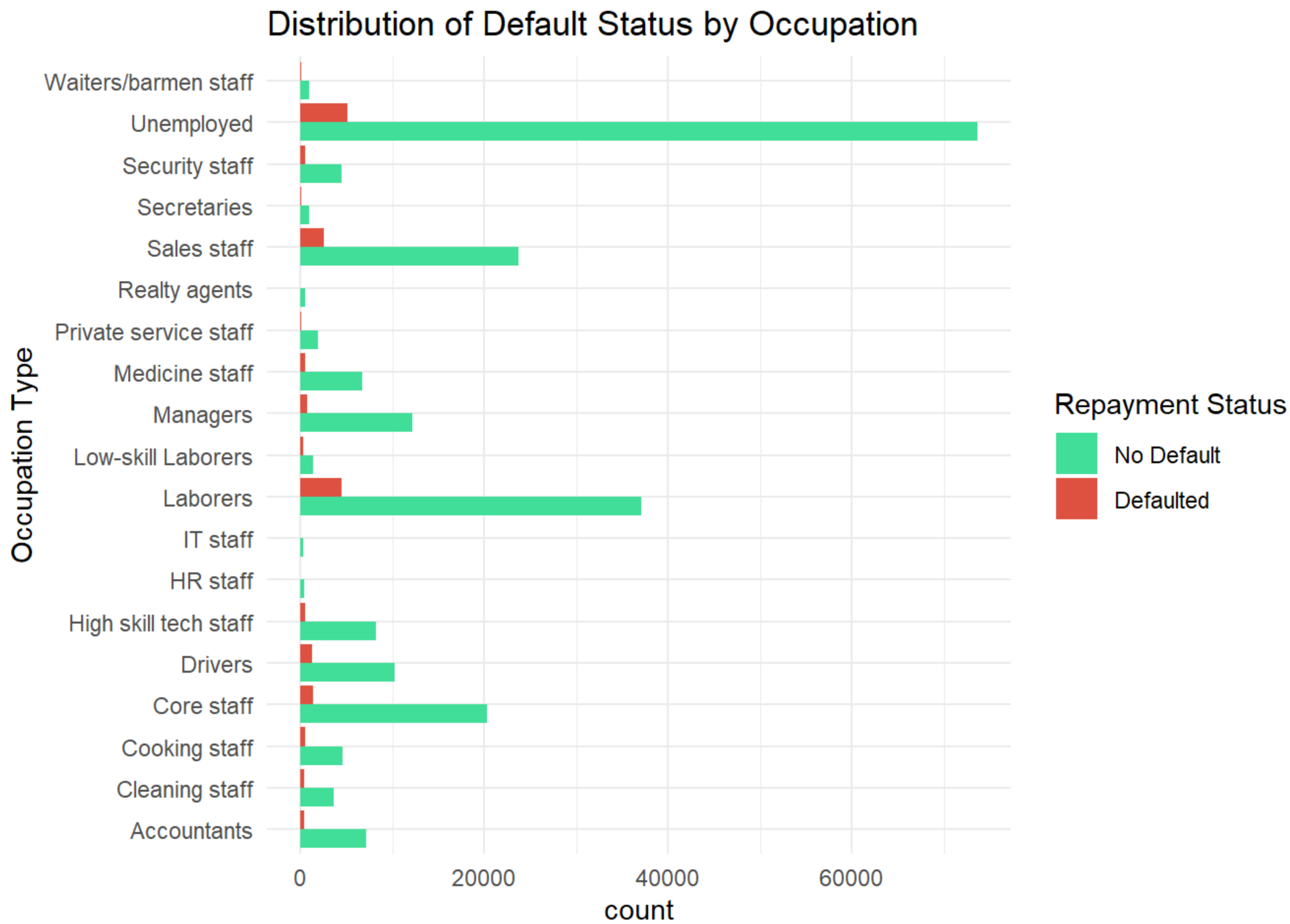


The Median Total Income is the same for both “No Default” and “Defaulted”. This indicates that there may not be a relationship between Median Total Income and Default Status.

Occupation vs Default Status

What is the relationship between a client’s occupation and default status? Certain Higher Paying Occupations like IT for instance might have lower rates of default compared to other groups like Watiers/barmen staff for instance. Perhaps, occupations that traditionally pay higher like IT Staff might have lower rates of default.

```
# Count Plot
ggplot(data = train_clean, aes(x = OCCUPATION_TYPE, fill = as.factor(TARGET))) + #plot occupation_
  geom_bar(position = 'dodge') + # position bars side by side
  labs(title = "Distribution of Default Status by Occupation", # custom plot title
       x = "Occupation Type", # custom x-axis
       fill = "Repayment Status") +  # Change the legend title here
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +          # Custom labels for the lege
  theme_minimal() + # Specify plot theme
  coord_flip() # Flip graph
```

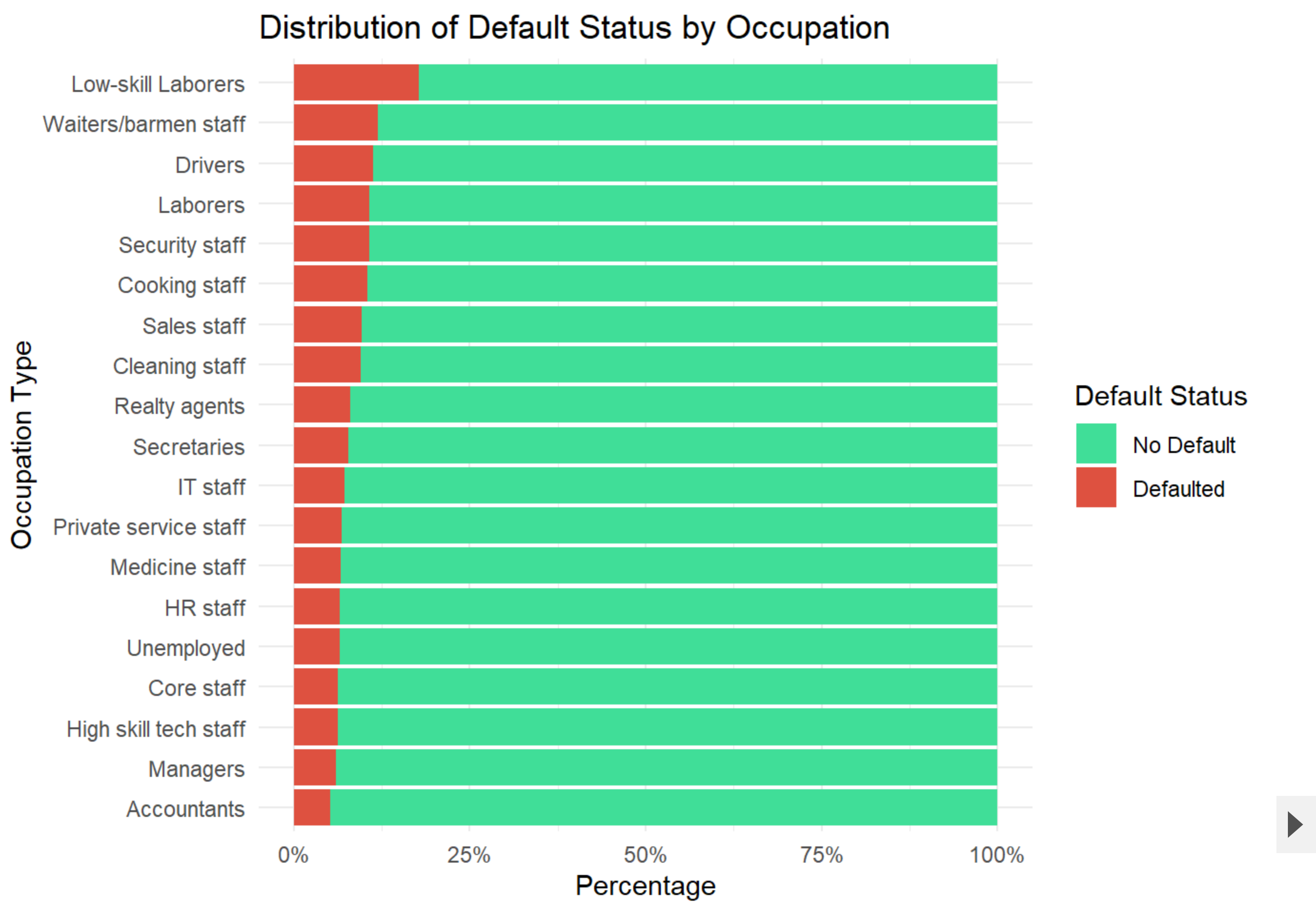



```
# Pctg plot
# First, calculate the percentage of defaults by occupation
default_percent <- train_clean %>% #Assign the train_clean dataset to default_percent
  group_by(OCCUPATION_TYPE) %>% #Group the target by occupation type
  summarize(default_rate = mean(as.numeric(TARGET))) # Calculate the mean of the target

# Reorder the occupation based on default rate
train_clean_01 <- train_clean %>% # Assign changes to another new dataset
  mutate(OCCUPATION_TYPE = factor(OCCUPATION_TYPE,
                                  levels = default_percent$OCCUPATION_TYPE[order(default_percent$

# Now plot with the reordered occupation types
ggplot(data = train_clean_01, aes(x = OCCUPATION_TYPE, fill = as.factor(TARGET))) +
  geom_bar(position = 'fill') + #bars filled in with percentages
  labs(title = "Distribution of Default Status by Occupation",
       x = "Occupation Type",  #custom x-axis title
       fill = "Default Status", #change the legend title here
       y = "Percentage") +  # #custom y-axis title
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"), # Custom colors
                    labels = c("No Default", "Defaulted")) +    # Custom Labels
  scale_y_continuous(labels = function(x) paste0(x * 100, "%")) +
```

```
theme_minimal() + # Specify plot theme
coord_flip() # Flip graph
```



The plots show the relationship between default status and occupation by count and percentage. There are however some slight differences in the graphs due the scaling of the count and percentage axes respectively. For instance, IT and HR Staff show no default on the count plot. They however do have default rates which are reflected on the percentage plot. The reason for this is that the count is much higher for certain groups like Unemployed and ggplot has to scale the count axis to account for the large count size. This unfortunately, however means that smaller groups like IT Staff, etc. have their default count omitted from the graph if it's a small number compared to the rest of the occupation types.

Additionally, in terms of the percentage visualization, Low-skill Laborers have the highest rate of default when compared to their overall group. Low-skill Laborers have a default rate of approximately 20% compared to a no default rate of 80% for the entire group. Afterwards this next group of occupations has a similar level of default:

- Waiters/barmen staff
- Drivers
- Laborers
- Security Staff
- Cooking Staff

- Sales Staff
- Cleaning Staff

The rate of default for these groups when compared to their overall group is approximately 15% which is less then the Low-Skill Laborers. Afterwards, Reality Agents and Secretaries have the same rate of default which is approximately 12%.

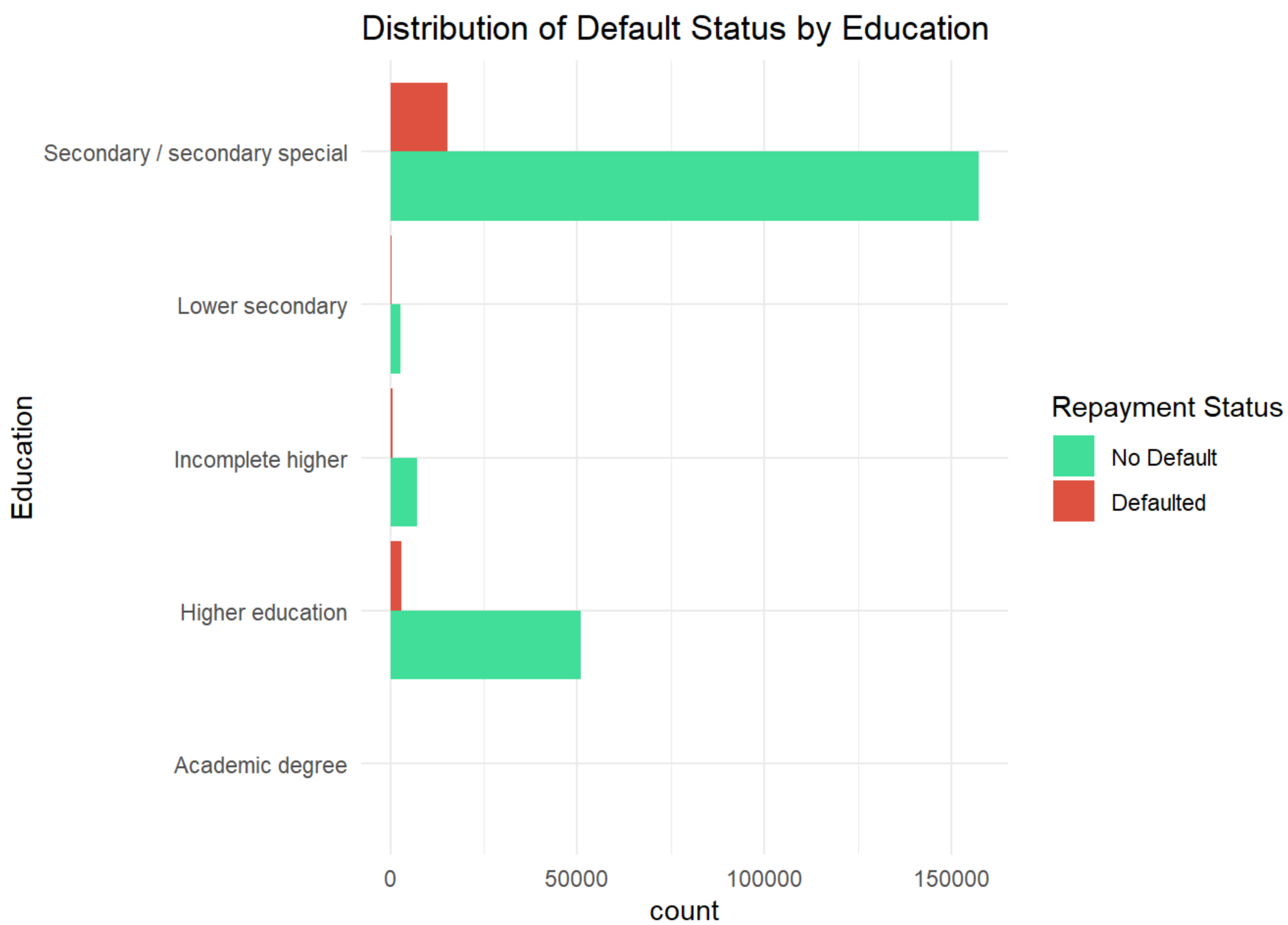
Finally, the rest of the occupations have a default rate that approximately hovers around 5%.

It seems that there is a trend in the default where blue-collar jobs or work that does not necessarily require a degree has a higher rate of default in their overall group when compared to the white-collar jobs.

