Title Section

Home Credit Default Rate

Group 13

- Kalyani Malokar
- Krisha Mehta
- Kunal Mehra
- William Cutchin

INFO-I-526: Applications of Machine Learning

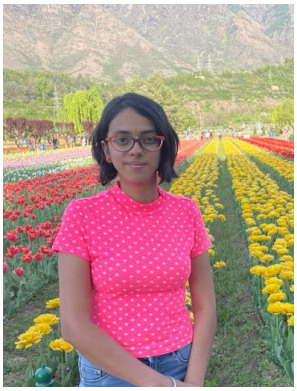
Indiana University Bloomington, Luddy School of Informatics, Computing, and Engineering

Professor Dr. James Shanahan

Date: April 11, 2023

Tierra Mallorca on Unsplash

Team - Introduction

Group Member	Member Picture	Group Member	Member Picture
Kalyani Malokar (kmalokar@iu.edu)		Krisha Mehta (krimeht@iu.edu)	

Group Member	Member Picture	Group Member	Member Picture
Kunal Mehra (kumehra@iu.edu)		William Cutchin (wcutchin@iu.edu)	

Team - Progress & Planning

Phase Leadership Plan

Phase Number	Team Member	Phase Objective Delegation
Phase 1: Project Proposal	William Cutchin (Phase Leader)	• Schedule Meetings, Organize Tasks, Lead Group Meetings • Format Project Proposal
Phase 1: Project Proposal	Krisha Mehta	• Data Description
Phase 1: Project Proposal	Group	• Machine Algorithms and Metrics
Phase 1: Project Proposal	Kalyani Malokar	• Machine Learning Pipeline (Diagram) • Gantt Chart of Tasks
Phase 1: Project Proposal	Kunal Mehra	• Machine Learning Pipeline Steps & Descriptions • Additional Algorithms (Loss Functions)
Phase 2: EDA & Basic Pipelines	Krisha Mehta (Phase Leader)	• Schedule Meetings, Organize Tasks, Lead Group Meetings • Data Retrieval
Phase 2: EDA & Basic Pipelines	Kalyani Malokar	• Feature Engineering (Round 1)

Phase 2: EDA & Basic Pipelines	Kunal Mehra	• Hyper Parameter Tuning (Round 1)
Phase 2: EDA & Basic Pipelines	William Cutchin	• Exploratory Data Analysis • Video Presentation
Phase 3: Feature Engineering & Hyperparameter Tuning	Kalyani Malokar (Phase Leader)	• Schedule Meetings, Organize Tasks, Lead Group Meetings • Feature Selection
Phase 3: Feature Engineering & Hyperparameter Tuning	Kunal Mehra	• Hyper Parameter Tuning (Round 2)
Phase 3: Feature Engineering & Hyperparameter Tuning	William Cutchin	• Feature Engineering (Round 2)
Phase 3: Feature Engineering & Hyperparameter Tuning	Krisha Mehta	• Video Presentation • Ensemble Methods
Phase 4: Final Submission	Kunal Mehra (Phase Leader)	• Neural Network Implementation • Schedule Meetings, Organize Tasks, Lead Group Meetings
Phase 4: Final Submission	William Cutchin	• Final Report • Video Presentation
Phase 4: Final Submission	Krisha Mehta	• Advanced Model Architectures
Phase 4: Final Submission	Kalyani Malokar	• Advanced Loss & Additional Functions

Credit Assignment Plan

Phase 1

Task	Task Description	Assigned Member	Estimated Hours	Actual Hours	Start Date	Completion Date

Format Project Proposal	Communicate to find group members' desired tasks, write the abstract, and collect and display Team Photos.	William Cutchin	5	5.5	28/3/2023	4/4/2023
Data Description	Create table figures of data sources with descriptions	Krisha Mehta	1	1	29/3/2023	4/4/2023
Machine Algorithms and Metrics	Research and select appropriate metrics and algorithms for the datasets.	Group	1.5	2	04/03/2023	04/04/2023
Machine Learning Pipeline (Diagram)	Construct a block diagram which visualizes the suggested pipeline steps.	Kalyani Malokar	1.5	2	03/31/2023	04/03/2023
Gantt Chart of Tasks	Construct a Gantt chart which displays the waterfall of tasks and their dependencies.	Kalyani Malokar	1	1	04/04/2023	04/04/2023
Machine Learning Pipeline Steps & Descriptions	Describe and reason through the steps the pipeline will take.	Kunal Mehra	2	2	03/28/2023	03/30/2023
Additional Algorithms (Loss Functions)	Select reasonable loss functions, describe them, and display their formula	Kunal Mehra	0.5	0.5	03/31/2023	04/04/2023

Phase 2

Task	Task Description	Assigned Member	Estimated Hours	Actual Hours	Start Date	Completion Date
Data Retrieval & Preprocessing	Retrieve data from the Kaggle API and begin loading the data and pre-processing.	Krisha Mehta	3	4	04/04/2023	04/05/2023
Feature Engineering (Round 1)	Develop and deploy initial feature engineering, applying statistical techniques and log experiments.	Kalyani Malokar	6	5.5	04/05/2023	04/06/2023
Machine Pipelines & Baseline Experimentation	Test ranges of parameters for the given features, record experiment results and optimize.	Kunal Mehra	6	5.5	04/05/2023	04/07/2023
Exploratory Data Analysis & Visual Analysis	Handle missing values, perform descriptive analysis, and identify correlations.	William Cutchin	4.5	6	04/07/2023	04/11/2023

Video Presentation	Summarize the project, describe work completed, layout plans for the future, and discuss blockers.	William Cutchin	2	4	04/10/2023	04/11/2023

Phase 3

Task	Task Description	Assigned Member	Estimated Hours	Actual Hours	Start Date	Completion Date
Feature Selection	Observe and compare results from Feature engineering and decide which features are worth exploring further.	Kalyani Malokar	7	TBD	04/11/2023	04/12/2023
Hyper Parameter Tuning (Round 2)	Test ranges of parameters for the given features that have been selected by the feature selection step.	Kunal Mehra	6	TBD	04/13/2023	04/14/2023
Feature Engineering (Round 2)	Develop and deploy feature engineering on new selected features, log experiments and explore adding or removing features. Log these experiments.	William Cutchin	5	TBD	04/14/2023	04/17/2023
Ensemble Methods	Combine the multiple models or pipelines used into a single process. Log the results and compare.	Krisha Mehta	3.5	TBD	04/14/2023	04/18/2023
Video Presentation	Summarize the project, describe work completed, layout plans for the future, and discuss blockers.	Krisha Mehta	2	TBD	04/14/2023	04/18/2023

Phase 4

Task	Task Description	Assigned Member	Estimated Hours	Actual Hours	Start Date	Completion Date
Neural Network Implementation	Develop and deploy an effective neural network, given the reasonings of previous ML algorithms. Test and log all experiments.	Kunal Mehra	8	TBD	04/18/2023	04/20/2023
Advanced Model Architectures	Combine and understand previous models to construct an effective and advanced model.	Krisha Mehta	4.5	TBD	04/21/2023	04/25/2023

Advanced Loss & Additional Functions	Continue to iterate and experiment with loss functions, optimizing further the model's performance.	Kalyani Malokar	4	TBD	04/21/2023	04/25/2023
Final Report Formatting	Accumulate all information and insight to be formatted into an attractive and logical report.	William Cutchin	8	TBD	04/18/2023	04/25/2023
Video Presentation	Summarize the project, describe work completed, report our successes and process, and describe how we could build on our submission.	William Cutchin	3	TBD	04/21/2023	04/25/2023
Final Report	Compile and discuss all progress, development, visualizations, and findings to present to peers with great efficiency.	All Members	25	TBD	04/24/2023	04/25/2023

Project Abstract

The problem that has been provided is the Home Credit Default Risk. In this problem, the machine learning team is looking to create algorithms and pipelines that will predict which individual will have successful repayment on their loan without a traditional credit score. To solve this problem, we employed many methods throughout the process of solving this problem. First, we used EDA and Visual EDA to find the most effective features in predicting loan repayment or default. Then, using these features and the data provided, we employed Logistic Regression, RandomForest, and DecisionTree algorithms to see which best predicted the target value, measured by the Area Under the Curve . We found that Logistic Regression was the most fruitful, yielding a test Area Under the Curve score of 74.38%. This was in comparison to the other algorithms which differed between 5 to 10% below the AUC score of the Logistic Regression.

Project Description

Project Description: Data - Import & Organize Data

In [ ]:
```
!pip install -q kaggle
```

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle (4).json
```

In [ ]:
```
!mkdir original_data
!mkdir original_zip
```

```
mkdir: cannot create directory ‘original_data’: File exists
mkdir: cannot create directory ‘original_zip’: File exists
```

In [ ]:
```
import os
os.environ["KAGGLE_CONFIG_DIR"] = '/content'
```

In [ ]:
```
!kaggle competitions download -c home-credit-default-risk -p /content/original_zip
```

```
home-credit-default-risk.zip: Skipping, found more recently modified local copy (use
--force to force download)
```

In [ ]:
```
! chmod 600 /content/kaggle.json
```

In [ ]:
```
!unzip original_zip/home-credit-default-risk.zip
```

```
Archive:  original_zip/home-credit-default-risk.zip
  inflating: HomeCredit_columns_description.csv
  inflating: POS_CASH_balance.csv
  inflating: application_test.csv
  inflating: application_train.csv
  inflating: bureau.csv
  inflating: bureau_balance.csv
  inflating: credit_card_balance.csv
  inflating: installments_payments.csv
  inflating: previous_application.csv
  inflating: sample_submission.csv
```

In [ ]:
```
# Move all of the original data files from the content directory to the original data
# This will help us seperate and organize concerns
!mv HomeCredit_columns_description.csv original_data/
!mv POS_CASH_balance.csv original_data/
!mv application_test.csv original_data/
!mv application_train.csv original_data/
!mv bureau.csv original_data/
!mv bureau_balance.csv original_data/
!mv credit_card_balance.csv original_data/
!mv installments_payments.csv original_data/
!mv previous_application.csv original_data/
!mv sample_submission.csv original_data/
```

In [ ]:
```
# Import numpy
import numpy as np
import pandas as pd

# Read each of the CSV files and sensibly name them in a pandas dataframe
```

```
df_app_train = pd.read_csv('original_data/application_train.csv')
df_app_test = pd.read_csv('original_data/application_test.csv')
df_bureau = pd.read_csv('original_data/bureau.csv')
df_bureau_bal = pd.read_csv('original_data/bureau_balance.csv')
df_pos_cash_bal = pd.read_csv('original_data/POS_CASH_balance.csv')
df_credit_card_bal = pd.read_csv('original_data/credit_card_balance.csv')
df_pre_app = pd.read_csv('original_data/previous_application.csv')
df_installments_payments = pd.read_csv('original_data/installments_payments.csv')

### Misc Data Frames
# df_sample_sub = pd.read_csv('original_data/sample_submission.csv') ## Need more ram
# df_home_credit_descr = pd.read_csv('original_data/HomeCredit_columns_description.cs
```

Project Description: Data Description

Data Description: application_train.csv

DATA DESCRIPTION: application_train.csv </br> This data table is the primary training data for the HCDR problem. Each of the columns holds some data about the loan applicatant. For each row there is one loan application and the unique applicant is identified by the SK_ID_CURR. This table also holds the target values 0 and 1, where 1 represents that the loan was not repaid and 0 means that the loan was sucessfully repaid.

In [ ]:

```
# Summary - application_train.csv
print("Number of Rows: " + str(df_app_train.shape[0]) + "\n" + "Number of Columns: "
print("Number of Missing Values: " + str(df_app_train.isna().sum().sum()))

df_app_train.head(10)
```

```
Number of Rows: 307511
Number of Columns: 122
Number of Missing Values: 9152465
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALT
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	
5	100008	0	Cash loans	M	N	
6	100009	0	Cash loans	F	Y	
7	100010	0	Cash loans	M	Y	
8	100011	0	Cash loans	F	N	
9	100012	0	Revolving loans	M	N	

10 rows × 122 columns

Data Description: application_test.csv

DATA DESCRIPTION: application_test.csv </br> This table is the test file for our algorithms to be run on and have a predicted target score. This table does not contain all of the same data as the train set, but they have the same features and do not include the target value. This will be used later to predict over for the submission scores of the problem.

In [ ]:

```
# Summary - application_test.csv
print("Number of Rows: " + str(df_app_test.shape[0]) + "\n" + "Number of Columns: " +
print("Number of Missing Values: " + str(df_app_test.isna().sum().sum()))

df_app_test.head(10)
```

```
Number of Rows: 48744
Number of Columns: 121
Number of Missing Values: 1404419
```

Out[ ]:

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_C
0	100001	Cash loans	F	N	Y	
1	100005	Cash loans	M	N	Y	
2	100013	Cash loans	M	Y	Y	
3	100028	Cash loans	F	N	Y	
4	100038	Cash loans	M	Y	N	
5	100042	Cash loans	F	Y	Y	
6	100057	Cash loans	M	Y	Y	
7	100065	Cash loans	M	N	Y	
8	100066	Cash loans	F	N	Y	
9	100067	Cash loans	F	Y	Y	

10 rows × 121 columns

Data Description: bureau.csv

DATA DESCRIPTION: bureau.csv </br> This dataset contains all of the data of the loan applicatant that has been provided from previous financial instituitons. These credits have thier own row and have the same unique identifier SK_ID_CURR and another identifier SK_ID_BUREAU. This will show all active credit, thier balances, and if they are overdue, among other information.

In [ ]:

```
# Summary - bureau.csv
print("Number of Rows: " + str(df_bureau.shape[0]) + "\n" + "Number of Columns: " + s
print("Number of Missing Values: " + str(df_bureau.isna().sum().sum()))

df_bureau.head(10)
```

```
Number of Rows: 1716428
Number of Columns: 17
Number of Missing Values: 3939947
```

Out[ ]:

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_DAY_OVER
0	215354	5714462	Closed	currency 1	-497	
1	215354	5714463	Active	currency 1	-208	
2	215354	5714464	Active	currency 1	-203	
3	215354	5714465	Active	currency 1	-203	
4	215354	5714466	Active	currency 1	-629	
5	215354	5714467	Active	currency 1	-273	
6	215354	5714468	Active	currency 1	-43	
7	162297	5714469	Closed	currency 1	-1896	
8	162297	5714470	Closed	currency 1	-1146	
9	162297	5714471	Active	currency 1	-1146	

Data Description: bureau_balance.csv

DATA DESCRIPTION: bureau_balance.csv </br> This data is similar to the bureau table, but it gives monthly previous credits of the bureau. Each of the monthly credits is a new row in the table and shares the same credit identifier SK_ID_BUREAU. This table only shows a breif summary of the credit showing closed, open, and the monthly balance.

In [ ]:

```
# Summary - bureau_balance.csv
print("Number of Rows: " + str(df_bureau_bal.shape[0]) + "\n" + "Number of Columns: "
print("Number of Missing Values: " + str(df_bureau_bal.isna().sum().sum()))

df_bureau_bal.head(10)
```

```
Number of Rows: 27299925
Number of Columns: 3
Number of Missing Values: 0
```

Out[ ]:

	SK_ID_BUREAU	MONTHS_BALANCE	STATUS
0	5715448	0	C
1	5715448	-1	C
2	5715448	-2	C
3	5715448	-3	C
4	5715448	-4	C
5	5715448	-5	C
6	5715448	-6	C
7	5715448	-7	C
8	5715448	-8	C
9	5715448	-9	0

Data Description: credit_card_balance.csv

DATA DESCRIPTION: credit_card_balance.csv </br> This table shows the monthly data about previously held credit cards. This data is linked to the other tables with the SK_ID_PREV and SK_ID_CURR identifiers. Amongst this data is the current balance, their credit imits, and withdraws from the account.