Education vs Default Status

What is the relationship between a client's education and default status? Higher Education Status could lead to clients making more income and being able to better pay of loans. Lower Education Status in contrast could result in less income which means that a client may have difficulty paying their loan.

```
# Count Plot
ggplot(data = train_clean, aes(x = NAME_EDUCATION_TYPE, fill = as.factor(TARGET))) +
  geom_bar(position = 'dodge') + # put the bars sided by side with dodge
  labs(title = "Distribution of Default Status by Education", #custom plot title
       x = "Education", #custom x-axis title
       fill = "Repayment Status") +  # Change the legend title here
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +          # Custom labels for the lege
  theme_minimal() + # Specify plot theme
  coord_flip() # Flip graph
```

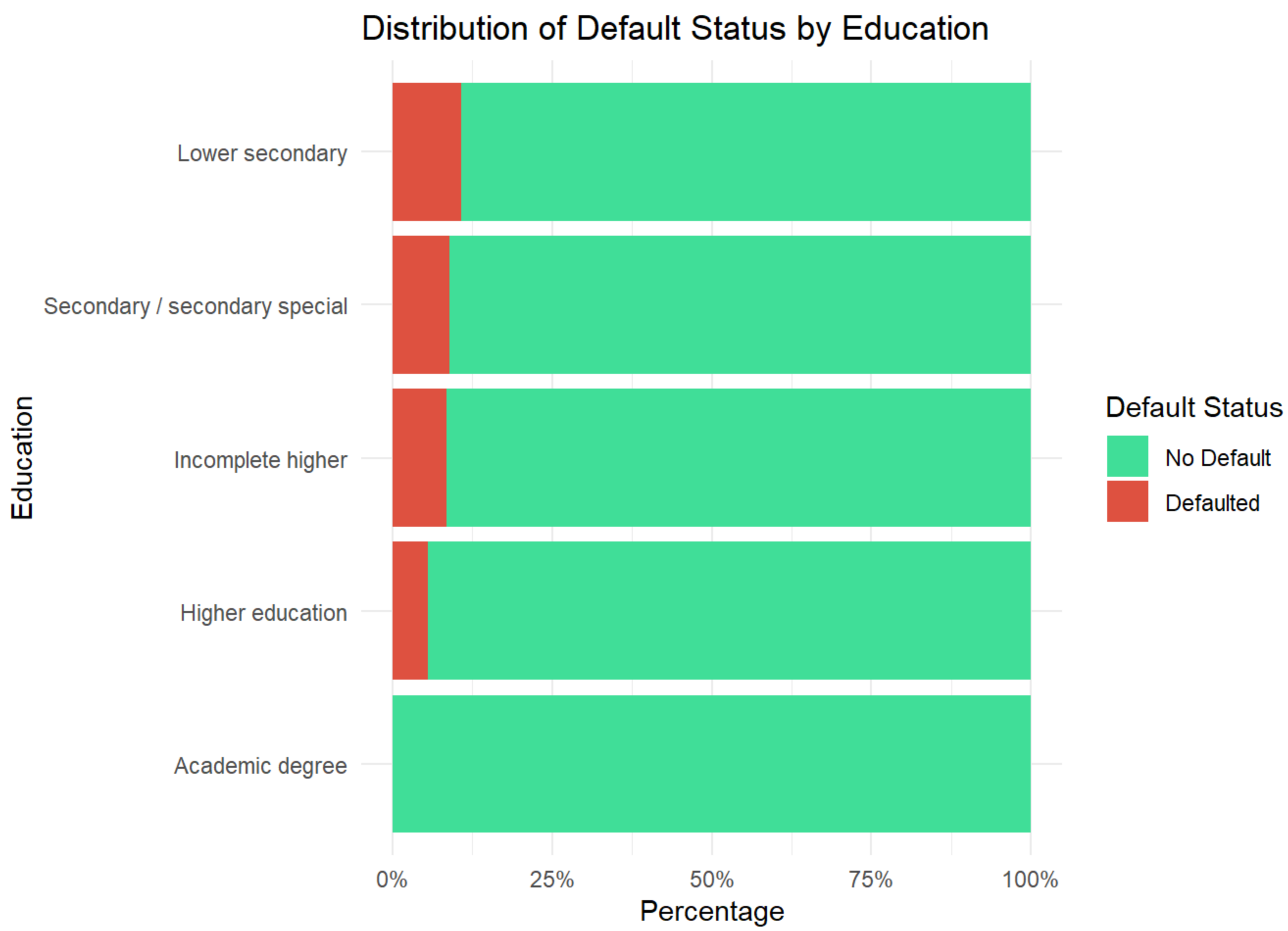



```
# Percentage Plot
# First, calculate the percentage of defaults by NAME_EDUCATION_TYPE
default_percent <- train_clean %>% # Create a new df to store these results
  group_by(NAME_EDUCATION_TYPE) %>% # Sort the df by NAME_EDUCATION_TYPE
  summarize(default_rate = mean(as.numeric(TARGET))) # convert target to numeric and calculate pr

# Reorder the NAME_EDUCATION_TYPE based on default rate
train_clean_01 <- train_clean %>% # Assign changes to another new dataset
  mutate(NAME_EDUCATION_TYPE = factor(NAME_EDUCATION_TYPE,
                                      levels = default_percent$NAME_EDUCATION_TYPE[order(default_perc

# Now plot with the reordered NAME_EDUCATION_TYPE types
ggplot(data = train_clean_01, aes(x = NAME_EDUCATION_TYPE, fill = as.factor(TARGET))) +
  geom_bar(position = 'fill') + #bars filled in with percentages
  labs(title = "Distribution of Default Status by Education", #custom title
       x = "Education", #custom x-axis title
       fill = "Default Status", #Change the legend title
       y = "Percentage") +  #custom y-axis title
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),  # Custom colors
                    labels = c("No Default", "Defaulted")) +          # Custom labels
  scale_y_continuous(labels = function(x) paste0(x * 100, "%")) + # Convert cnt to pctg.
```

```
  theme_minimal() +  # Specify plot theme
  coord_flip() # Flip graph
```



The first visualization shows the count of default status by education attainment. The second visualization shows the percentage of default status by education attainment. Based on the second plot, there is a decrease in the default rate as education level increases with Higher Education having the lowest default rate.

It should be noted that everyone who has an academic degree was able to pay off their loan. The issue however is that academic degree has relatively few observations when compared to the other educational groups. This can be seen from the first visualization where academic degree does not have as many observations as Secondary/secondary special for instance. Consequently, in order to adequately scale the higher counts for the rest of the groups, Academic Degree been excluded.

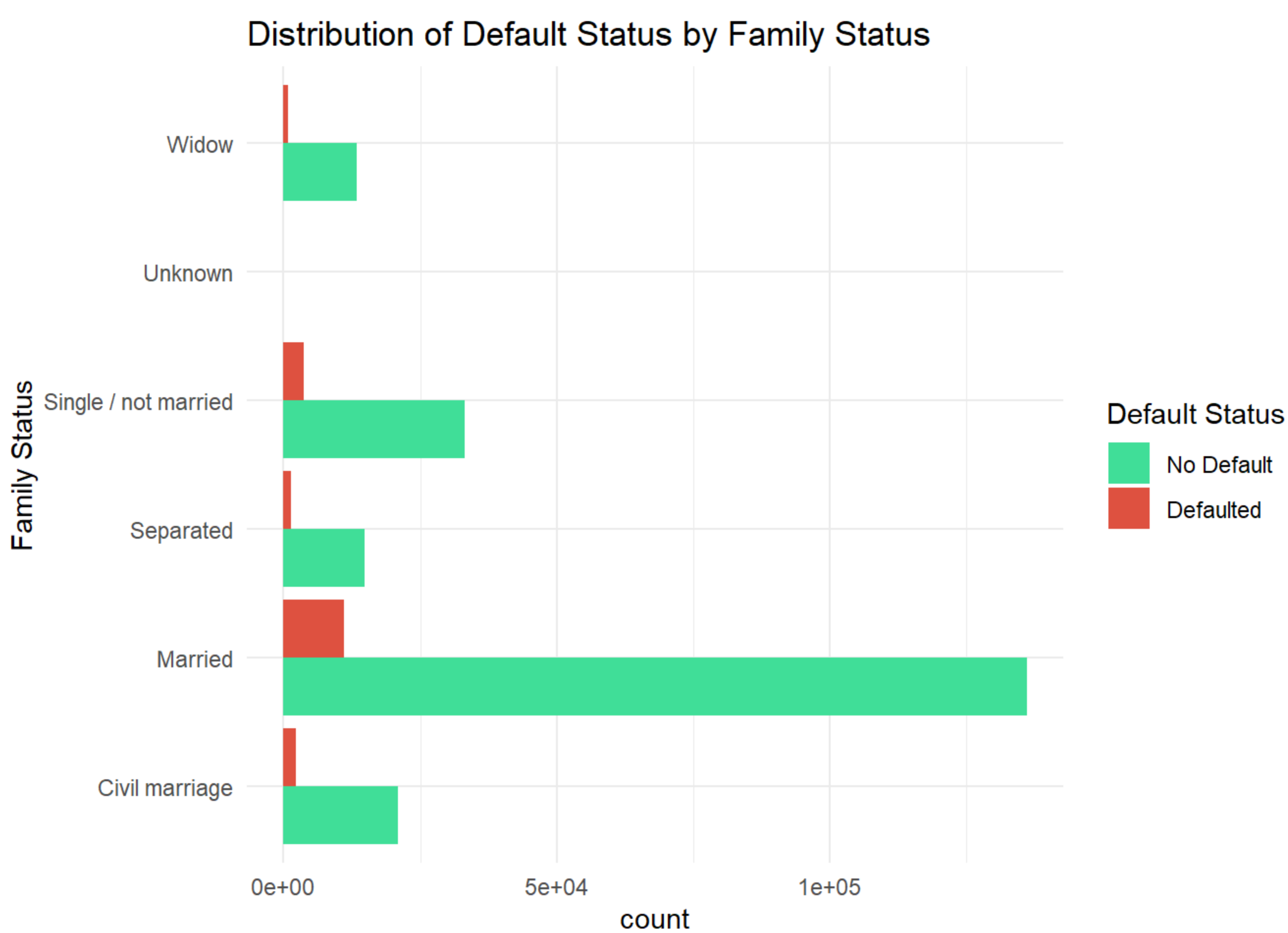
This also means that people who have academic degrees may not necessarily get loans from Home Credit very frequently since they probably can get loans from more conventional banks. (A person with an academic degree will likely make more money which means that they can get a loan from a conventional bank. Consequently, this means that the number of people who would get a loan from Home Credit is low which is also why their count is relatively low compared to the other educational groups.)

Nonetheless, academic degree aside, it seems that the groups with higher educational attainment have a a slightly default rate compared to lower educational attainment.

Family Status vs Default Status

The Family Status could have an impact on a client's ability to pay back loans. For instance, a married couple will generally have more income then an unmarried client. Thus, a married couple may be able to better pay of their loan and avoid default compared to an unmarried client.

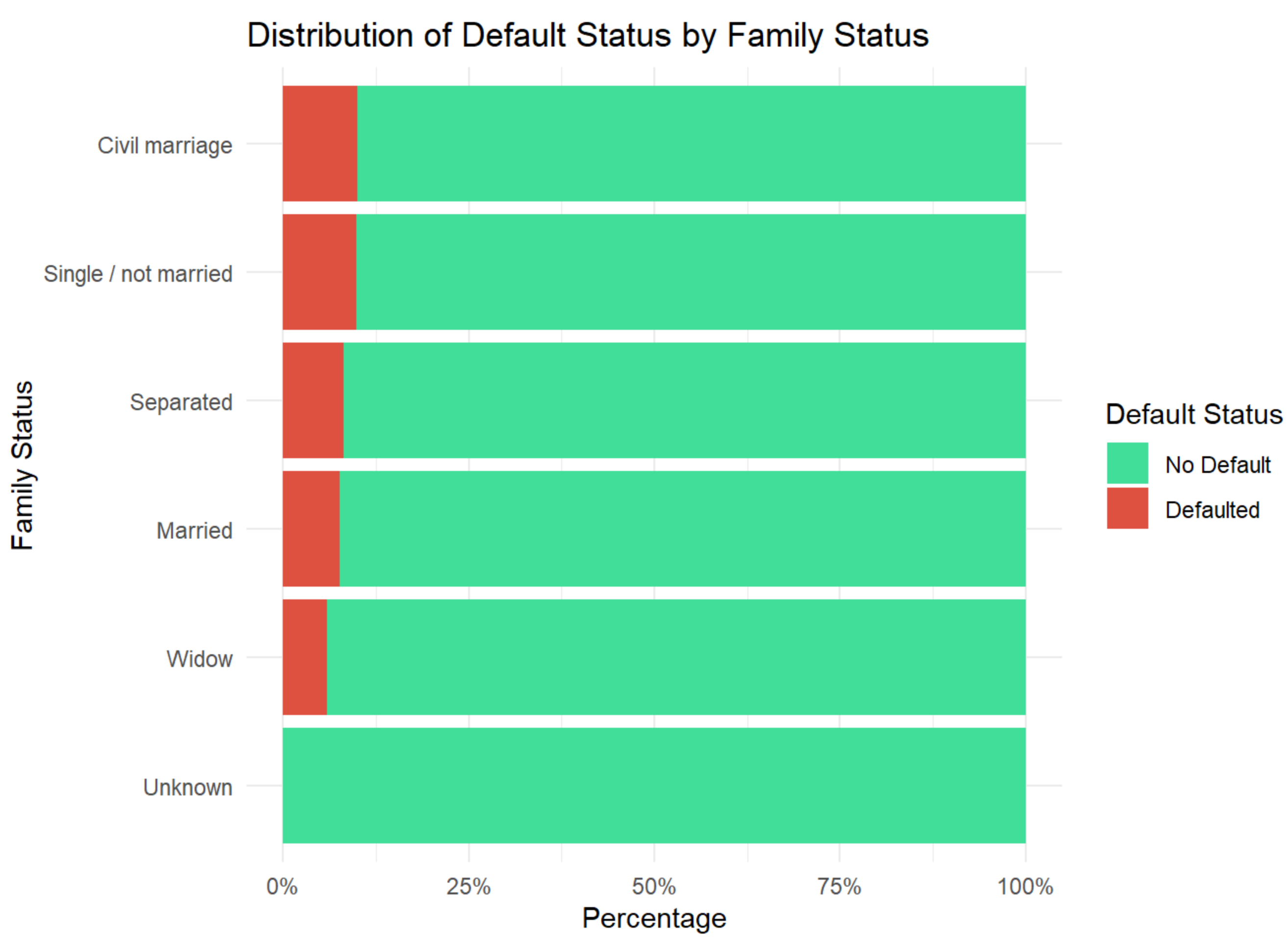
```
# COunt Plot
ggplot(data = train_clean, aes(x = NAME_FAMILY_STATUS, fill = as.factor(TARGET))) + # Plot Family_
  geom_bar(position = 'dodge') + # bars side by side
  labs(title = "Distribution of Default Status by Family Status", # Custom title
       x = "Family Status", # custom x-axis title
       fill = "Default Status") +  # Change the legend title here
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +          # Custom labels for the leg
  theme_minimal() + # Assign a theme
  coord_flip() # Flilp the graph
```




```
# First, calculate the percentage of defaults by NAME_FAMILY_STATUS
default_percent <- train_clean %>% # Create a new df to store these results
  group_by(NAME_FAMILY_STATUS) %>% # Sort the df by NAME_FAMILY_STATUS
  summarize(default_rate = mean(as.numeric(TARGET))) # convert target to numeric and calculate pr
```

```
# Reorder the occupation based on default rate
train_clean_01 <- train_clean %>% # Assign changes to another new dataset
  mutate(NAME_FAMILY_STATUS = factor(NAME_FAMILY_STATUS,
                                     levels =  default_percent$NAME_FAMILY_STATUS[order(default_perc

# Now plot with the reordered occupation types
ggplot(data = train_clean_01, aes(x = NAME_FAMILY_STATUS, fill = as.factor(TARGET))) + #Plot with
  geom_bar(position = 'fill') + #bars filled in with percentages
  labs(title = "Distribution of Default Status by Family Status", # custom title
       x = "Family Status", #custom x-axis title
       fill = "Default Status", # Change the legend title here
       y = "Percentage") +  #Custome y-axis title
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),  # Assign custom colors
                    labels = c("No Default", "Defaulted")) +       # Custom labels for the legend
  scale_y_continuous(labels = function(x) paste0(x * 100, "%")) + # Convert counts to pctg.
  theme_minimal() + #Assign a theme
  coord_flip() # Flipl the graph
```



The first bar plot shows the relationship between Family Status and Default Status in counts while the second bar plot shows the same relationship in percentages. It appears that the Civil Marriage and Single/not married groups have the highest rate of default compared to the other Family Status groups. (Rate of default for both groups is roughly 10%) Afterwards, Separated and Married have the same rate of default at 9% which is slightly lower. Finally, Widow is slightly lower at 9%.

The Unknown group however has no defaults with every observation successfully paying their loan off. It should be noted that the “Unknown” group has relatively few observations when compared to the other Family Status groups.

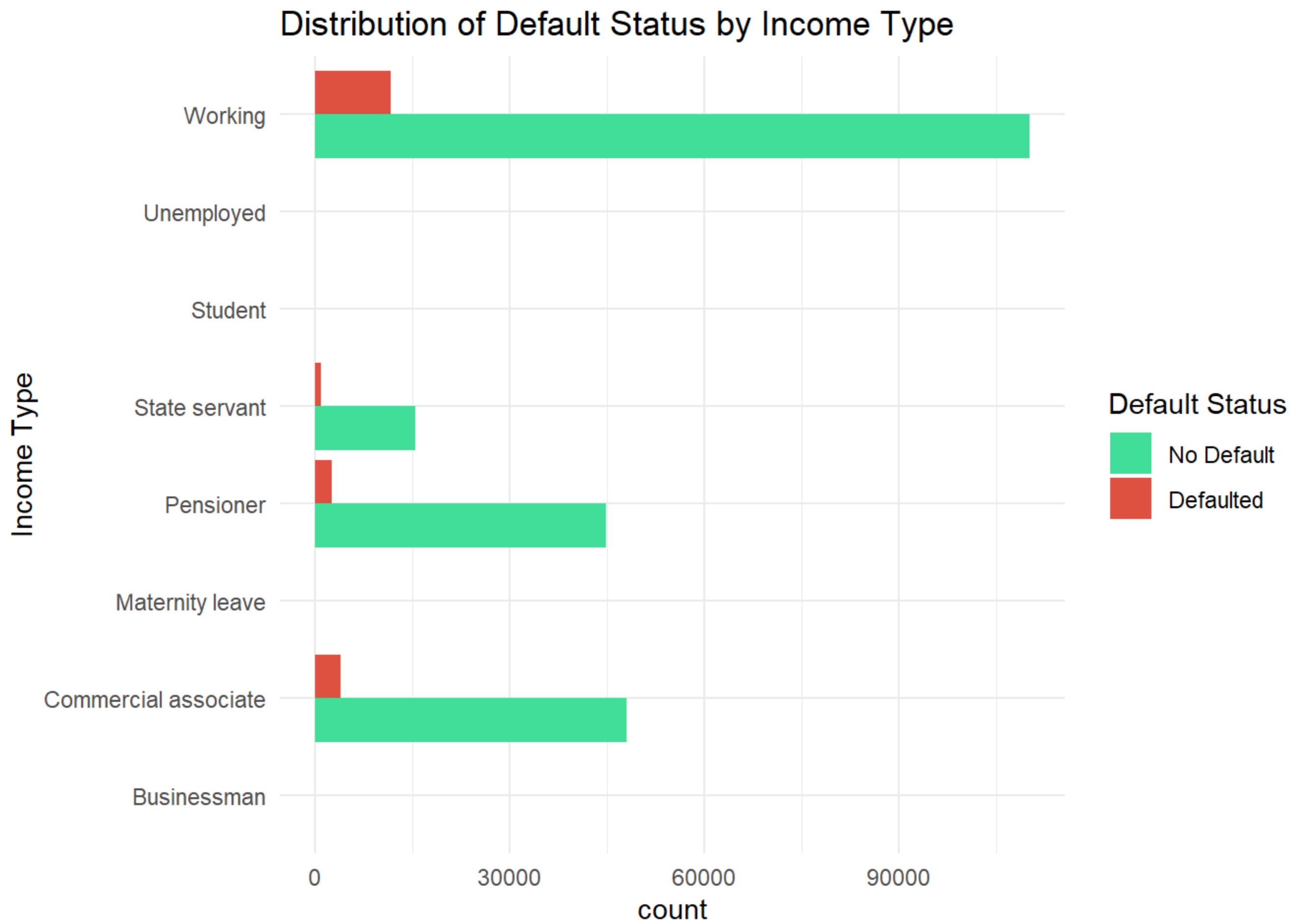
This can be seen from the first visualization where the “Unknown” group does not have as many observations as the “Married” group for instance. Consequently in order to adequately scale the much higher counts for the rest of the groups, ggplot has excluded “Unknown” from the first plot.

Finally, it seems like family status may not have an impact on the default rate. For instance, Separated and Married are opposite family classes yet they have the same default rate. Likewise Single/not married and Civil Marriage which could be perceived as opposites also have the same default rate.

Income Type vs Default Status

What is the relationship between a client’s income type and default status? It could be that clients with high paying jobs default less when compared to clients with low paying jobs.

```
# Count Plot
ggplot(data = train_clean, aes(x = NAME_INCOME_TYPE, fill = as.factor(TARGET))) +
  geom_bar(position = 'dodge') + # side by side bars
  labs(title = "Distribution of Default Status by Income Type", # custom title
       x = "Income Type", #custom x-axis
       fill = "Default Status") +  # Change the legend title here
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +          # Custom labels for the leg
  theme_minimal() + # Assign a theme
  coord_flip() # Flip the graph
```

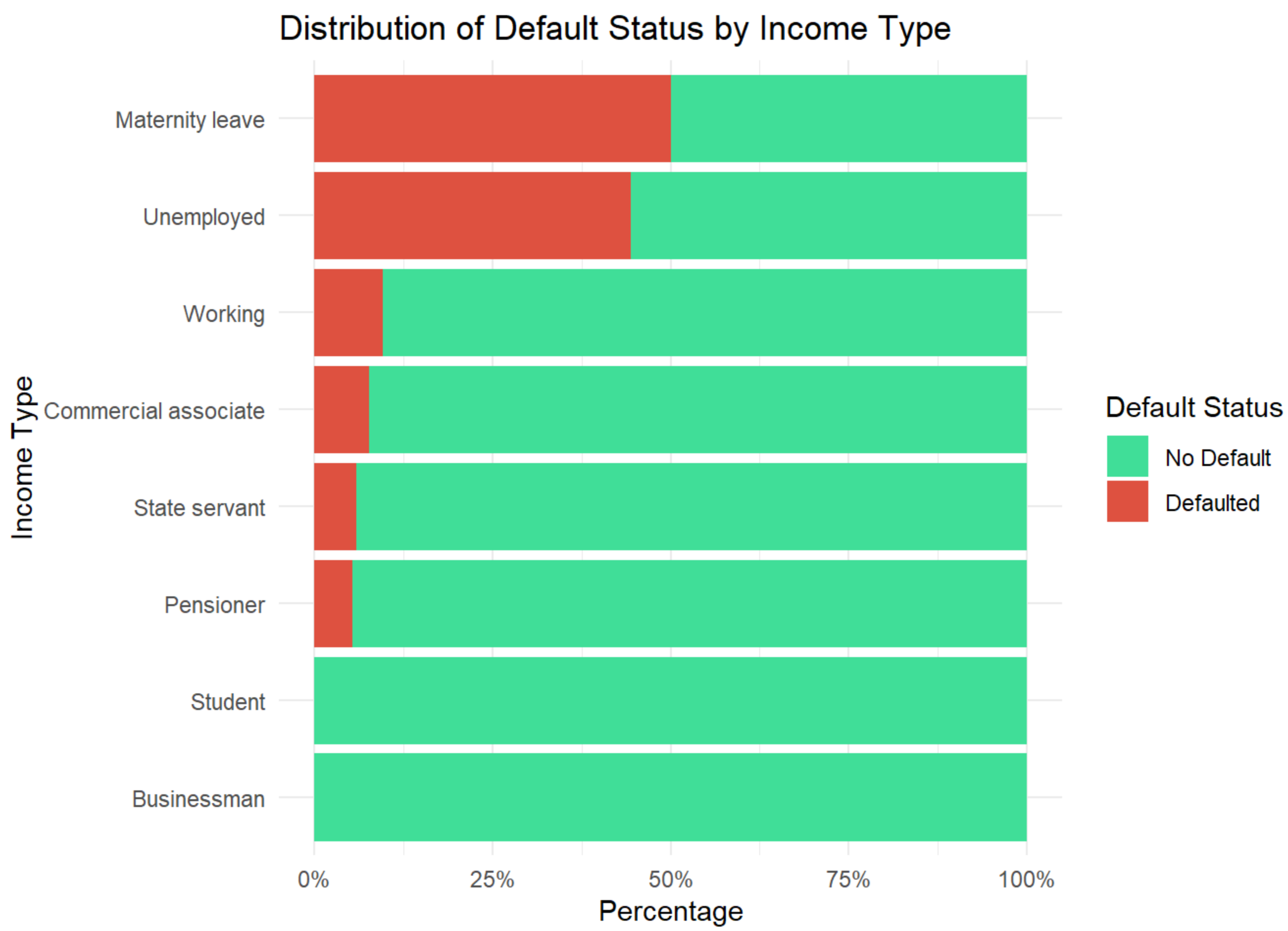



```
# First, calculate the percentage of defaults by NAME_INCOME_TYPE
default_percent <- train_clean %>% # Create a new df to store these results
  group_by(NAME_INCOME_TYPE) %>% # Sort the df by NAME_INCOME_TYPE
  summarize(default_rate = mean(as.numeric(TARGET))) # Calculate default_rate for NAME_INCOME_TYPI

# Reorder the occupation based on default rate
train_clean_01 <- train_clean %>% # Create another dataframe called train_clean_01
  mutate(NAME_INCOME_TYPE = factor(NAME_INCOME_TYPE, #Convert NAME_INCOME_TYPE to a factor
                                   levels =  default_percent$NAME_INCOME_TYPE[order(default_percen

# Percentage Plot
# Now plot with the reordered occupation types
ggplot(data = train_clean_01, aes(x = NAME_INCOME_TYPE, fill = as.factor(TARGET))) + # Plot percer
  geom_bar(position = 'fill') + #Fill in bar graph w/ pctg.
  labs(title = "Distribution of Default Status by Income Type", # custom title
       x = "Income Type", #custom x-axis title
       fill = "Default Status", #change the legend title here
       y = "Percentage") +  #custom y-axis title
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),  #Assign custom colors
                    labels = c("No Default", "Defaulted")) +      # Custom labels for the legend
  scale_y_continuous(labels = function(x) paste0(x * 100, "%")) + # convert cnt to pctg.
```

```
  theme_minimal() + # Assign a theme
  coord_flip() # Flip the graph
```



The first plot shows how the default status varies by "income type" in counts while the second plot shows how default status the same relationship as percentages. Additionally, the second bar graph shows that Maternity Leave and "Unemployed" have the highest rates of default when compared to their overall percentage. Interestingly, however counts for both categories do not appear on the first bar plot. This indicates that the the count of overall clients in "Maternity Leave" and "Unemployed" is low when compared to other groups like "Working".

Thus, ggplot has left these classes out to accommodate the much larger classes like "Working" and "Commercial Associate". However, even though the number of clients in "Maternity Leave" and "Unemployed" are small, it is interesting that a large portion of the clients still defaulted. For instance, out of all the clients (women in this case) who are classified as maternity leave, more than 50% of the clients defaulted on their loans. Likewise out of all the clients who are unemployed, almost exactly 50% defaulted on their loans. This does seem to align with broader trends where "employment status" and "maternity leave" impact one's ability to do other things. (The amount of clients who fall into "Unemployed" and "Maternity Leave" for Home Credit is small, so it can't be said for certain whether this is a definitive trend. There are only 4 clients in the maternity leave and 18 clients who are unemployed in the data set for the cleaned data. The unclean data had 5 clients in maternity leave and 22 unemployed clients.)

Afterwards, the next 4 categories, “Working”, “Commercial Associate”, “State Servant” and “Pensioner” all have much lower levels of default while “Student and”Business” have no default at all. There are however a few things that should be noted:

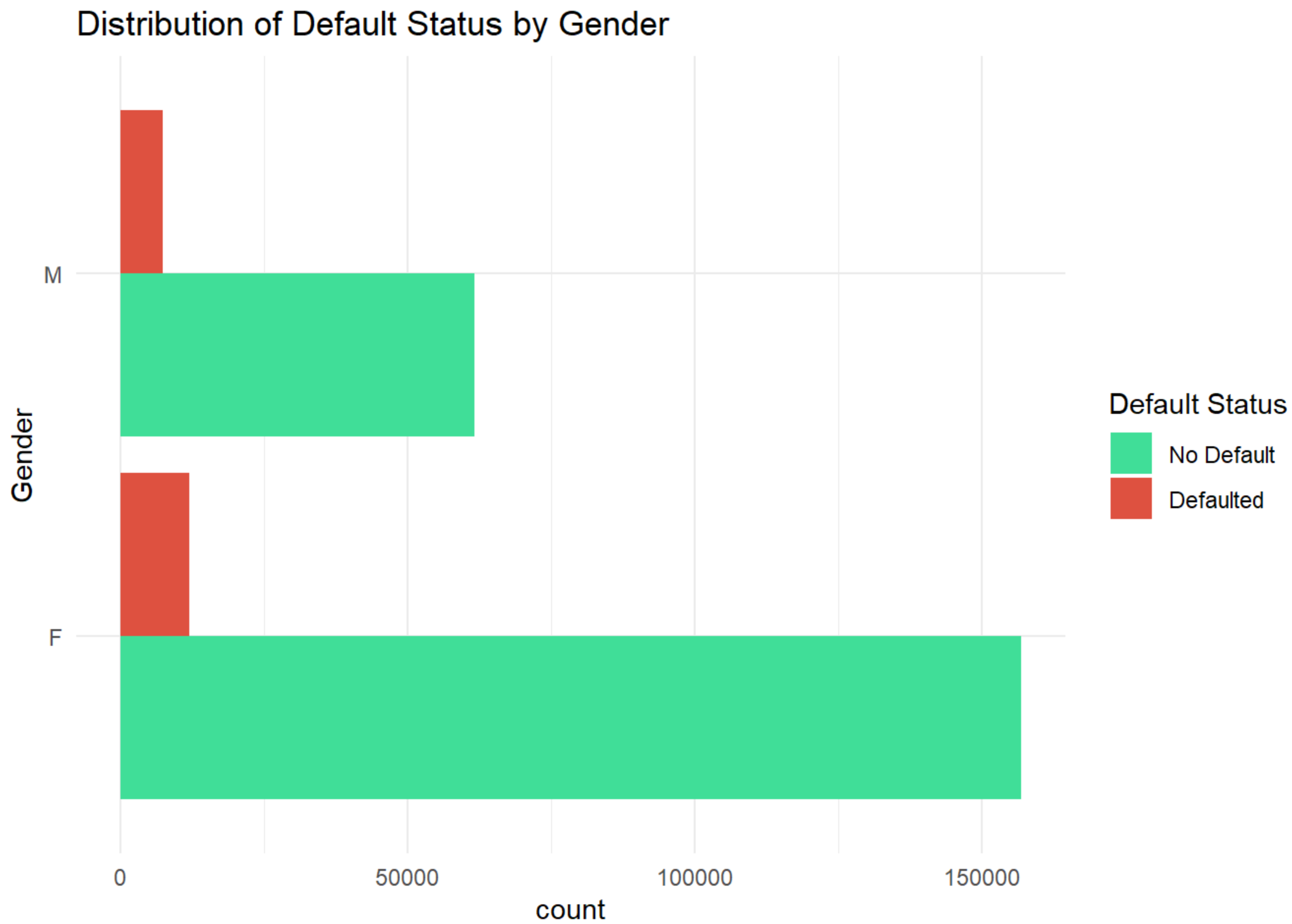
In the first plot, Student and Businessmen do not show up on the first plot which indicates that their count is relatively small compared to the other groups. This is also supported by the table which shows that there are only 16 students and 4 businessmen. Thus, to accommodate larger groups like Working, the values were left out of the count plot. However, in the second plot, it shows that students and businessmen do not have any defaults. This is interesting however the sample size of students and businessmen in this data set is quite small. Thus, this might not be proof a strong association between being a student or a businessman and not defaulting.

Finally the last 3 groups with relatively large counts (State Servant, Working and Pensioners) have similar level of defaults. The default rate for these groups hovers around 8 to 10%. Thus, it seems that Home Credit may already give preference to people who have an employment type. This results in larger observations for these groups and a similar level of default as well. However for the rare occasions when Home Credit does give loans to Maternity Leave or Unemployed Clients, the default rate is higher, despite Home Credit likely doing stricter checks on these clients. (Pre-Existing Stricter Checks would also explain why Maternity Leave and Unemployed represent a small portion of the Income Type Variable.)