In [ ]:

```
# Summary - credit_card_balance.csv
print("Number of Rows: " + str(df_credit_card_bal.shape[0]) + "\n" + "Number of Colum
print("Number of Missing Values: " + str(df_credit_card_bal.isna().sum().sum()))

df_credit_card_bal.head(10)
```

```
Number of Rows: 3840312
Number of Columns: 23
Number of Missing Values: 5877356
```

Out[ ]:

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL	AMT_D
0	2562384	378907	-6	56.970	135000	
1	2582071	363914	-1	63975.555	45000	
2	1740877	371185	-7	31815.225	450000	
3	1389973	337855	-4	236572.110	225000	
4	1891521	126868	-1	453919.455	450000	
5	2646502	380010	-7	82903.815	270000	
6	1079071	171320	-6	353451.645	585000	
7	2095912	118650	-7	47962.125	45000	
8	2181852	367360	-4	291543.075	292500	
9	1235299	203885	-5	201261.195	225000	

10 rows × 23 columns

Data Description: installments_payments.csv

DATA DESCRIPTION: installment_payments.csv </br> This data set shows previous installment payments on loans at the Home Credit Company, which is the company being aided by this data exploration and machine learning. These data are identified by the SK_ID_PREV and SK_ID_CURR identifiers. This data shows specifically the installment payment amount, the amount paid, the version of the installment payment, and more.

In [ ]:

```
# Summary - installments_payments.csv
print("Number of Rows: " + str(df_installments_payments.shape[0]) + "\n" + "Number of
print("Number of Missing Values: " + str(df_installments_payments.isna().sum().sum())

df_installments_payments.head(10)
```

```
Number of Rows: 13605401
Number of Columns: 8
Number of Missing Values: 5810
```

Out[ ]:

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS_INSTAL
0	1054186	161674	1.0	6	-
1	1330831	151639	0.0	34	-
2	2085231	193053	2.0	1	
3	2452527	199697	1.0	3	-
4	2714724	167756	1.0	2	-
5	1137312	164489	1.0	12	-
6	2234264	184693	4.0	11	
7	1818599	111420	2.0	4	
8	2723183	112102	0.0	14	
9	1413990	109741	1.0	4	

Data Description: previous_application.csv

DATA DESCRIPTION: previous_application.csv </br> The previous_application data set shows applications that have been provided to Home Credit, the company requesting service. These data display the type of loan, the amount loaned, the payments on that loan, and more data around this topic. They are linked to the currently open loans through their SK_ID_PREV and SK_ID_CURR identifiers.

In [ ]:

```
# Summary - previous_application.csv
print("Number of Rows: " + str(df_pre_app.shape[0]) + "\n" + "Number of Columns: " +
print("Number of Missing Values: " + str(df_pre_app.isna().sum().sum()))

df_pre_app.head(10)
```

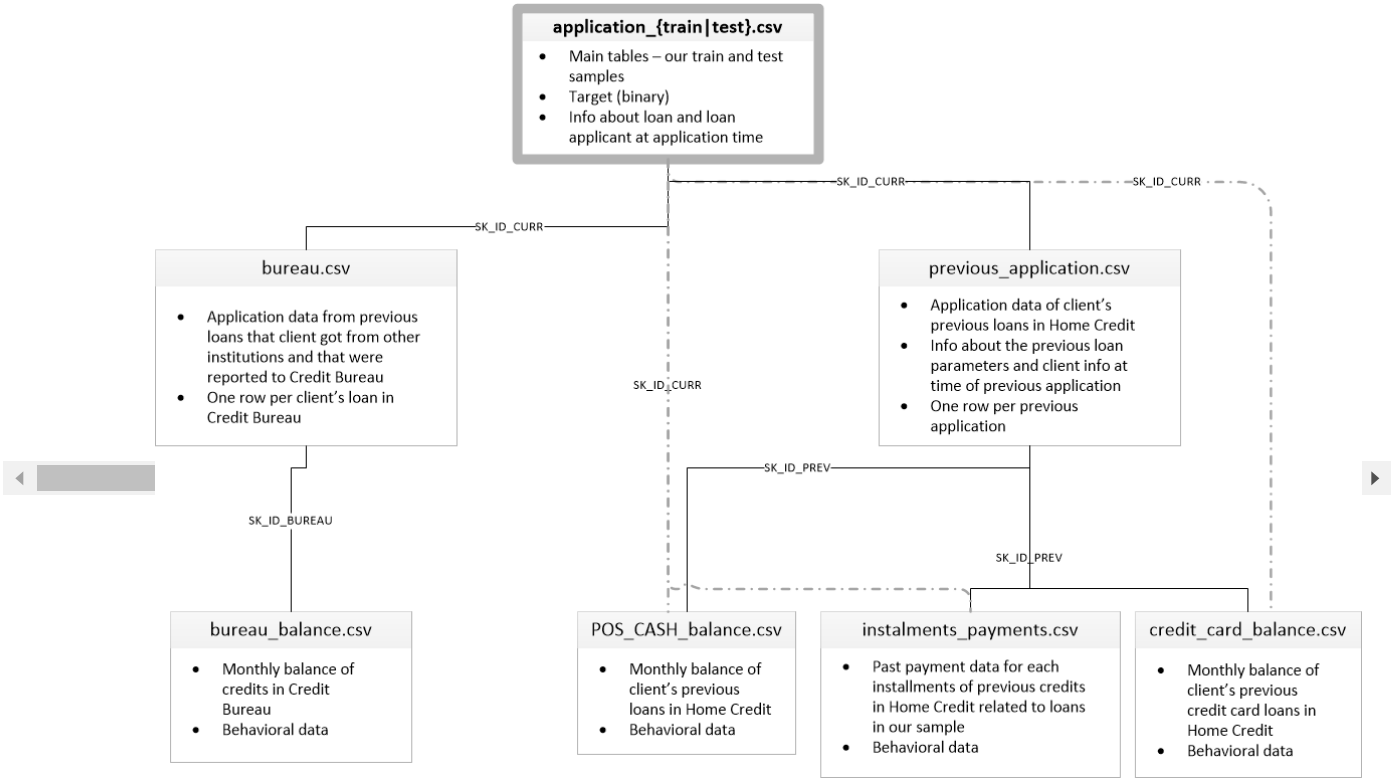
```
Number of Rows: 1670214
Number of Columns: 37
Number of Missing Values: 11109336
```

Out[ ]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDI
0	2030495	271877	Consumer loans	1730.430	17145.0	17145
1	2802425	108129	Cash loans	25188.615	607500.0	679671
2	2523466	122040	Cash loans	15060.735	112500.0	136444
3	2819243	176158	Cash loans	47041.335	450000.0	470790
4	1784265	202054	Cash loans	31924.395	337500.0	404055
5	1383531	199383	Cash loans	23703.930	315000.0	340573
6	2315218	175704	Cash loans	NaN	0.0	0
7	1656711	296299	Cash loans	NaN	0.0	0
8	2367563	342292	Cash loans	NaN	0.0	0
9	2579447	334349	Cash loans	NaN	0.0	0

10 rows × 37 columns

Project Description: Visualization



DESCRIPTION </br> In the above figure we can see each of the previously described data sets and their relationship through identifier key to the application_test|train.csv data sets. This is useful in understanding how to handle these data and how they should be cleaned and preprocessed in latter experiments.