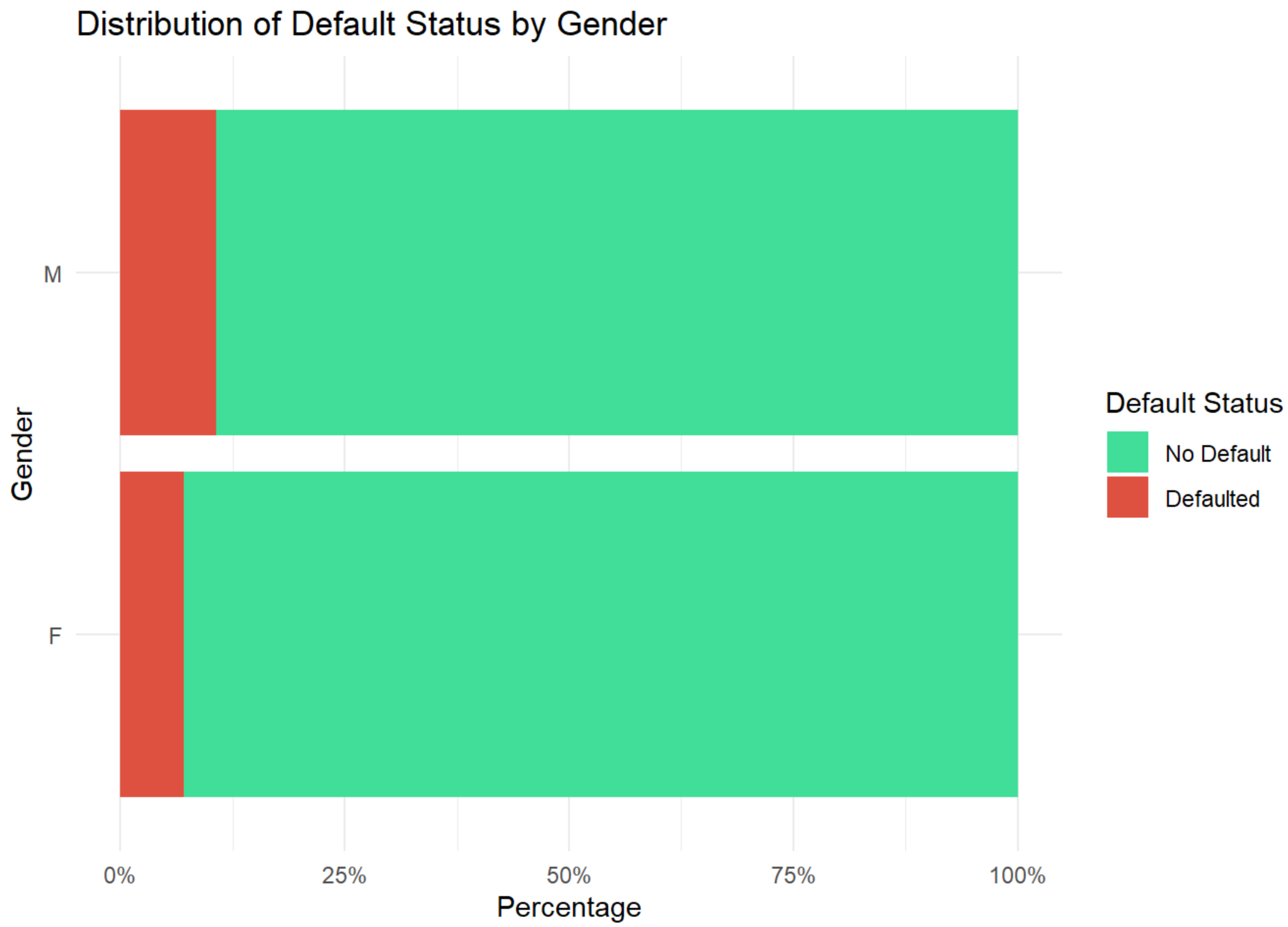
Gender vs Default Status

Is there a relationship between gender and a client’s default status? It is well known that gender plays a role in income, career advancement, etc. and several other phenomena. Thus it’s possible that gender could have an influence on the client’s default status.

```
# Count Visualization
ggplot(data = train_clean, aes(x = CODE_GENDER, fill = as.factor(TARGET))) + #x should be gender
  geom_bar(position = 'dodge') + # side by side bar
  labs(title = "Distribution of Default Status by Gender", # custom title
       x = "Gender", # custom x-axis title
       fill = "Default Status") +  # Change the legend title here
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                    labels = c("No Default", "Defaulted")) +          # Custom labels for the leg
  theme_minimal() + # Assign a theme
  coord_flip() # flip the graph
```




```
# Percentage Visualization
ggplot(data = train_clean, aes(x = CODE_GENDER, fill = as.factor(TARGET))) + #x should be gender
  geom_bar(position = 'fill') + # fill each bar with percentage
  labs(title = "Distribution of Default Status by Gender", #custom title
       x = "Gender", # custom x-axis title
       fill = "Default Status", #change the legend title
       y = "Percentage") +  #custom y-axis title
  scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),  # Assign custom colors
                    labels = c("No Default", "Defaulted")) +      # Custom labels for the legend
  scale_y_continuous(labels = function(x) paste0(x * 100, "%")) +  # Convert to percentages
  theme_minimal() + # Assign a theme
  coord_flip() # flip the graph
```



The first plot shows distribution of default status by gender as counts. The second plot shows the same relationship but as a percentage. The second plot indicates that out of all the men in the dataset, approximately 10% have defaulted on their loans. Likewise for all the women in this dataset, approximately 8% of the women in the dataset have defaulted on their loans. This indicates that gender may have an influence on determining default or difficulty in paying the loan.

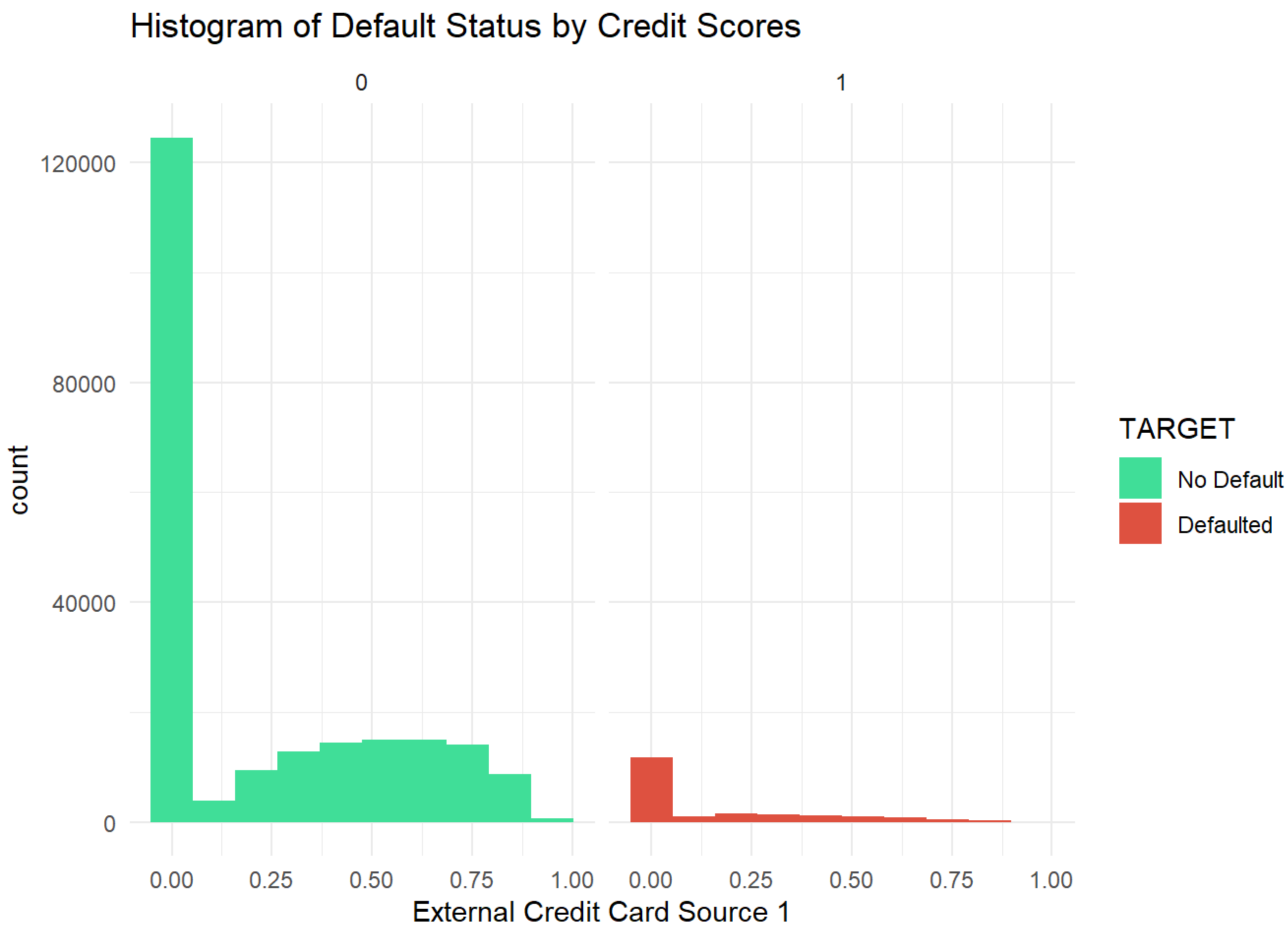
Histograms

Default Status by Credit Scores Source 1

Is there a relationship between the credit scores and a client's default status? Credit Scores are often utilized to predict whether someone will pay back a loan. Although in Home Credit's case a lot of customers might not have credit scores since Home Credit loans to underserved populations. Either way though, credit scores could still be helpful for predicting who can pay a loan back successfully.

```
train_clean |>  # call train clean and avoid calling it in ggplot
     ggplot(mapping = aes(x = EXT_SOURCE_1, fill = TARGET)) + # use Ext_Source_1 to graph plot
     geom_histogram(bins = 10) + # custom bins of 10
     facet_wrap(facets = ~TARGET) + # Create 2 histograms
     labs(title = "Histogram of Default Status by Credit Scores", # title
          x = "External Credit Card Source 1") + # X-axis
      scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
```

```
                   labels = c("No Default", "Defaulted")) + # Custom labels
  theme_minimal() # Assign a theme
```



The plot shows that there are more customers in each of credit score bins for customers who did not default compared to customers who do default. This seems to suggest that having a credit score in itself is indicative of default or no default. Interestingly, there is a large number of customers who did not default despite having a credit score of zero. This however could be due to the imputation of giving customers a zero if they did not have credit scores.

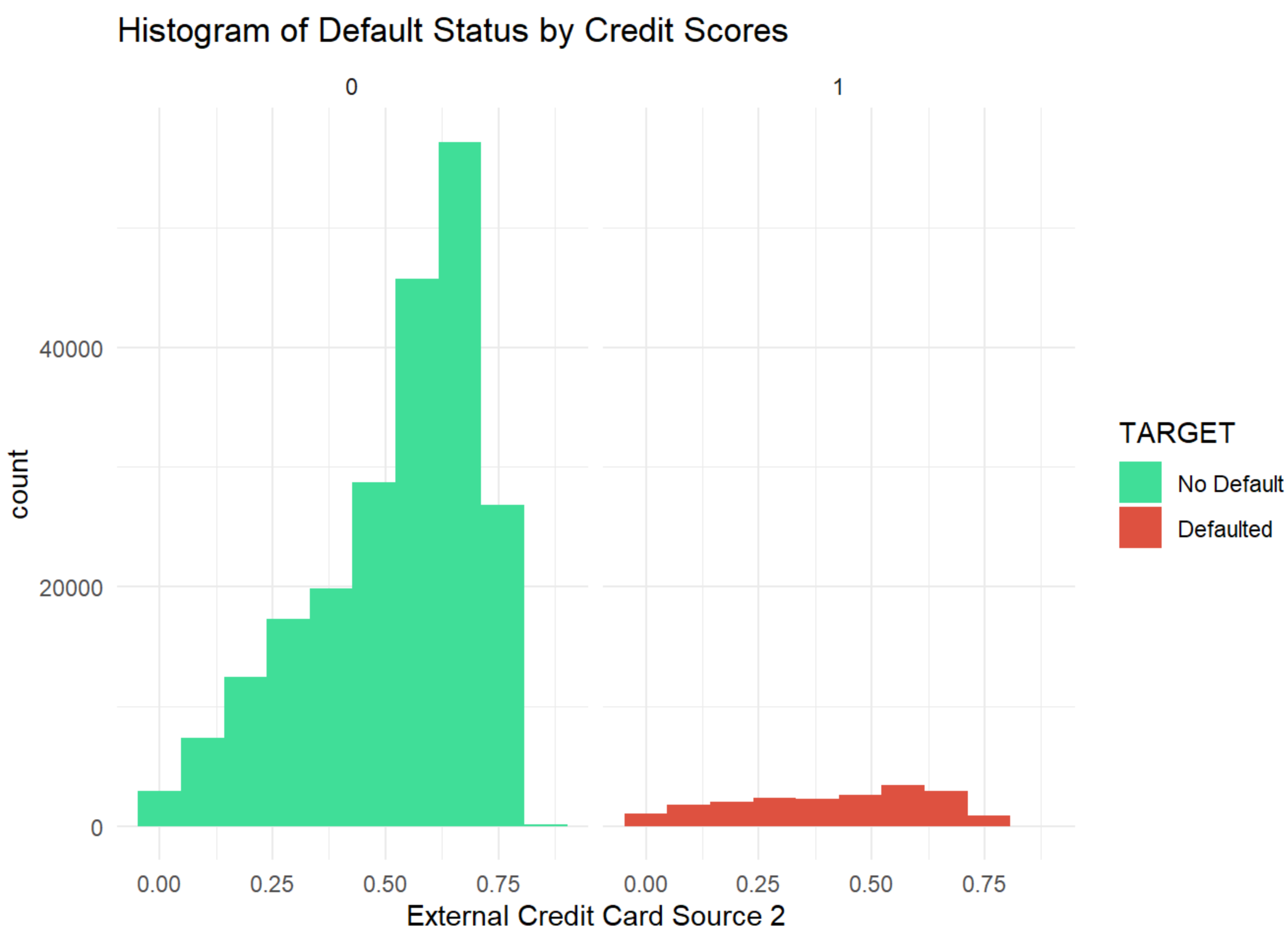
The logic as previously mentioned in the cleaning data section was that this Credit Card Agency was stricter with giving credit scores compared to the other credit card agencies. Thus, to reflect that strictness, zero was given to any customers who did not have a credit score.

Default Status by Credit Scores Source 2

Is there a relationship between the credit scores from Source 2 and a client's default status?

```
train_clean |> # call train clean and avoid calling it in ggplot
     ggplot(mapping = aes(x = EXT_SOURCE_2, fill = TARGET)) + # use Ext_Source_2 to graph plot
     geom_histogram(bins = 10) + # custom bins of 10
     facet_wrap(facets = ~TARGET) + # Create 2 histograms
     labs(title = "Histogram of Default Status by Credit Scores", # title
          x = "External Credit Card Source 2") + # X-axis
```

```
     scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                       labels = c("No Default", "Defaulted")) + # Custom labels
     theme_minimal() # Assign a theme
```

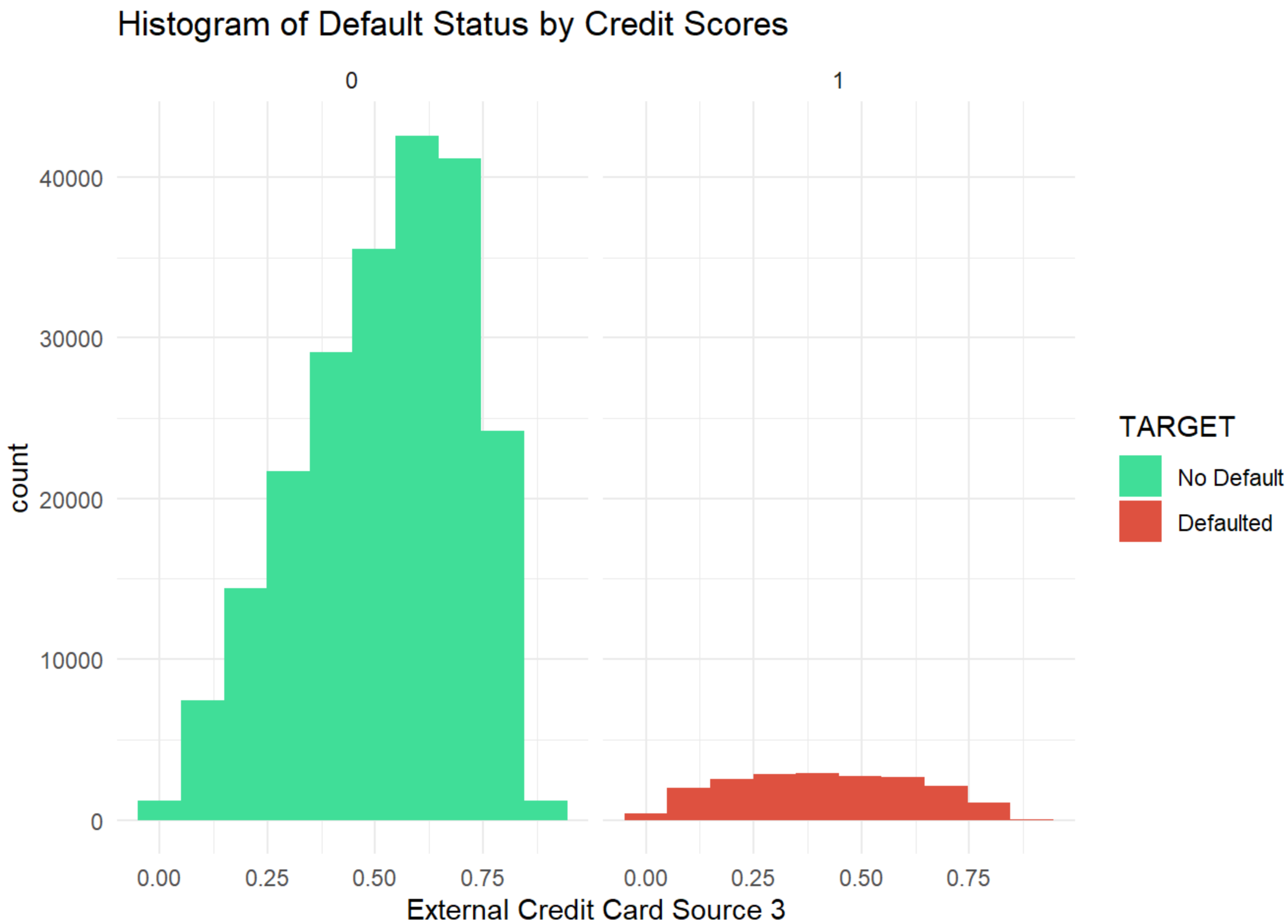


The plot shows a similar trend to the histogram for External Credit Card Source 1 in that there are more customers in each of credit score bins for customers who do not default compared to defaulting customers. This again seems to suggest that having a credit score in itself is indicative of default or no default. Furthermore this trend is much more pronounced in this plot with "no default" showing much higher counts of credit scores across all the bins when compared to the "defaulted" group.

Default Status by Credit Scores Source 3

Is there a relationship between the credit scores from Source 2 and a client's default status?

```
train_clean |> # call train clean and avoid calling it in ggplot
    ggplot(mapping = aes(x = EXT_SOURCE_3, fill = TARGET)) + #x-axis should be ext_source_3
    geom_histogram(bins = 10) + # custom bins of 10
    facet_wrap(facets = ~TARGET) + # Create 2 histograms
    labs(title = "Histogram of Default Status by Credit Scores", # custom title
         x = "External Credit Card Source 3") + # custom x-axis title
     scale_fill_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                       labels = c("No Default", "Defaulted")) + # Custom labels
     theme_minimal() # Assign a theme
```



The plot shows also shows the same trend to the histogram for External Credit Card Source 2 in that there are more customers in each of credit score bins for customers who do not default compared to defaulting customers. This combined with Credit Card Source 1 and 2 further suggests that having a credit score in itself is indicative of default or no default. Furthermore, just like the Credit Card Source 2 histogram, this trend is much more pronounced when compared to Credit Card Source 1. The plot with “no default” showing much higher counts of credit scores across all the bins when compared to the “defaulted” group.

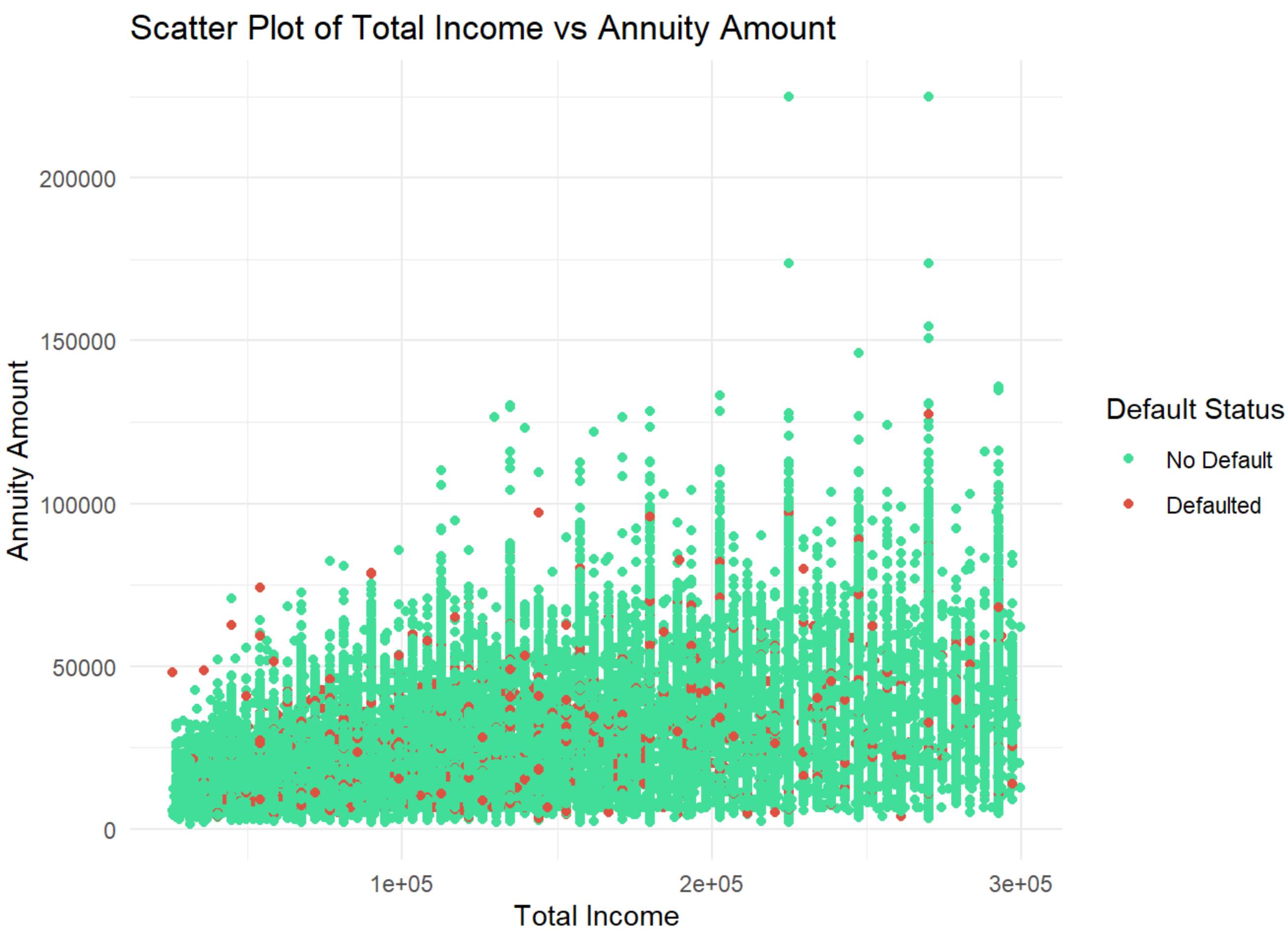
Scatterplots

Total Income vs Annuity Amount

Does Total Income have an influence on the Annuity Amount (Loan Amount) Perhaps clients with a higher income can apply for bigger annuity amounts. This in turn could have an impact on being able to pay back loans. Perhaps clients with higher Annuity Amounts default more then clients with lower Annuity Amounts.

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = AMT_INCOME_TOTAL, y = AMT_ANNUITY, color = as.factor(TARGET))) +  # Ma
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Total Income vs Annuity Amount", # custom title
       x = "Total Income", # custom x-axis
       y = "Annuity Amount", # custom y -axis
       color = "Default Status") +  # Add a label for the color legend
```

```
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Assign a theme
```



This scatter plot seeks to determine the influence of income on the annuity amount. It however appears that there is no relationship between the variables. There are multiple points in where the annuity amount continues to increase while income remains the same.

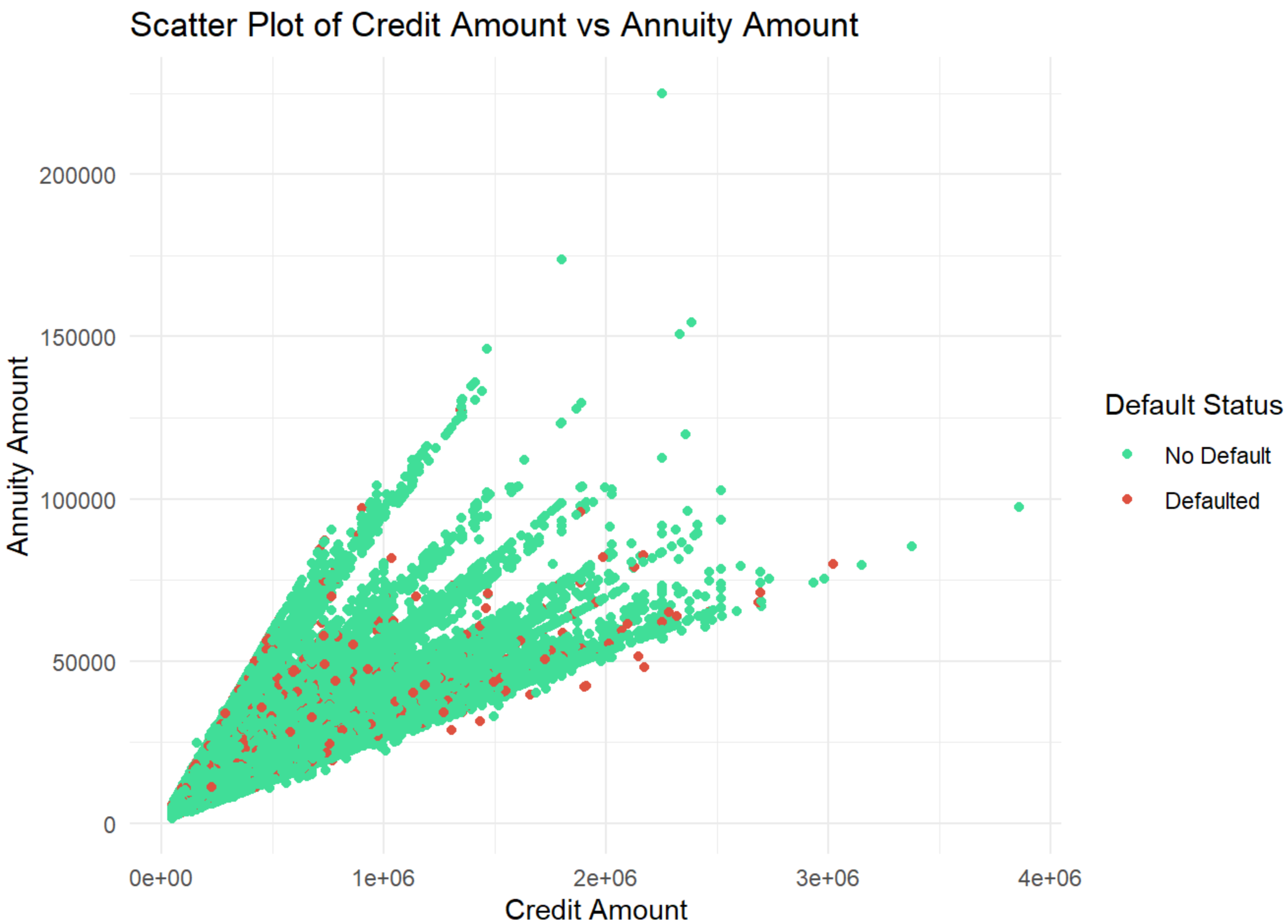
There is additionally no strong trends with "default" and "no default" for the scatter plot. The points are randomly scattered and don't suggest a strong pattern.

Credit Amount vs Annuity Amount

Does Credit Amount have an influence on the Annuity Amount. Perhaps, clients with a higher credit can apply for larger annuities. A visualization could be utilized to explore this relationship.

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = AMT_CREDIT, y = AMT_ANNUITY, color = as.factor(TARGET))) +  # Map TARGI
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Credit Amount vs Annuity Amount", # custom title
       x = "Credit Amount", # custom x-axis
       y = "Annuity Amount", # custom y-axis
```

```
     color = "Default Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Assign a theme
```



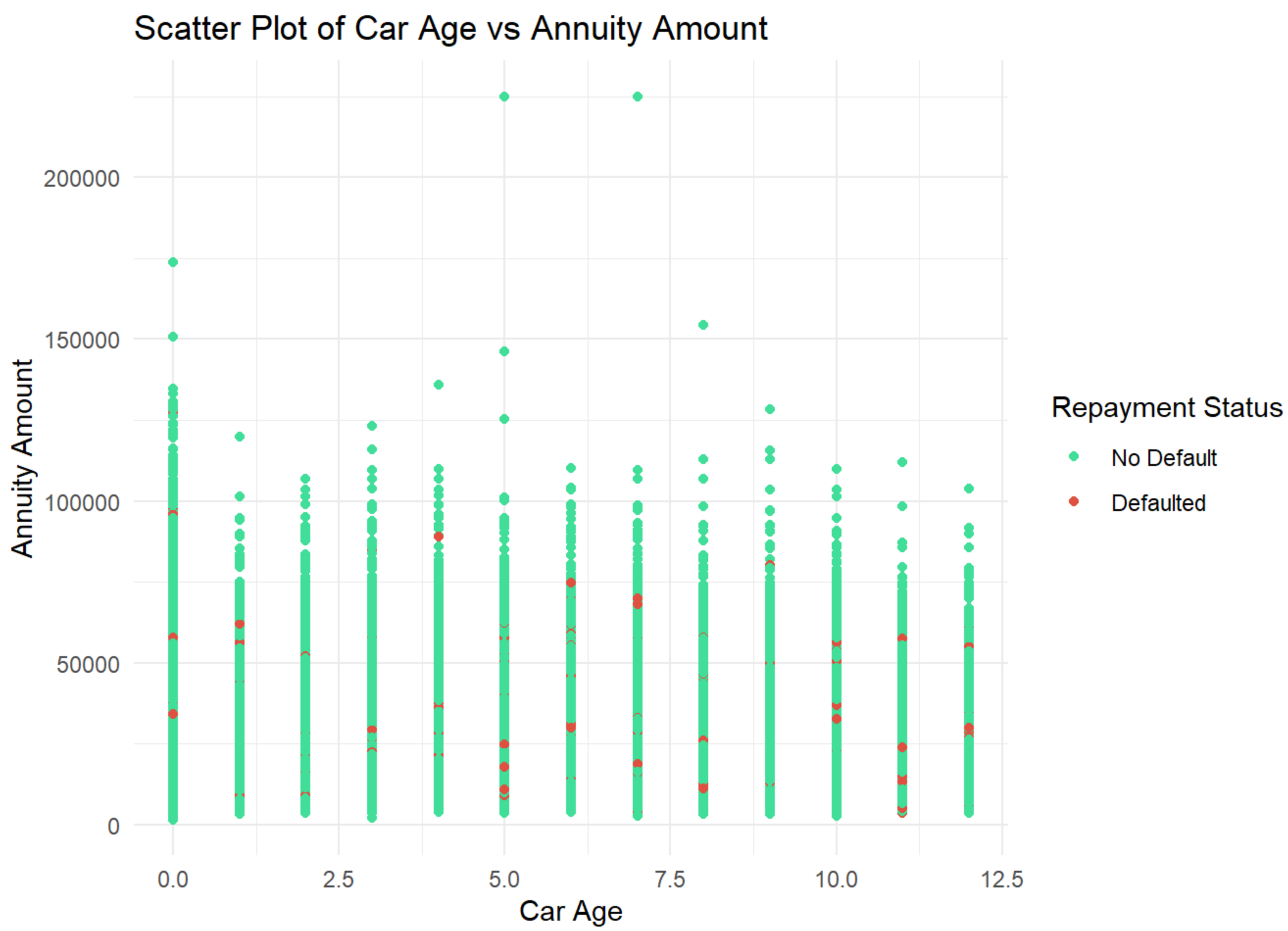
The scatter plot shows the influence of Credit Amount on the Annuity Amount which is a positive relationship. This is because the plot shows when Credit Amount increases, the Annuity Amount also increases as well. Furthermore, there seems to be a somewhat weak positive relationship with default and increasing credit amount. As the Credit Amount increases, "Defaulted" also increases which also corresponds to an increase in the Annuity Amount as well. (It should be noted that No Default also increases as well which indicates that default and no default may increase in general when credit amount goes up.)

Car Age vs Annuity Amount

Does the age of a client's car have any contribution since cars are typically used for collateral in loans? In the case of Annuity Amount, perhaps an older car will result in a lower annuity amount since older cars are worth less. A visualization could be utilized to explore this relationship.

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = OWN_CAR_AGE, y = AMT_ANNUITY, color = as.factor(TARGET))) +  # Map TAR(
  geom_point() +  # Scatter plot
```

```
  labs(title = "Scatter Plot of Car Age vs Annuity Amount", # custom title
       x = "Car Age", # custom x-axis
       y = "Annuity Amount", # custom y-axis
       color = "Repayment Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) + # Custom labels for the legend
  theme_minimal() # Assign a theme
```



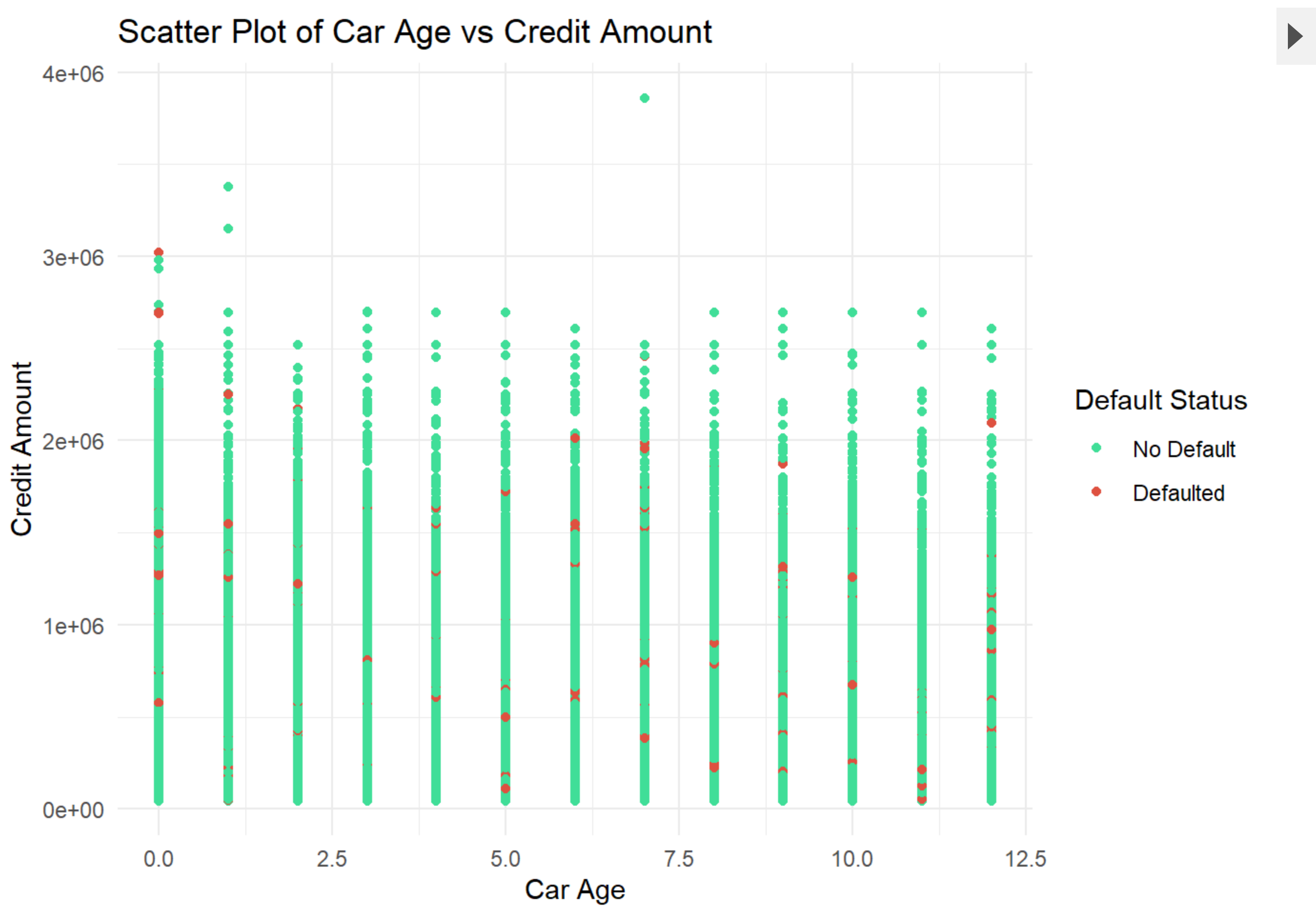
This scatterplot shows the relationship between Car Age and Annuity Amount. There is no relationship between the two variables. For instance, at age zero for the car, the annuity amount continues to increase. This can also be observed for every car age which indicates that there is no relationship. Additionally, there is no noticeable trend of grouping between "Defaulted" and "No Defaulted". "Defaulted" appears to be randomly scattered.

Car Age vs Credit Amount

Does the age of a client's car have any contribution since cars are typically used for collateral in loans? Perhaps older cars lead to a lower a credit since cars depreciate over time. Thus, their value as collateral diminishes overtime. A visualization could be utilized to explore this relationship.

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = OWN_CAR_AGE, y = AMT_CREDIT, color = as.factor(TARGET))) +  # Map TARGI
```

```r
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Car Age vs Credit Amount", # custom title
       x = "Car Age", # custom x-axis
       y = "Credit Amount", # custom y-axis
       color = "Default Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Custom Theme
```