Project Description: Tasks

Here is a breif Description of the tasks at hand for this current phase

Phase 2

Task	Task Description	Assigned Member	Estimated Hours	Actual Hours	Start Date	Completion Date
Data Retrieval	Retrieve data from the Kaggle API and begin loading the data and pre-processing.	Krisha Mehta	5	6	04/04/2023	04/05/2023
Feature Engineering (Round 1)	Develop and deploy initial feature engineering, applying statistical techniques and log experiments.	Kalyani Malokar	6	5.5	04/05/2023	04/06/2023
Hyper Parameter Tuning (Round 1)	Test ranges of parameters for the given features, record experiment results and optimize.	Kunal Mehra	6	5.5	04/05/2023	04/07/2023
Exploratory Data Analysis	Handle missing values, perform descriptive analysis, and identify correlations.	William Cutchin	4.5	7	04/07/2023	04/11/2023
Video Presentation	Summarize the project, describe work completed, layout plans for the future, and discuss blockers.	William Cutchin	2	4	04/10/2023	04/11/2023

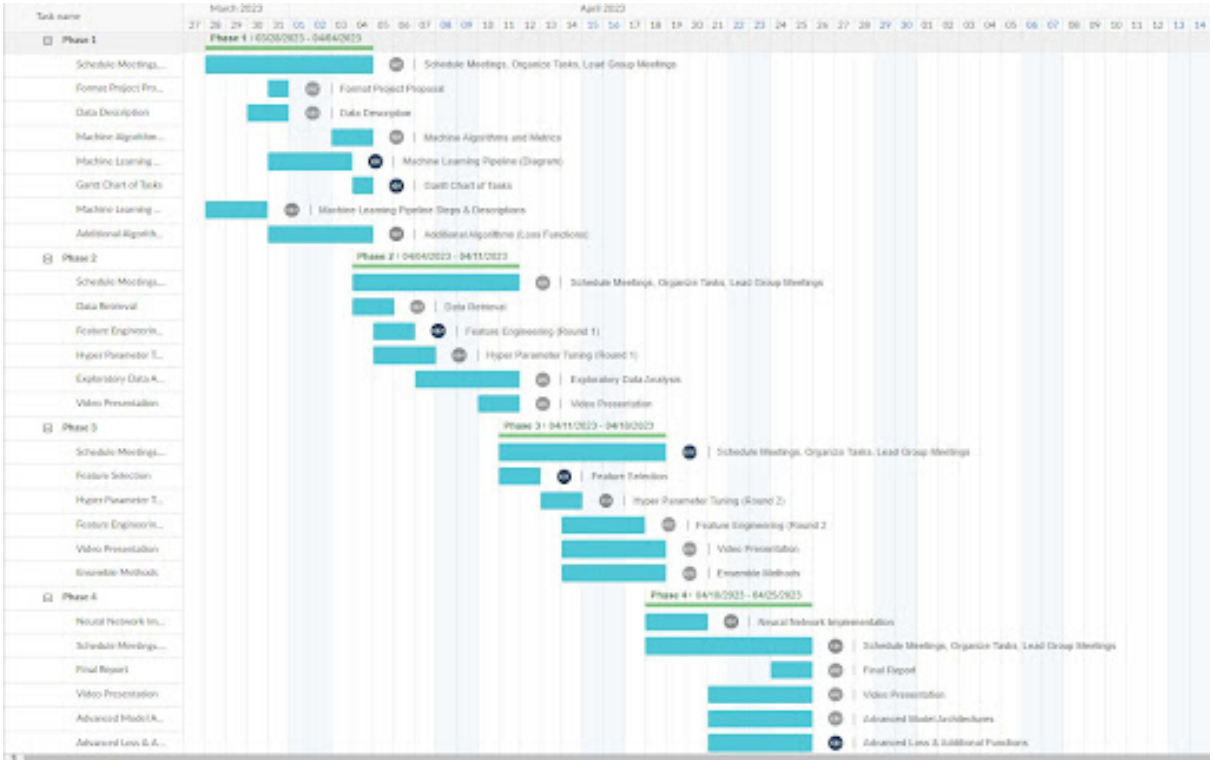
Here is a more detailed description, these tasks will still have their credit assigned to the same member but with more detail

- Data Retrieval
  - Using the kaggle api import the HCDR data and set up data frames for each of the given tables
  - Use these data frames to preform data descriptions for each of the tables
  - Organize the data into sensible folders within the development enviornment
- Exploritory Data Analysis
  - Using the data frames created by the Data Retrieval, on each table run a set of commands that will show the summary statistics, the size and shape of these data, some information about missing values, and any other important artifacts of the data as deemed important
  - Visulize these data
    - Do a full analysis on the application_train.csv data table showing correlation, distributions, how they interact with the target variables, missing data, and a

pairwise plot
    - repeat these steps for the next tables, but first run the correlation test to see which features will be worth exploring deeper
- Feature Engineering
    - Take the most important features that have surfaced from the Exploritory data analysis and merge the important data for the piplines to preform on.
- Machine Learning Pipeline
    - List the ML algorithms being used, thier parameters, the loss functions being used and thier preformance
    - Log these experiments
    - Prepare a submission file for the kaggle competition
- Hyperparameter Tuning
    - Do an inital round of hyperparameter tuning by giving some options to the ML pipelines and deciding which work best for this current round
- Video Presentation
    - Explain the past, present, and future of this problem to the class and our educational team. Showing where we came from, what we have done, and what we are doing to improve our preformance and results.

Project Description: Tasks - Visualization

Gantt Chart Visualization



Exploritory Data Analysis (EDA)

EDA: Data Dictionary

In [ ]:

```
#####################################
# Exploritory Data Analysis: Methods
#####################################

def EDA(eda_list):

  # Pulling information from df list
  df_name = eda_list[0]
  df = eda_list[1]

  # Header Section
  print("***********************************************")
  print("                                               ")
  print("           DATAFRAME: " + df_name + "          ")
  print("                                               ")
  print("***********************************************")

  print("\n")

  # Data Frame: Size & Shape
  print("================================================")
  print("Data Frame: Size, Shape & Total Missing Values")
  print("------------------------------------------------")

  print("Number of Rows: " + str(df.shape[0]))
  print("Number of Columns: " + str(df.shape[1]))
  print("Number of Total Missing Values: " + str(df.isna().sum().sum()))
  print("Data Frame Shape: " + str(df.shape))

  print("================================================")

  print("\n")

  # Data Frame: Missing Values by Feature
  print("================================================")
  print("Data Frame: Missing Values by Feature")
  print("------------------------------------------------")

  print("Number of Missing Values by Feature: " + str(df.isna().sum()))

  print("================================================")

  print("\n")

  # Data Frame: Data Types
  print("================================================")
  print("Data Frame: Data Types")
  print("------------------------------------------------")

  print(df.dtypes)
```

```
    print("================================================")

    print("\n")

    # Data Frame: Data Type Count
    print("================================================")
    print("Data Frame: Data Types")
    print("------------------------------------------------")

    print(df.dtypes.value_counts())

    print("================================================")

    print("\n")

    # Data Frame: Summary Statistics
    print("================================================")
    print("Data Frame: Summary Statistics")
    print("------------------------------------------------")

    print(df.describe())

    print("================================================")

    print("\n")

    # Data Frame: Correlation Statistics
    print("================================================")
    print("Data Frame: Correlation Statistics")
    print("------------------------------------------------")

    print(df.corr())

    print("================================================")

    print("\n")

    # Data Frame: Additional Text Based Analysis
    print("================================================")
    print("Data Frame: Additional Information")
    print("------------------------------------------------")

    print(df.info())

    print("================================================")
```