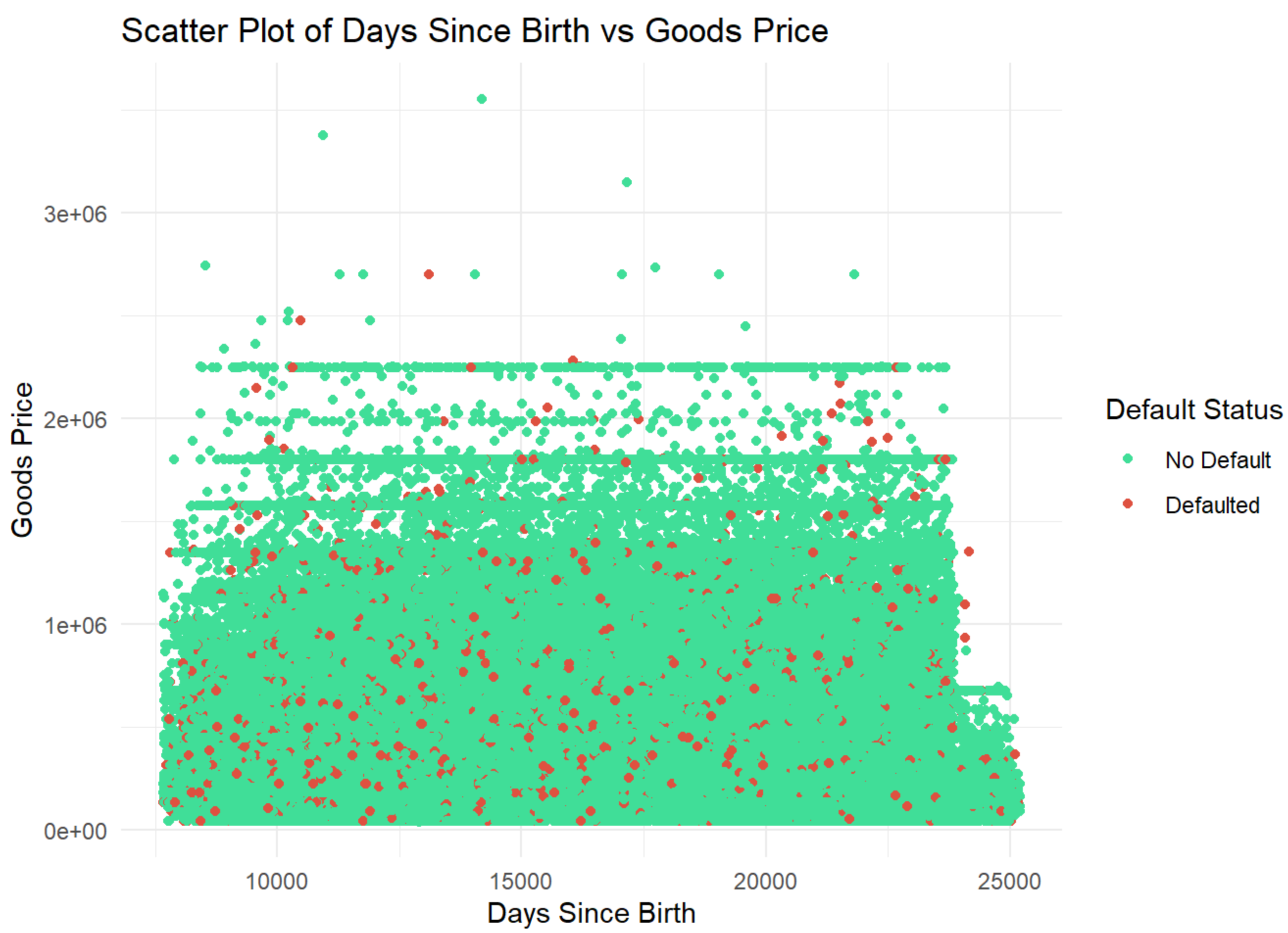


This scatterplot shows the relationship between Car Age and Credit Amount. There is no relationship between the two variables. For instance, at age zero for the car, the credit amount continues to increase. This can also be observed for every car age which indicates that there is no relationship. Additionally, there is no noticeable trend of grouping between "Defaulted" and "No Defaulted". "Defaulted" appears to be randomly scattered.

Days Since Birth vs Goods Price

Age could also play a role in default. (Days Since Birth represents Age.) Perhaps, younger clients default more because they are just starting out which means they won't have much assets or cash. This also means that the goods they seek to purchase may be smaller when compared to older clients. A visualization can be utilized to explore this relationship.

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = DAYS_BIRTH, y = AMT_GOODS_PRICE  , color = as.factor(TARGET))) +  # Ma
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Days Since Birth vs Goods Price", # custom title
       x = "Days Since Birth", # custom x-axis
       y = "Goods Price", # custom y-axis
       color = "Default Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Custom Theme
```

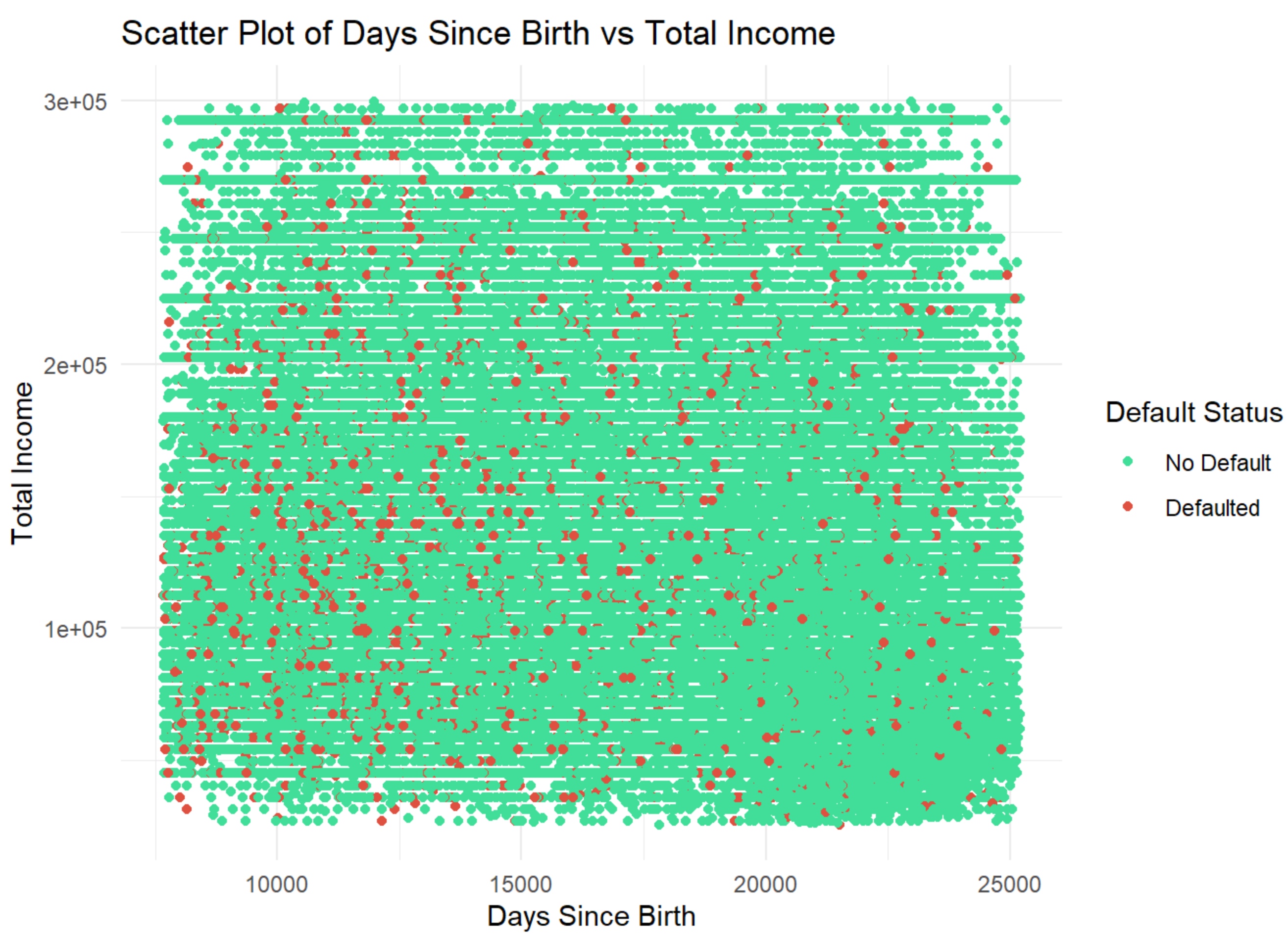


This scatterplot shows the relationship between Car Age and Credit Amount. There is no relationship between the two variables. For instance, at age zero for the car, the credit amount continues to increase. This can also be observed for every car age which indicates that there is no relationship. Additionally, there is no noticeable trend of grouping between "Defaulted" and "No Defaulted". "Defaulted" appears to be randomly scattered.

Days Since Birth vs Total Income

Age could also play a role in default. Perhaps younger clients default more because they are just starting out which means they won't have much assets or income. A visualization can be utilized to explore this relationship.

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = DAYS_BIRTH, y = AMT_INCOME_TOTAL  , color = as.factor(TARGET))) +  # Ma
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Days Since Birth vs Total Income", # custom title
       x = "Days Since Birth",  # custom x-axis
       y = "Total Income",  # custom y-axis
       color = "Default Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Custom Theme
```

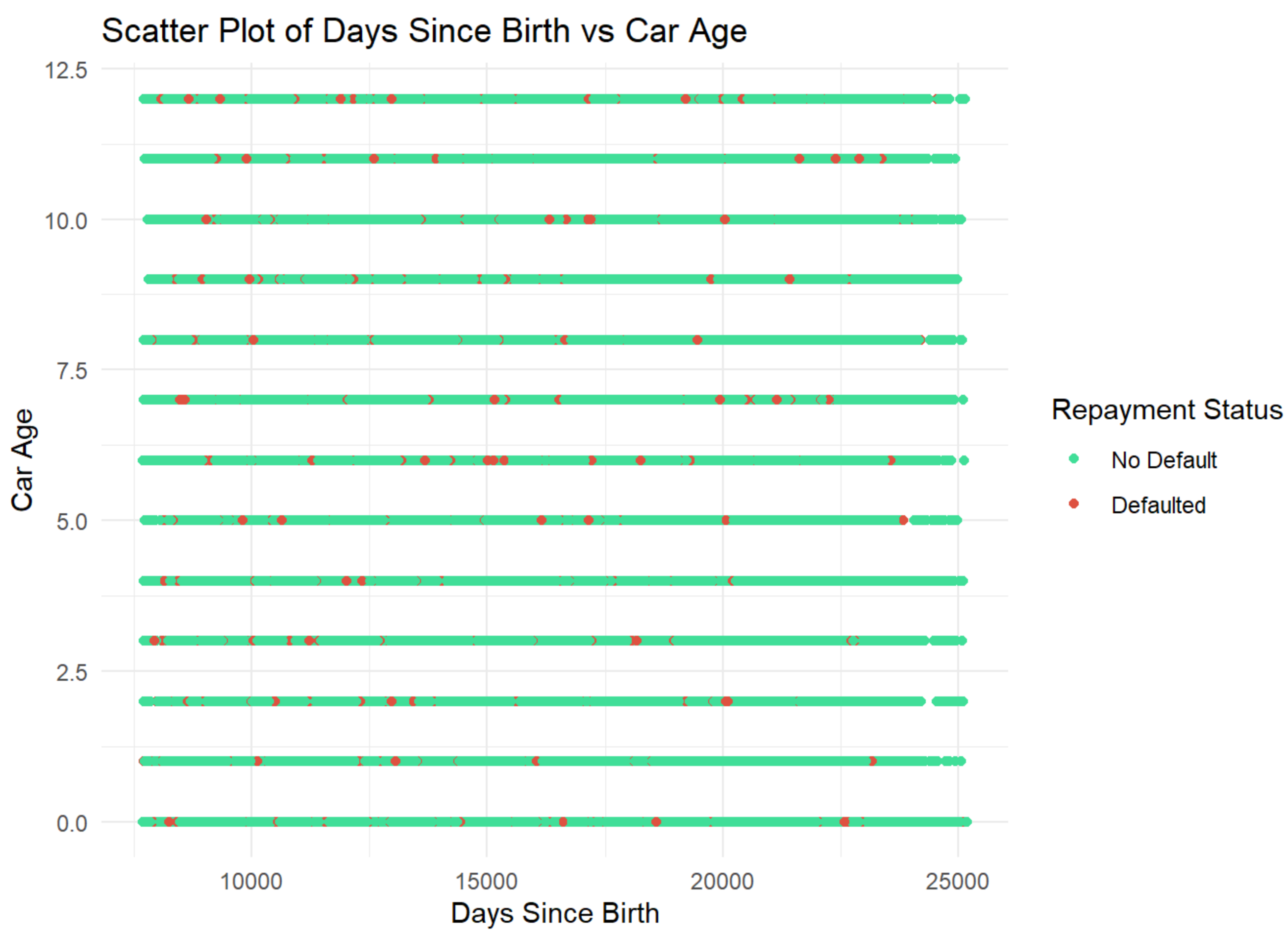


This scatterplot shows the relationship between Days Since Birth and Total Income. There however is no relationship between these two variables with the plot being densely scattered. Additionally, “Defaulted” is randomly scattered with no apparent pattern.

Days Since Birth vs Car Age

Age could also play a role in default. Perhaps younger clients default more because they are just starting out which means they won’t have much assets or cash. This also means that the goods they seek to purchase may be smaller when compared to older clients. A visualization can be utilized to explore this relationship.

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = DAYS_BIRTH , y = OWN_CAR_AGE, color = as.factor(TARGET))) +  # Map TAR(
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Days Since Birth vs Car Age", # custom title
       x = "Days Since Birth", # custom x-axis
       y = "Car Age", # custom y-axis
       color = "Repayment Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Custom Theme
```



This scatterplot shows the relationship between Days Since Birth and Car Age. There is no relationship between the two variables. For instance, at 10,000 Days Since Birth, there is no change in the Car Age. Additionally, there is no noticeable trend or grouping between “Defaulted” and “No Defaulted”. “Defaulted” appears to be randomly scattered.

Highly Correlated Variables

```
# Step 1: Select numeric columns excluding 'TARGET'
numeric_columns <- sapply(train_clean, is.numeric)
```

```r
train_clean_numeric <- train_clean[, numeric_columns]

# Step 2: Exclude the 'TARGET' column
train_clean_numeric <- subset(train_clean_numeric)

# Step 3: Calculate correlation matrix (use complete observations)
correlation_matrix <- cor(train_clean_numeric, use = "complete.obs")

# Step 4: Convert correlation matrix to a long format
cor_long <- as.data.frame(as.table(correlation_matrix))

# Step 5: Filter out self-correlations (correlations where a variable is correlated with itself)
cor_long <- cor_long[cor_long$Var1 != cor_long$Var2, ]

# Step 6: Remove redundant pairs (keep only one combination of each variable pair)
cor_long <- cor_long[!duplicated(t(apply(cor_long[, 1:2], 1, sort))), ]

# Step 7: Sort by the absolute value of correlations
cor_long <- cor_long[order(abs(cor_long$Freq), decreasing = TRUE), ]

# Step 8: Extract top 10 positive correlations
top_10_positive <- head(cor_long[cor_long$Freq > 0, ], 10)

# Step 9: Extract top 10 negative correlations
top_10_negative <- head(cor_long[cor_long$Freq < 0, ], 10)

# Step 10: View the top 10 positive and top 10 negative correlations
print("Top 10 Positive Correlations:")
```

```
[1] "Top 10 Positive Correlations:"
```

```r
print(top_10_positive)
```

```
                           Var1                     Var2      Freq
1017     OBS_60_CNT_SOCIAL_CIRCLE OBS_30_CNT_SOCIAL_CIRCLE 0.9979763
120               AMT_GOODS_PRICE               AMT_CREDIT 0.9858663
1056     DEF_60_CNT_SOCIAL_CIRCLE DEF_30_CNT_SOCIAL_CIRCLE 0.8593348
50                CNT_FAM_MEMBERS             CNT_CHILDREN 0.8477911
158               AMT_GOODS_PRICE              AMT_ANNUITY 0.7783780
119                   AMT_ANNUITY               AMT_CREDIT 0.7769875
744                FLOORSMAX_MEDI           ELEVATORS_MEDI 0.5134187
633               LIVINGAREA_MEDI          APARTMENTS_MEDI 0.4878360
646  House_Attribute_Low_Variance          APARTMENTS_MEDI 0.4758115
950  House_Attribute_Low_Variance          LIVINGAREA_MEDI 0.4696326
```

```r
print("Top 10 Negative Correlations:")
```

```
[1] "Top 10 Negative Correlations:"
```

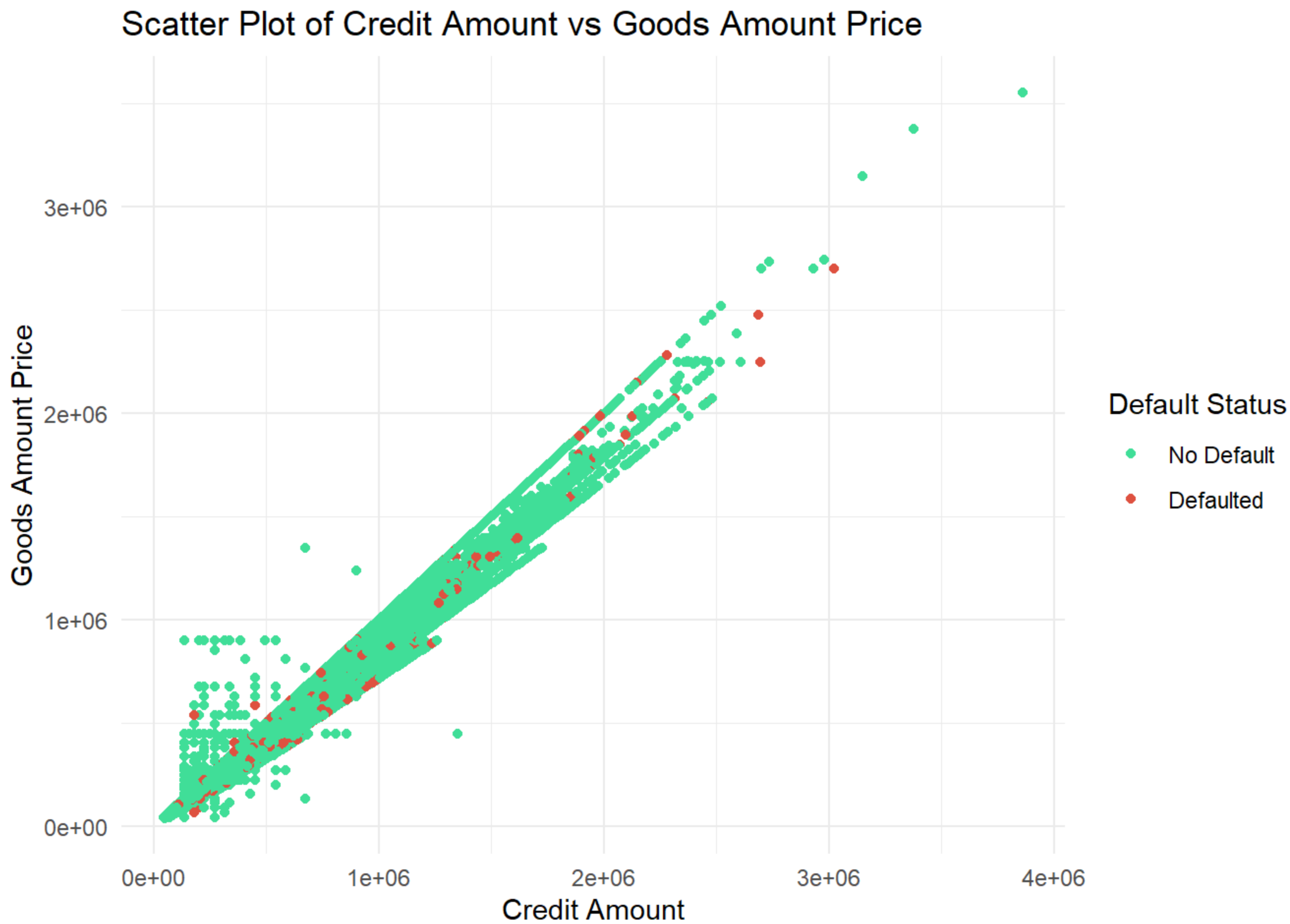
```
print(top_10_negative)
```

```
                       Var1             Var2        Freq
46               DAYS_BIRTH      CNT_CHILDREN -0.35211960
278          CNT_FAM_MEMBERS        DAYS_BIRTH -0.29059512
47         DAYS_REGISTRATION      CNT_CHILDREN -0.18453909
316          CNT_FAM_MEMBERS DAYS_REGISTRATION -0.17053525
277              OWN_CAR_AGE        DAYS_BIRTH -0.10623049
279 HOUR_APPR_PROCESS_START        DAYS_BIRTH -0.10251348
84               DAYS_BIRTH  AMT_INCOME_TOTAL -0.09024576
280              EXT_SOURCE_1        DAYS_BIRTH -0.06848200
315              OWN_CAR_AGE DAYS_REGISTRATION -0.06774107
85         DAYS_REGISTRATION  AMT_INCOME_TOTAL -0.06508720
```

Several of the variables that I thought would have strong relationships for the scatterplots show the opposite. Thus, a correlation matrix will be created and the top 10 positive and negative correlations will be selected. Afterwards visualizations will be created to analyze the variables that I believe are the most interesting.

Credit Amount vs Goods Amount Price

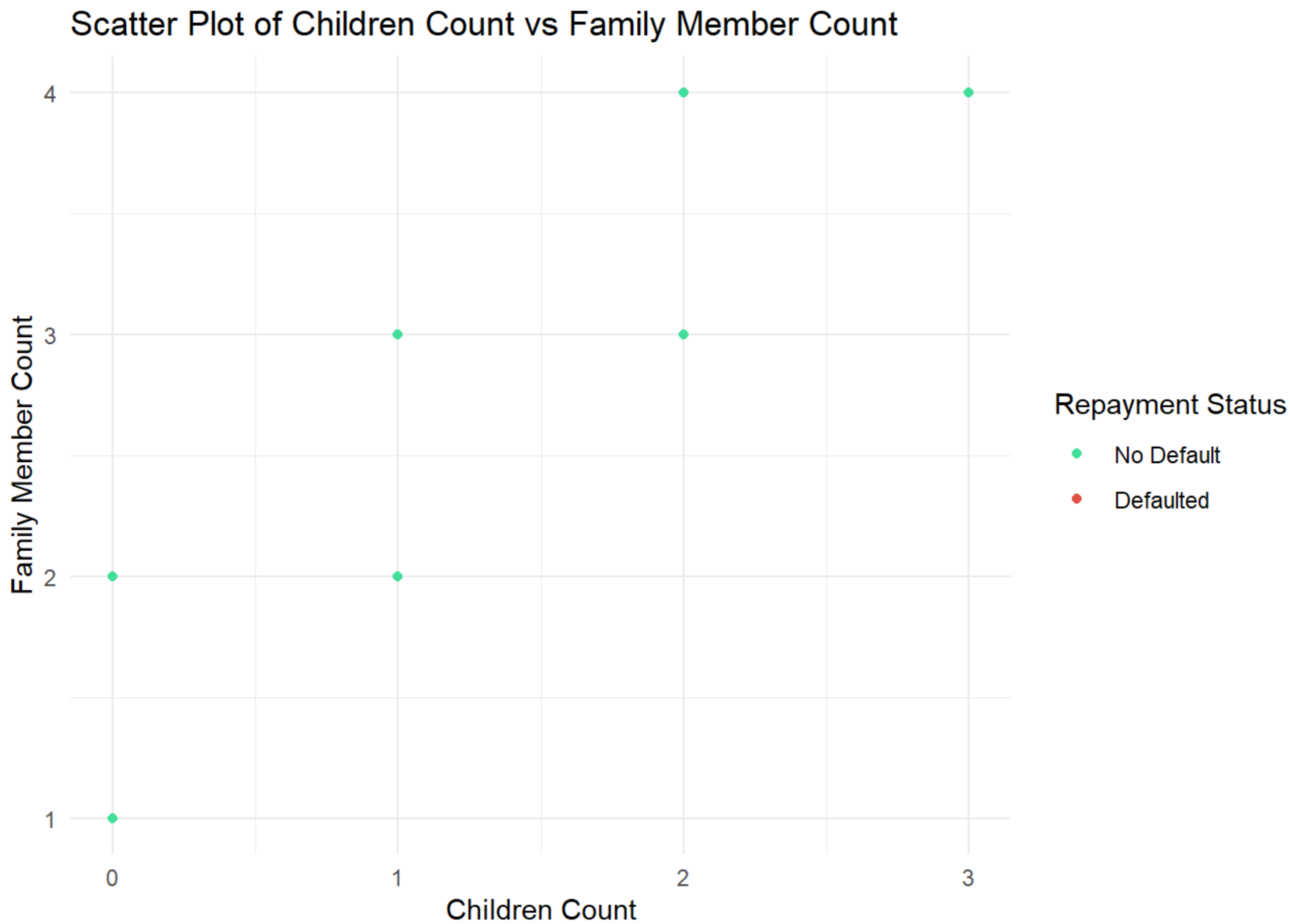
```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = AMT_CREDIT , y = AMT_GOODS_PRICE  , color = as.factor(TARGET))) +  # Ma
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Credit Amount vs Goods Amount Price", # custom title
       x = "Credit Amount", # custom x-axis title
       y = "Goods Amount Price", # custom y-axis title
       color = "Default Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Assign a theme
```



This Scatter Plot shows the relationship between Credit Amount and the Goods Amount Price. There is a strong positive relationship between Credit Amount and Goods Amount Price. Additionally both "No Default" and "Defaulted" are increasing as well. There however does not appear to be a distinct area of the scatter plot where the "Default" points and clustered together and the "No Default" points are clustered together.

Children Count vs Family Count

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = CNT_CHILDREN , y = CNT_FAM_MEMBERS  , color = as.factor(TARGET))) +  #
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Children Count vs Family Member Count", # custom title
       x = "Children Count", # custom x-axis title
       y = "Family Member Count", # custom y-axis title
       color = "Repayment Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Assign a theme
```



This Scatter Plot shows the relationship between Children Count and the Family Member Count. There is a somewhat positive relationship between Children Count and the Family Member Count. Interestingly only "No Default" is present in the graph. This suggests that perhaps clients who have children mostly don't default.

The table shows that there are a small number of clients who have children and do default on their loans.

Goods Price Amount vs Annuity Amount

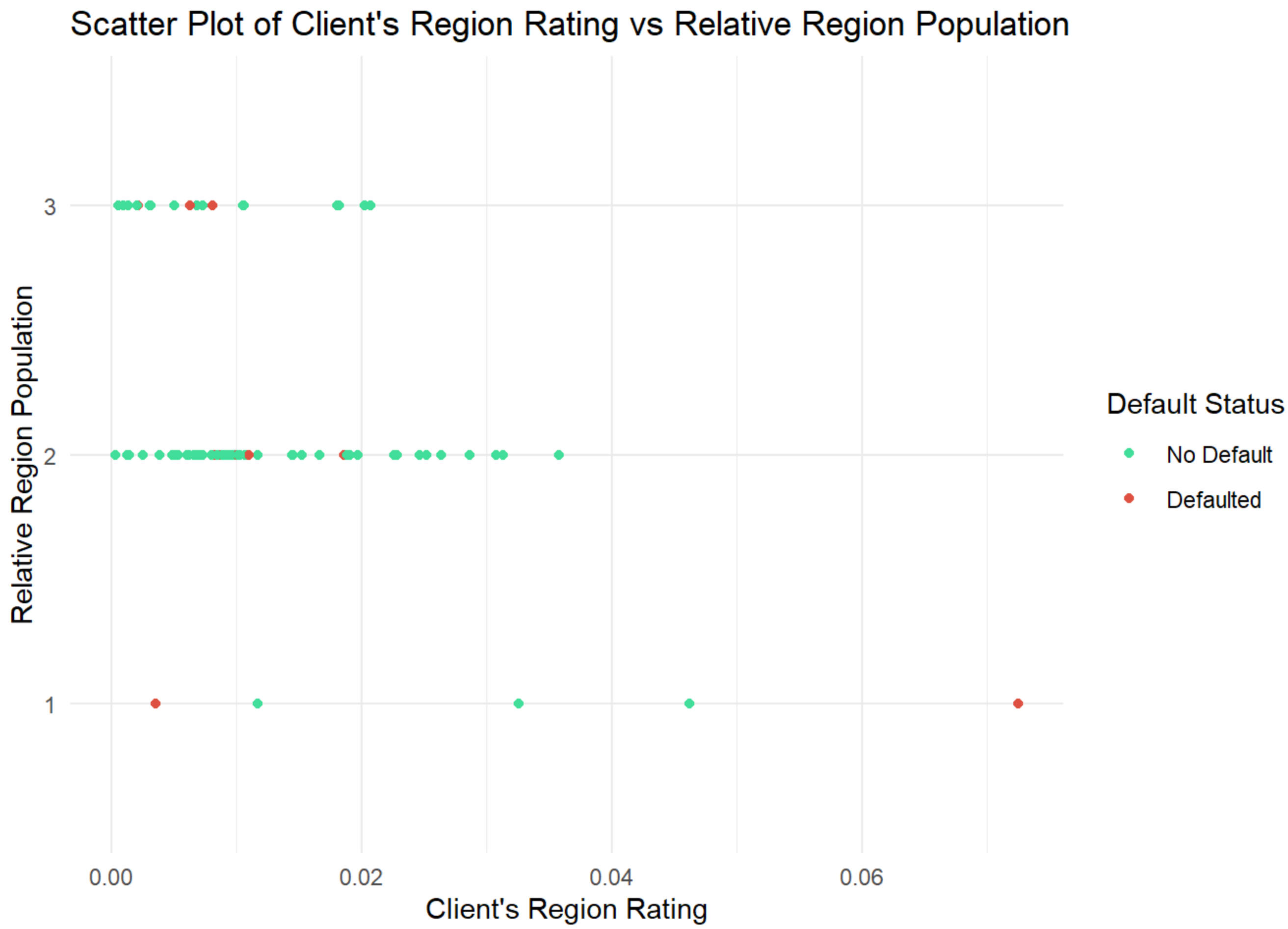
```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = AMT_GOODS_PRICE , y = AMT_ANNUITY  , color = as.factor(TARGET))) +  # I
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Goods Amount Price vs Annuity Amount", # custom title
       x = "Goods Amount Price", # custom x-axis title
       y = "Annuity Amount", # custom y-axis title
       color = "Default Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Assign a theme
```

This scatter plot shows the relationship between the Goods Amount Price and Annuity Amount. There is a positive relationship between these two variables with Annuity Amount increasing as Goods Amount Price goes up. Additionally there does appear to be a cluster of "Defaulted" from 500,000 to 1,000,000 dollars for Goods Amount Price. This seems to suggest that a lot of defaults may occur when the client is attempting to purchase an item that is worth roughly 500k to 1M.

Afterwards, the there is not a really distinct group and the rest of the default observations somewhat increase as the Goods Amount Price increases.