EDA Data Dictionary: application_train.csv

In [ ]:

```
# Entering information to call the EDA Method
eda_info_app_train = ['Application Train', df_app_train]

# Calling EDA Method
EDA(eda_info_app_train)
```

```
**********************************************

          DATAFRAME: Application Train

**********************************************


================================================
Data Frame: Size, Shape & Total Missing Values
------------------------------------------------
Number of Rows: 307511
Number of Columns: 122
Number of Total Missing Values: 9152465
Data Frame Shape: (307511, 122)
================================================


================================================
Data Frame: Missing Values by Feature
------------------------------------------------
Number of Missing Values by Feature: SK_ID_CURR                         0
TARGET                              0
NAME_CONTRACT_TYPE                  0
CODE_GENDER                         0
FLAG_OWN_CAR                        0
                                  ...
AMT_REQ_CREDIT_BUREAU_DAY       41519
AMT_REQ_CREDIT_BUREAU_WEEK      41519
AMT_REQ_CREDIT_BUREAU_MON       41519
AMT_REQ_CREDIT_BUREAU_QRT       41519
AMT_REQ_CREDIT_BUREAU_YEAR      41519
Length: 122, dtype: int64
================================================


================================================
Data Frame: Data Types
------------------------------------------------
SK_ID_CURR                       int64
TARGET                           int64
NAME_CONTRACT_TYPE              object
CODE_GENDER                     object
FLAG_OWN_CAR                    object
                                 ...
AMT_REQ_CREDIT_BUREAU_DAY      float64
AMT_REQ_CREDIT_BUREAU_WEEK     float64
AMT_REQ_CREDIT_BUREAU_MON      float64
AMT_REQ_CREDIT_BUREAU_QRT      float64
AMT_REQ_CREDIT_BUREAU_YEAR     float64
Length: 122, dtype: object
================================================


================================================
Data Frame: Data Types
------------------------------------------------
float64    65
int64      41
object     16
dtype: int64
```

```
================================================


================================================
Data Frame: Summary Statistics
------------------------------------------------
         SK_ID_CURR         TARGET   CNT_CHILDREN  AMT_INCOME_TOTAL  \
count  307511.000000  307511.000000  307511.000000      3.075110e+05
mean   278180.518577       0.080729       0.417052      1.687979e+05
std    102790.175348       0.272419       0.722121      2.371231e+05
min    100002.000000       0.000000       0.000000      2.565000e+04
25%    189145.500000       0.000000       0.000000      1.125000e+05
50%    278202.000000       0.000000       0.000000      1.471500e+05
75%    367142.500000       0.000000       1.000000      2.025000e+05
max    456255.000000       1.000000      19.000000      1.170000e+08

         AMT_CREDIT    AMT_ANNUITY  AMT_GOODS_PRICE  \
count  3.075110e+05  307499.000000     3.072330e+05
mean   5.990260e+05   27108.573909     5.383962e+05
std    4.024908e+05   14493.737315     3.694465e+05
min    4.500000e+04    1615.500000     4.050000e+04
25%    2.700000e+05   16524.000000     2.385000e+05
50%    5.135310e+05   24903.000000     4.500000e+05
75%    8.086500e+05   34596.000000     6.795000e+05
max    4.050000e+06  258025.500000     4.050000e+06

       REGION_POPULATION_RELATIVE     DAYS_BIRTH  DAYS_EMPLOYED  ...  \
count                307511.000000  307511.000000  307511.000000  ...
mean                      0.020868  -16036.995067   63815.045904  ...
std                       0.013831    4363.988632  141275.766519  ...
min                       0.000290  -25229.000000  -17912.000000  ...
25%                       0.010006  -19682.000000   -2760.000000  ...
50%                       0.018850  -15750.000000   -1213.000000  ...
75%                       0.028663  -12413.000000    -289.000000  ...
max                       0.072508   -7489.000000  365243.000000  ...

       FLAG_DOCUMENT_18  FLAG_DOCUMENT_19  FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  \
count     307511.000000     307511.000000     307511.000000     307511.000000
mean           0.008130          0.000595          0.000507          0.000335
std            0.089798          0.024387          0.022518          0.018299
min            0.000000          0.000000          0.000000          0.000000
25%            0.000000          0.000000          0.000000          0.000000
50%            0.000000          0.000000          0.000000          0.000000
75%            0.000000          0.000000          0.000000          0.000000
max            1.000000          1.000000          1.000000          1.000000

       AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
count                265992.000000              265992.000000
mean                      0.006402                   0.007000
std                       0.083849                   0.110757
min                       0.000000                   0.000000
25%                       0.000000                   0.000000
50%                       0.000000                   0.000000
75%                       0.000000                   0.000000
max                       4.000000                   9.000000

       AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
count                265992.000000              265992.000000
mean                      0.034362                   0.267395
std                       0.204685                   0.916002
```

```
min                      0.000000                  0.000000
25%                      0.000000                  0.000000
50%                      0.000000                  0.000000
75%                      0.000000                  0.000000
max                      8.000000                 27.000000

       AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
count               265992.000000               265992.000000
mean                     0.265474                    1.899974
std                      0.794056                    1.869295
min                      0.000000                    0.000000
25%                      0.000000                    0.000000
50%                      0.000000                    1.000000
75%                      0.000000                    3.000000
max                    261.000000                   25.000000

[8 rows x 106 columns]
================================================


================================================
Data Frame: Correlation Statistics
------------------------------------------------
<ipython-input-264-21e78107dac0>:83: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  print(df.corr())
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	\
SK_ID_CURR	1.000000	-0.002108	-0.001129	
TARGET	-0.002108	1.000000	0.019187	
CNT_CHILDREN	-0.001129	0.019187	1.000000	
AMT_INCOME_TOTAL	-0.001820	-0.003982	0.012882	
AMT_CREDIT	-0.000343	-0.030369	0.002145	
...	...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	-0.002193	0.002704	-0.000366	
AMT_REQ_CREDIT_BUREAU_WEEK	0.002099	0.000788	-0.002436	
AMT_REQ_CREDIT_BUREAU_MON	0.000485	-0.012462	-0.010808	
AMT_REQ_CREDIT_BUREAU_QRT	0.001025	-0.002022	-0.007836	
AMT_REQ_CREDIT_BUREAU_YEAR	0.004659	0.019930	-0.041550	

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	\
SK_ID_CURR	-0.001820	-0.000343	-0.000433	
TARGET	-0.003982	-0.030369	-0.012817	
CNT_CHILDREN	0.012882	0.002145	0.021374	
AMT_INCOME_TOTAL	1.000000	0.156870	0.191657	
AMT_CREDIT	0.156870	1.000000	0.770138	
...	...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	0.002944	0.004238	0.002185	
AMT_REQ_CREDIT_BUREAU_WEEK	0.002387	-0.001275	0.013881	
AMT_REQ_CREDIT_BUREAU_MON	0.024700	0.054451	0.039148	
AMT_REQ_CREDIT_BUREAU_QRT	0.004859	0.015925	0.010124	
AMT_REQ_CREDIT_BUREAU_YEAR	0.011690	-0.048448	-0.011320	

	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	\
SK_ID_CURR	-0.000232	0.000849	
TARGET	-0.039645	-0.037227	
CNT_CHILDREN	-0.001827	-0.025573	
AMT_INCOME_TOTAL	0.159610	0.074796	
AMT_CREDIT	0.986968	0.099738	
...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	0.004677	0.001399	
AMT_REQ_CREDIT_BUREAU_WEEK	-0.001007	-0.002149	
AMT_REQ_CREDIT_BUREAU_MON	0.056422	0.078607	
AMT_REQ_CREDIT_BUREAU_QRT	0.016432	-0.001279	
AMT_REQ_CREDIT_BUREAU_YEAR	-0.050998	0.001003	

	DAYS_BIRTH	DAYS_EMPLOYED	...	FLAG_DOCUMENT_18	\
SK_ID_CURR	-0.001500	0.001366	...	0.000509	
TARGET	0.078239	-0.044932	...	-0.007952	
CNT_CHILDREN	0.330938	-0.239818	...	0.004031	
AMT_INCOME_TOTAL	0.027261	-0.064223	...	0.003130	
AMT_CREDIT	-0.055436	-0.066838	...	0.034329	
...	...	...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	0.002255	0.000472	...	0.013281	
AMT_REQ_CREDIT_BUREAU_WEEK	-0.001336	0.003072	...	-0.004640	
AMT_REQ_CREDIT_BUREAU_MON	0.001372	-0.034457	...	-0.001565	
AMT_REQ_CREDIT_BUREAU_QRT	-0.011799	0.015345	...	-0.005125	
AMT_REQ_CREDIT_BUREAU_YEAR	-0.071983	0.049988	...	-0.047432	

	FLAG_DOCUMENT_19	FLAG_DOCUMENT_20	\
SK_ID_CURR	0.000167	0.001073	
TARGET	-0.001358	0.000215	
CNT_CHILDREN	0.000864	0.000988	
AMT_INCOME_TOTAL	0.002408	0.000242	
AMT_CREDIT	0.021082	0.031023	
...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	0.001126	-0.000120	

```
AMT_REQ_CREDIT_BUREAU_WEEK         -0.001275         -0.001770
AMT_REQ_CREDIT_BUREAU_MON          -0.002729          0.001285
AMT_REQ_CREDIT_BUREAU_QRT          -0.001575         -0.001010
AMT_REQ_CREDIT_BUREAU_YEAR         -0.007009         -0.012126

                            FLAG_DOCUMENT_21  AMT_REQ_CREDIT_BUREAU_HOUR  \
SK_ID_CURR                          0.000282                   -0.002672
TARGET                              0.003709                    0.000930
CNT_CHILDREN                       -0.002450                   -0.000410
AMT_INCOME_TOTAL                   -0.000589                    0.000709
AMT_CREDIT                         -0.016148                   -0.003906
...                                      ...                         ...
AMT_REQ_CREDIT_BUREAU_DAY          -0.001130                    0.230374
AMT_REQ_CREDIT_BUREAU_WEEK          0.000081                    0.004706
AMT_REQ_CREDIT_BUREAU_MON          -0.003612                   -0.000018
AMT_REQ_CREDIT_BUREAU_QRT          -0.002004                   -0.002716
AMT_REQ_CREDIT_BUREAU_YEAR         -0.005457                   -0.004597

                            AMT_REQ_CREDIT_BUREAU_DAY  \
SK_ID_CURR                                  -0.002193
TARGET                                       0.002704
CNT_CHILDREN                                -0.000366
AMT_INCOME_TOTAL                             0.002944
AMT_CREDIT                                   0.004238
...                                               ...
AMT_REQ_CREDIT_BUREAU_DAY                    1.000000
AMT_REQ_CREDIT_BUREAU_WEEK                   0.217412
AMT_REQ_CREDIT_BUREAU_MON                   -0.005258
AMT_REQ_CREDIT_BUREAU_QRT                   -0.004416
AMT_REQ_CREDIT_BUREAU_YEAR                  -0.003355

                            AMT_REQ_CREDIT_BUREAU_WEEK  \
SK_ID_CURR                                    0.002099
TARGET                                        0.000788
CNT_CHILDREN                                 -0.002436
AMT_INCOME_TOTAL                              0.002387
AMT_CREDIT                                   -0.001275
...                                                ...
AMT_REQ_CREDIT_BUREAU_DAY                     0.217412
AMT_REQ_CREDIT_BUREAU_WEEK                    1.000000
AMT_REQ_CREDIT_BUREAU_MON                    -0.014096
AMT_REQ_CREDIT_BUREAU_QRT                    -0.015115
AMT_REQ_CREDIT_BUREAU_YEAR                    0.018917

                            AMT_REQ_CREDIT_BUREAU_MON  \
SK_ID_CURR                                   0.000485
TARGET                                      -0.012462
CNT_CHILDREN                                -0.010808
AMT_INCOME_TOTAL                             0.024700
AMT_CREDIT                                   0.054451
...                                               ...
AMT_REQ_CREDIT_BUREAU_DAY                   -0.005258
AMT_REQ_CREDIT_BUREAU_WEEK                  -0.014096
AMT_REQ_CREDIT_BUREAU_MON                    1.000000
AMT_REQ_CREDIT_BUREAU_QRT                   -0.007789
AMT_REQ_CREDIT_BUREAU_YEAR                  -0.004975

                            AMT_REQ_CREDIT_BUREAU_QRT  \
SK_ID_CURR                                   0.001025
TARGET                                      -0.002022
```

```
CNT_CHILDREN                                -0.007836
AMT_INCOME_TOTAL                             0.004859
AMT_CREDIT                                   0.015925
...                                               ...
AMT_REQ_CREDIT_BUREAU_DAY                   -0.004416
AMT_REQ_CREDIT_BUREAU_WEEK                  -0.015115
AMT_REQ_CREDIT_BUREAU_MON                   -0.007789
AMT_REQ_CREDIT_BUREAU_QRT                    1.000000
AMT_REQ_CREDIT_BUREAU_YEAR                   0.076208

                            AMT_REQ_CREDIT_BUREAU_YEAR
SK_ID_CURR                                    0.004659
TARGET                                        0.019930
CNT_CHILDREN                                 -0.041550
AMT_INCOME_TOTAL                              0.011690
AMT_CREDIT                                   -0.048448
...                                                ...
AMT_REQ_CREDIT_BUREAU_DAY                    -0.003355
AMT_REQ_CREDIT_BUREAU_WEEK                    0.018917
AMT_REQ_CREDIT_BUREAU_MON                    -0.004975
AMT_REQ_CREDIT_BUREAU_QRT                     0.076208
AMT_REQ_CREDIT_BUREAU_YEAR                    1.000000

[106 rows x 106 columns]
================================================


================================================
Data Frame: Additional Information
------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
None
================================================
```

EDA Data Dictionary: application_test.csv

In [ ]:

```
# Entering information to call the EDA Method
eda_info_app_test = ['Application Test', df_app_test]

# Calling EDA Method
EDA(eda_info_app_test)
```

```
**********************************************

          DATAFRAME: Application Test

**********************************************


================================================
Data Frame: Size, Shape & Total Missing Values
------------------------------------------------
Number of Rows: 48744
Number of Columns: 121
Number of Total Missing Values: 1404419
Data Frame Shape: (48744, 121)
================================================


================================================
Data Frame: Missing Values by Feature
------------------------------------------------
Number of Missing Values by Feature: SK_ID_CURR                        0
NAME_CONTRACT_TYPE               0
CODE_GENDER                      0
FLAG_OWN_CAR                     0
FLAG_OWN_REALTY                  0
                               ...
AMT_REQ_CREDIT_BUREAU_DAY     6049
AMT_REQ_CREDIT_BUREAU_WEEK    6049
AMT_REQ_CREDIT_BUREAU_MON     6049
AMT_REQ_CREDIT_BUREAU_QRT     6049
AMT_REQ_CREDIT_BUREAU_YEAR    6049
Length: 121, dtype: int64
================================================


================================================
Data Frame: Data Types
------------------------------------------------
SK_ID_CURR                     int64
NAME_CONTRACT_TYPE            object
CODE_GENDER                   object
FLAG_OWN_CAR                  object
FLAG_OWN_REALTY               object
                               ...
AMT_REQ_CREDIT_BUREAU_DAY    float64
AMT_REQ_CREDIT_BUREAU_WEEK   float64
AMT_REQ_CREDIT_BUREAU_MON    float64
AMT_REQ_CREDIT_BUREAU_QRT    float64
AMT_REQ_CREDIT_BUREAU_YEAR   float64
Length: 121, dtype: object
================================================


================================================
Data Frame: Data Types
------------------------------------------------
float64    65
int64      40
object     16
dtype: int64
```

```
================================================


================================================
Data Frame: Summary Statistics
------------------------------------------------
         SK_ID_CURR  CNT_CHILDREN  AMT_INCOME_TOTAL    AMT_CREDIT  \
count   48744.000000  48744.000000      4.874400e+04  4.874400e+04
mean   277796.676350      0.397054      1.784318e+05  5.167404e+05
std    103169.547296      0.709047      1.015226e+05  3.653970e+05
min    100001.000000      0.000000      2.694150e+04  4.500000e+04
25%    188557.750000      0.000000      1.125000e+05  2.606400e+05
50%    277549.000000      0.000000      1.575000e+05  4.500000e+05
75%    367555.500000      1.000000      2.250000e+05  6.750000e+05
max    456250.000000     20.000000      4.410000e+06  2.245500e+06

         AMT_ANNUITY  AMT_GOODS_PRICE  REGION_POPULATION_RELATIVE  \
count   48720.000000     4.874400e+04                48744.000000
mean    29426.240209     4.626188e+05                    0.021226
std     16015.368315     3.367102e+05                    0.014428
min      2295.000000     4.500000e+04                    0.000253
25%     17973.000000     2.250000e+05                    0.010006
50%     26199.000000     3.960000e+05                    0.018850
75%     37390.500000     6.300000e+05                    0.028663
max    180576.000000     2.245500e+06                    0.072508