Client's Region Rating vs Relative Region Population

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = REGION_POPULATION_RELATIVE , y = REGION_RATING_CLIENT, color = as.facto
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Client's Region Rating vs Relative Region Population", # custom t
       x = "Client's Region Rating", # custom x-axis title
       y = "Relative Region Population", # custom y-axis title
       color = "Default Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Assign a theme
```

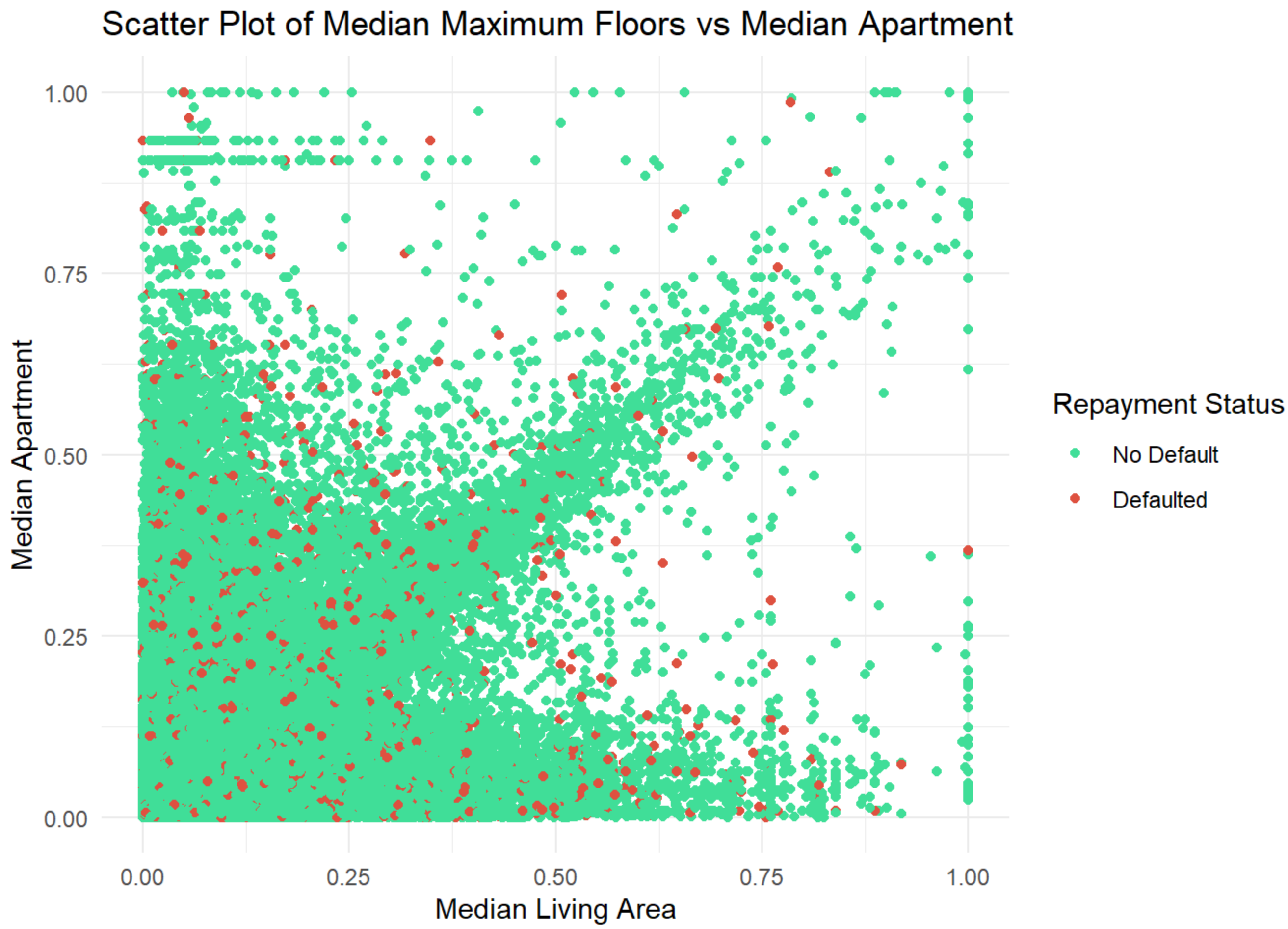


The correlation between these two variables is -.50. It however appears that there may not be any relationship based on this scatterplot. For instance, the Relative Region Population (normalized population of a client's region) has the same score even though the Client's Region Rating changes. (Client's Region Rating represents the score that Home Credit has given to a client's region.)

Additionally there is no distinct grouping or trend between "No Default" and "Defaulted".

Median Living Area vs Median Living Area

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = LIVINGAREA_MEDI , y = APARTMENTS_MEDI  , color = as.factor(TARGET))) +
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Median Maximum Floors vs Median Apartment", # custom title
       x = "Median Living Area", # custom x-axis title
       y = "Median Apartment", # custom y-axis title
       color = "Repayment Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Assign a theme
```



Several of the housing variables like LIVINGAREA_MEDI and APARTMENTS_MEDI are shown to be somewhat correlated at 0.49. It however seems unlikely that all these variables would have a strong effect on predicting default. Thus, only of 2 of the correlated variables will be selected for experimentation and to see if there is any relationship with default.

Thus, only Median Living Are and Median Apartment have been selected to see if there is a relationship due to relatively high correlation at .49. (I am not selecting FLOORSMAX_MEDI/ELEVATORS_MEDI which has a correlation of 0.5134187. This is because I believe that most clients are unlikely to have elevators in their houses which makes this variable combination unrepresentative of the population.)

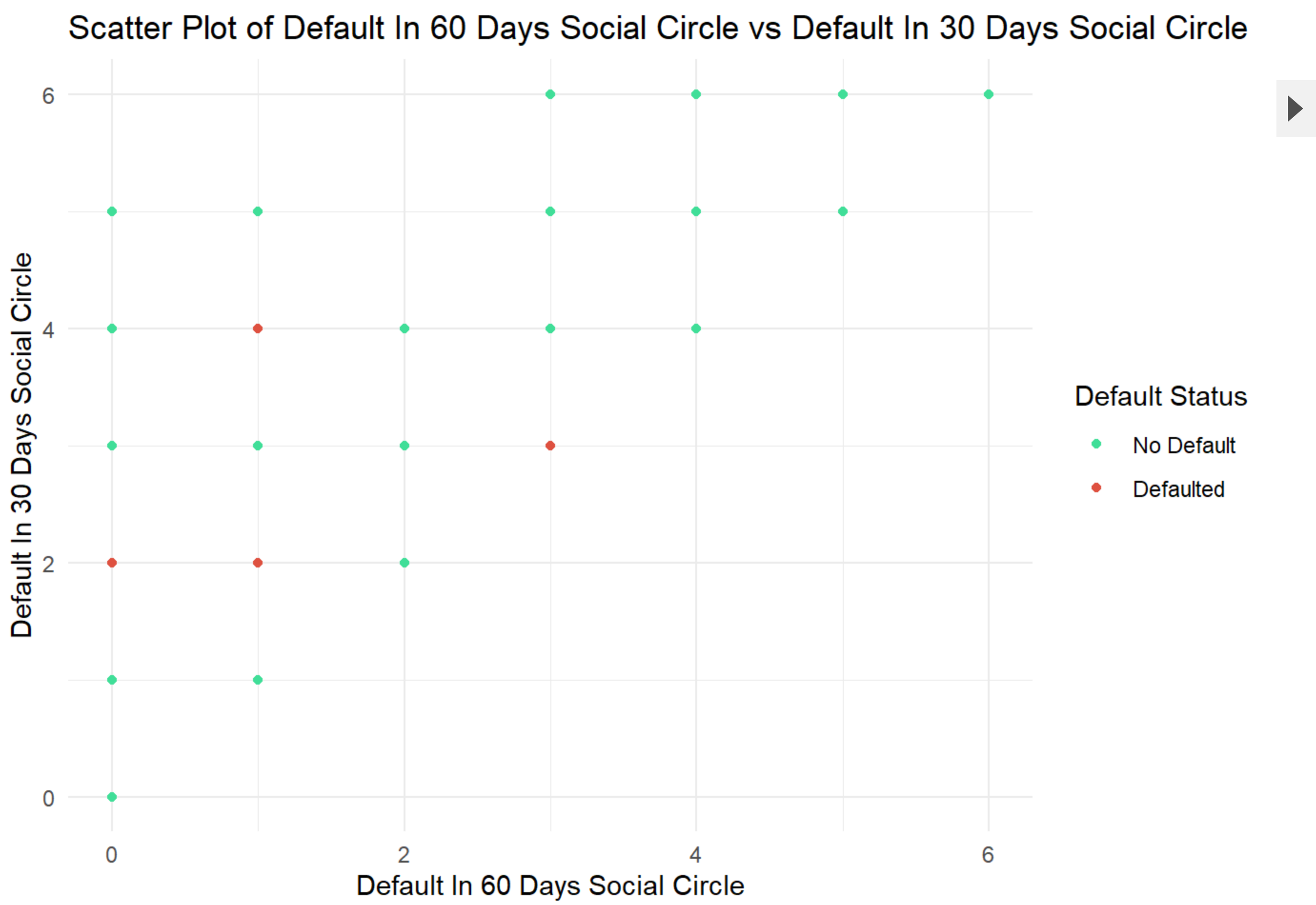
Finally, for the overall plot, it appears that there is somewhat moderate positive relationship with the Median Living Area associated with an increase in Median Apartments. (This mainly occurs when Median Living Area is approximately .50)

Default also seems to have a very weak cluster from a Median Living Area of 0 to 0.25. Beyond that, defaulted appears to be randomly scattered.

Default In 60 Days Social Circle vs Default In 30 Days Social Circle

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = DEF_60_CNT_SOCIAL_CIRCLE , y = DEF_30_CNT_SOCIAL_CIRCLE  , color = as.
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Default In 60 Days Social Circle vs Default In 30 Days Social Cir
```

```
    x = "Default In 60 Days Social Circle", # custom x-axis title
    y = "Default In 30 Days Social Circle", # custom y-axis title
    color = "Default Status") +  # Add a label for the color legend
  scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                     labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
  theme_minimal() # Assign a theme
```

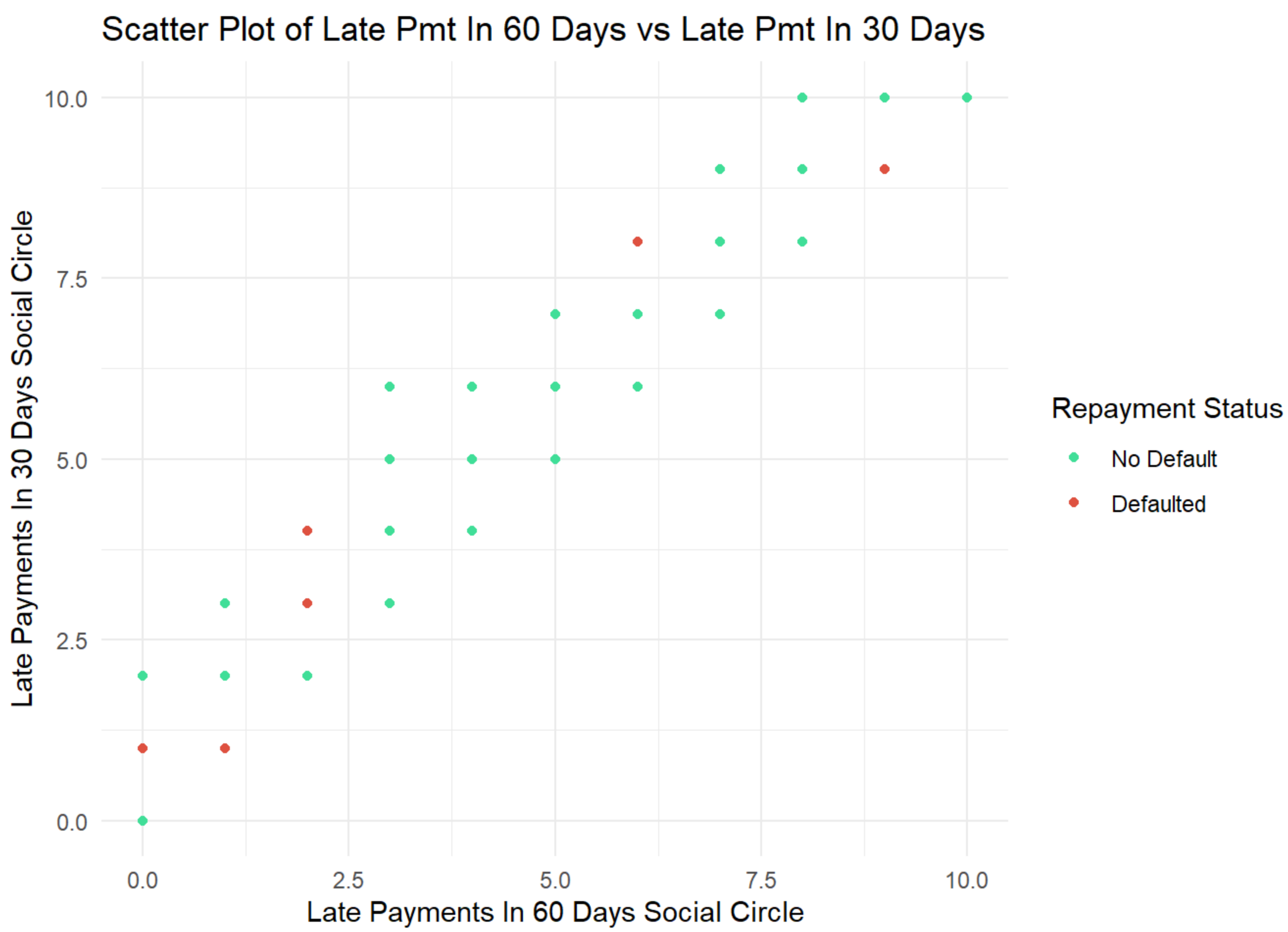


This is a plot that compares the relationship between Default in 60 days (How many people in the Client's social circle defaulted on their loans within 60 days.) and Default in 30 days. (How many people in the Client's social circle defaulted on their loans within 30 days.)

There does appear to be a positive relationship between the two variables with an increase in Defaulting in 60 days corresponding to an increase in defaulting in 30 days. No Default follows this pattern, however Defaulted seems to have no strong pattern. For instance, there are 0 clients who defaulted in 60 days but 2 clients who did default in 30 days.

```
train_clean |> # call train clean and avoid calling it in ggplot
  ggplot(mapping = aes(x = OBS_60_CNT_SOCIAL_CIRCLE , y = OBS_30_CNT_SOCIAL_CIRCLE  , color = as.
  geom_point() +  # Scatter plot
  labs(title = "Scatter Plot of Late Pmt In 60 Days vs Late Pmt In 30 Days", # custom title
    x = "Late Payments In 60 Days Social Circle", # custom x-axis title
    y = "Late Payments In 30 Days Social Circle", # custom y-axis title
```

```
    color = "Repayment Status") +  # Add a label for the color legend
scale_color_manual(values = c("0" = "#40DE98", "1" = "#DE5140"),   # Assign custom colors
                   labels = c("No Default", "Defaulted")) +  # Custom labels for the legend
theme_minimal() # Assign a theme
```



This scatterplot shows a similar relationship between late payments (60 days past due) and late payments (30 days past due). The relationship is positive. There however does not seem to be a strong pattern for defaulted. For instance, even though late payments (60 days) changes from 0 to 1.25, late payment still remains the same at 1.25. This can be seen for other observations as well such as 2.3 for late payments (60 days) where late payment 60 remains at 2.3 while late payment(30 days) increases from 3.75 to 3.8

Results

The EDA was very interesting and revealed a lot of things. For instance, I thought that some variables would have strong relationships but the relationship was not very strong. Here are some of those variables:

- Birth vs Car Age
- Birth vs Total Income
- Birth vs Goods Price
- Total Income vs Annuity Amount

- Martial Status

I was particularly surprised that there was not a strong relationship between Total Income and Annuity Amount. Martial Status also surprised me as well.

There were other variables that did have an influence and provided some insights into potential causes of default.

- Credit Amount vs Goods Price
- Goods Amount Price vs Annuity Amount
- The External Credit Scores
- Gender
- Income
- Education

I believe that it should now be possible to build a model that is parsimonious using the above predictors to better predict default. Additionally there were several missing values that had to be addressed carefully with imputation depending on the context. Some columns like Car Age were imputed to zero because it meant that the owner did not own a car while other columns like the external credit scores used values from both columns to make educated guesses about the values. Moreover low variance columns also needed to be addressed, since they don't offer any predictive value and may slow down modelling.

Finally, after completing the EDA, it is very apparent that most clients pay their loans since 0 represents successful loan payback which is 91% of the data. This can also be seen in indirectly in every scatterplot visualization. Thus, a better metric like "Precision" or "F1 Score" should be used in place of Accuracy to gain a more nuanced insight into Home Credit's operations and to evaluate any new models.