         DAYS_BIRTH  DAYS_EMPLOYED  DAYS_REGISTRATION  ...  FLAG_DOCUMENT_18  \
count  48744.000000   48744.000000       48744.000000  ...      48744.000000
mean  -16068.084605   67485.366322       -4967.652716  ...          0.001559
std     4325.900393  144348.507136        3552.612035  ...          0.039456
min   -25195.000000  -17463.000000      -23722.000000  ...          0.000000
25%   -19637.000000   -2910.000000       -7459.250000  ...          0.000000
50%   -15785.000000   -1293.000000       -4490.000000  ...          0.000000
75%   -12496.000000    -296.000000       -1901.000000  ...          0.000000
max    -7338.000000  365243.000000           0.000000  ...          1.000000

       FLAG_DOCUMENT_19  FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  \
count           48744.0           48744.0           48744.0
mean                0.0               0.0               0.0
std                 0.0               0.0               0.0
min                 0.0               0.0               0.0
25%                 0.0               0.0               0.0
50%                 0.0               0.0               0.0
75%                 0.0               0.0               0.0
max                 0.0               0.0               0.0

       AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
count                42695.000000               42695.000000
mean                     0.002108                   0.001803
std                      0.046373                   0.046132
min                      0.000000                   0.000000
25%                      0.000000                   0.000000
50%                      0.000000                   0.000000
75%                      0.000000                   0.000000
max                      2.000000                   2.000000

       AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
count                42695.000000               42695.000000
mean                     0.002787                   0.009299
std                      0.054037                   0.110924
```

```
min                      0.000000                  0.000000
25%                      0.000000                  0.000000
50%                      0.000000                  0.000000
75%                      0.000000                  0.000000
max                      2.000000                  6.000000

       AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
count                42695.000000                42695.000000
mean                     0.546902                    1.983769
std                      0.693305                    1.838873
min                      0.000000                    0.000000
25%                      0.000000                    0.000000
50%                      0.000000                    2.000000
75%                      1.000000                    3.000000
max                      7.000000                   17.000000

[8 rows x 105 columns]
================================================


================================================
Data Frame: Correlation Statistics
------------------------------------------------
<ipython-input-264-21e78107dac0>:83: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  print(df.corr())
```

	SK_ID_CURR	CNT_CHILDREN	AMT_INCOME_TOTAL	\
SK_ID_CURR	1.000000	0.000635	0.001278	
CNT_CHILDREN	0.000635	1.000000	0.038962	
AMT_INCOME_TOTAL	0.001278	0.038962	1.000000	
AMT_CREDIT	0.005014	0.027840	0.396572	
AMT_ANNUITY	0.007112	0.056770	0.457833	
...	...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	0.001083	0.001539	0.004989	
AMT_REQ_CREDIT_BUREAU_WEEK	0.001178	0.007523	-0.002867	
AMT_REQ_CREDIT_BUREAU_MON	0.000430	-0.008337	0.008691	
AMT_REQ_CREDIT_BUREAU_QRT	-0.002092	0.029006	0.007410	
AMT_REQ_CREDIT_BUREAU_YEAR	0.003457	-0.039265	0.003281	

	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	\
SK_ID_CURR	0.005014	0.007112	0.005097	
CNT_CHILDREN	0.027840	0.056770	0.025507	
AMT_INCOME_TOTAL	0.396572	0.457833	0.401995	
AMT_CREDIT	1.000000	0.777733	0.988056	
AMT_ANNUITY	0.777733	1.000000	0.787033	
...	...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	0.004882	0.006681	0.004865	
AMT_REQ_CREDIT_BUREAU_WEEK	0.002904	0.003085	0.003358	
AMT_REQ_CREDIT_BUREAU_MON	-0.000156	0.005695	-0.000254	
AMT_REQ_CREDIT_BUREAU_QRT	-0.007750	0.012443	-0.008490	
AMT_REQ_CREDIT_BUREAU_YEAR	-0.034533	-0.044901	-0.036227	

	REGION_POPULATION_RELATIVE	DAYS_BIRTH	\
SK_ID_CURR	0.003324	0.002325	
CNT_CHILDREN	-0.015231	0.317877	
AMT_INCOME_TOTAL	0.199773	0.054400	
AMT_CREDIT	0.135694	-0.046169	
AMT_ANNUITY	0.150864	0.047859	
...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	-0.011773	-0.000386	
AMT_REQ_CREDIT_BUREAU_WEEK	-0.008321	0.012422	
AMT_REQ_CREDIT_BUREAU_MON	0.000105	0.014094	
AMT_REQ_CREDIT_BUREAU_QRT	-0.026650	0.088752	
AMT_REQ_CREDIT_BUREAU_YEAR	0.001015	-0.095551	

	DAYS_EMPLOYED	DAYS_REGISTRATION	...	\
SK_ID_CURR	-0.000845	0.001032	...	
CNT_CHILDREN	-0.238319	0.175054	...	
AMT_INCOME_TOTAL	-0.154619	0.067973	...	
AMT_CREDIT	-0.083483	0.030740	...	
AMT_ANNUITY	-0.137772	0.064450	...	
...	...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	-0.000785	-0.000152	...	
AMT_REQ_CREDIT_BUREAU_WEEK	-0.014058	0.008692	...	
AMT_REQ_CREDIT_BUREAU_MON	-0.013891	0.007414	...	
AMT_REQ_CREDIT_BUREAU_QRT	-0.044351	0.046011	...	
AMT_REQ_CREDIT_BUREAU_YEAR	0.064698	-0.036887	...	

	FLAG_DOCUMENT_18	FLAG_DOCUMENT_19	\
SK_ID_CURR	-0.006286	NaN	
CNT_CHILDREN	-0.000862	NaN	
AMT_INCOME_TOTAL	-0.006624	NaN	
AMT_CREDIT	-0.000197	NaN	
AMT_ANNUITY	-0.010762	NaN	
...	...	...	
AMT_REQ_CREDIT_BUREAU_DAY	-0.001515	NaN	

```
AMT_REQ_CREDIT_BUREAU_WEEK          0.009205               NaN
AMT_REQ_CREDIT_BUREAU_MON          -0.003248               NaN
AMT_REQ_CREDIT_BUREAU_QRT          -0.010480               NaN
AMT_REQ_CREDIT_BUREAU_YEAR         -0.009864               NaN

                            FLAG_DOCUMENT_20  FLAG_DOCUMENT_21  \
SK_ID_CURR                               NaN               NaN
CNT_CHILDREN                             NaN               NaN
AMT_INCOME_TOTAL                         NaN               NaN
AMT_CREDIT                               NaN               NaN
AMT_ANNUITY                              NaN               NaN
...                                      ...               ...
AMT_REQ_CREDIT_BUREAU_DAY                NaN               NaN
AMT_REQ_CREDIT_BUREAU_WEEK               NaN               NaN
AMT_REQ_CREDIT_BUREAU_MON                NaN               NaN
AMT_REQ_CREDIT_BUREAU_QRT                NaN               NaN
AMT_REQ_CREDIT_BUREAU_YEAR               NaN               NaN

                            AMT_REQ_CREDIT_BUREAU_HOUR  \
SK_ID_CURR                                   -0.000307
CNT_CHILDREN                                  0.006362
AMT_INCOME_TOTAL                              0.010227
AMT_CREDIT                                   -0.001092
AMT_ANNUITY                                   0.008428
...                                                ...
AMT_REQ_CREDIT_BUREAU_DAY                     0.151506
AMT_REQ_CREDIT_BUREAU_WEEK                   -0.002345
AMT_REQ_CREDIT_BUREAU_MON                     0.023510
AMT_REQ_CREDIT_BUREAU_QRT                    -0.003075
AMT_REQ_CREDIT_BUREAU_YEAR                    0.011938

                            AMT_REQ_CREDIT_BUREAU_DAY  \
SK_ID_CURR                                   0.001083
CNT_CHILDREN                                 0.001539
AMT_INCOME_TOTAL                             0.004989
AMT_CREDIT                                   0.004882
AMT_ANNUITY                                  0.006681
...                                               ...
AMT_REQ_CREDIT_BUREAU_DAY                    1.000000
AMT_REQ_CREDIT_BUREAU_WEEK                   0.035567
AMT_REQ_CREDIT_BUREAU_MON                    0.005877
AMT_REQ_CREDIT_BUREAU_QRT                    0.006509
AMT_REQ_CREDIT_BUREAU_YEAR                   0.002002

                            AMT_REQ_CREDIT_BUREAU_WEEK  \
SK_ID_CURR                                    0.001178
CNT_CHILDREN                                  0.007523
AMT_INCOME_TOTAL                             -0.002867
AMT_CREDIT                                    0.002904
AMT_ANNUITY                                   0.003085
...                                                ...
AMT_REQ_CREDIT_BUREAU_DAY                     0.035567
AMT_REQ_CREDIT_BUREAU_WEEK                    1.000000
AMT_REQ_CREDIT_BUREAU_MON                     0.054291
AMT_REQ_CREDIT_BUREAU_QRT                     0.024957
AMT_REQ_CREDIT_BUREAU_YEAR                   -0.000252

                            AMT_REQ_CREDIT_BUREAU_MON  \
SK_ID_CURR                                   0.000430
CNT_CHILDREN                                -0.008337
```

```
AMT_INCOME_TOTAL                          0.008691
AMT_CREDIT                               -0.000156
AMT_ANNUITY                               0.005695
...                                            ...
AMT_REQ_CREDIT_BUREAU_DAY                 0.005877
AMT_REQ_CREDIT_BUREAU_WEEK                0.054291
AMT_REQ_CREDIT_BUREAU_MON                 1.000000
AMT_REQ_CREDIT_BUREAU_QRT                 0.005446
AMT_REQ_CREDIT_BUREAU_YEAR                0.026118

                           AMT_REQ_CREDIT_BUREAU_QRT  \
SK_ID_CURR                                -0.002092
CNT_CHILDREN                               0.029006
AMT_INCOME_TOTAL                           0.007410
AMT_CREDIT                                -0.007750
AMT_ANNUITY                                0.012443
...                                             ...
AMT_REQ_CREDIT_BUREAU_DAY                  0.006509
AMT_REQ_CREDIT_BUREAU_WEEK                 0.024957
AMT_REQ_CREDIT_BUREAU_MON                  0.005446
AMT_REQ_CREDIT_BUREAU_QRT                  1.000000
AMT_REQ_CREDIT_BUREAU_YEAR                -0.013081

                           AMT_REQ_CREDIT_BUREAU_YEAR
SK_ID_CURR                                  0.003457
CNT_CHILDREN                               -0.039265
AMT_INCOME_TOTAL                            0.003281
AMT_CREDIT                                 -0.034533
AMT_ANNUITY                                -0.044901
...                                              ...
AMT_REQ_CREDIT_BUREAU_DAY                   0.002002
AMT_REQ_CREDIT_BUREAU_WEEK                 -0.000252
AMT_REQ_CREDIT_BUREAU_MON                   0.026118
AMT_REQ_CREDIT_BUREAU_QRT                  -0.013081
AMT_REQ_CREDIT_BUREAU_YEAR                  1.000000

[105 rows x 105 columns]
================================================


================================================
Data Frame: Additional Information
------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48744 entries, 0 to 48743
Columns: 121 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(40), object(16)
memory usage: 45.0+ MB
None
================================================
```

EDA Data Dictionary: bureau.csv

In [ ]:

```
# Entering information to call the EDA Method
eda_info_bureau = ['Bureau', df_bureau]
```

```
# Calling EDA Method
EDA(eda_info_bureau)
```

```
**********************************************

           DATAFRAME: Bureau

**********************************************


================================================
Data Frame: Size, Shape & Total Missing Values
------------------------------------------------
Number of Rows: 1716428
Number of Columns: 17
Number of Total Missing Values: 3939947
Data Frame Shape: (1716428, 17)
================================================


================================================
Data Frame: Missing Values by Feature
------------------------------------------------
Number of Missing Values by Feature: SK_ID_CURR                        0
SK_ID_BUREAU                       0
CREDIT_ACTIVE                      0
CREDIT_CURRENCY                    0
DAYS_CREDIT                        0
CREDIT_DAY_OVERDUE                 0
DAYS_CREDIT_ENDDATE           105553
DAYS_ENDDATE_FACT             633653
AMT_CREDIT_MAX_OVERDUE       1124488
CNT_CREDIT_PROLONG                 0
AMT_CREDIT_SUM                    13
AMT_CREDIT_SUM_DEBT           257669
AMT_CREDIT_SUM_LIMIT          591780
AMT_CREDIT_SUM_OVERDUE             0
CREDIT_TYPE                        0
DAYS_CREDIT_UPDATE                 0
AMT_ANNUITY                  1226791
dtype: int64
================================================


================================================
Data Frame: Data Types
------------------------------------------------
SK_ID_CURR                    int64
SK_ID_BUREAU                  int64
CREDIT_ACTIVE                object
CREDIT_CURRENCY              object
DAYS_CREDIT                   int64
CREDIT_DAY_OVERDUE            int64
DAYS_CREDIT_ENDDATE         float64
DAYS_ENDDATE_FACT           float64
AMT_CREDIT_MAX_OVERDUE      float64
CNT_CREDIT_PROLONG            int64
AMT_CREDIT_SUM              float64
AMT_CREDIT_SUM_DEBT         float64
AMT_CREDIT_SUM_LIMIT        float64
AMT_CREDIT_SUM_OVERDUE      float64
CREDIT_TYPE                  object
DAYS_CREDIT_UPDATE            int64
```

```
AMT_ANNUITY               float64
dtype: object
================================================


================================================
Data Frame: Data Types
------------------------------------------------
float64    8
int64      6
object     3
dtype: int64
================================================


================================================
Data Frame: Summary Statistics
------------------------------------------------
         SK_ID_CURR  SK_ID_BUREAU   DAYS_CREDIT  CREDIT_DAY_OVERDUE  \
count  1.716428e+06  1.716428e+06  1.716428e+06        1.716428e+06
mean   2.782149e+05  5.924434e+06 -1.142108e+03        8.181666e-01
std    1.029386e+05  5.322657e+05  7.951649e+02        3.654443e+01
min    1.000010e+05  5.000000e+06 -2.922000e+03        0.000000e+00
25%    1.888668e+05  5.463954e+06 -1.666000e+03        0.000000e+00
50%    2.780550e+05  5.926304e+06 -9.870000e+02        0.000000e+00
75%    3.674260e+05  6.385681e+06 -4.740000e+02        0.000000e+00
max    4.562550e+05  6.843457e+06  0.000000e+00        2.792000e+03

       DAYS_CREDIT_ENDDATE  DAYS_ENDDATE_FACT  AMT_CREDIT_MAX_OVERDUE  \
count         1.610875e+06       1.082775e+06            5.919400e+05
mean          5.105174e+02      -1.017437e+03            3.825418e+03
std           4.994220e+03       7.140106e+02            2.060316e+05
min          -4.206000e+04      -4.202300e+04            0.000000e+00
25%          -1.138000e+03      -1.489000e+03            0.000000e+00
50%          -3.300000e+02      -8.970000e+02            0.000000e+00
75%           4.740000e+02      -4.250000e+02            0.000000e+00
max           3.119900e+04       0.000000e+00            1.159872e+08

       CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  AMT_CREDIT_SUM_DEBT  \
count        1.716428e+06    1.716415e+06         1.458759e+06
mean         6.410406e-03    3.549946e+05         1.370851e+05
std          9.622391e-02    1.149811e+06         6.774011e+05
min          0.000000e+00    0.000000e+00        -4.705600e+06
25%          0.000000e+00    5.130000e+04         0.000000e+00
50%          0.000000e+00    1.255185e+05         0.000000e+00
75%          0.000000e+00    3.150000e+05         4.015350e+04
max          9.000000e+00    5.850000e+08         1.701000e+08

       AMT_CREDIT_SUM_LIMIT  AMT_CREDIT_SUM_OVERDUE  DAYS_CREDIT_UPDATE  \
count          1.124648e+06            1.716428e+06        1.716428e+06
mean           6.229515e+03            3.791276e+01       -5.937483e+02
std            4.503203e+04            5.937650e+03        7.207473e+02
min           -5.864061e+05            0.000000e+00       -4.194700e+04
25%            0.000000e+00            0.000000e+00       -9.080000e+02
50%            0.000000e+00            0.000000e+00       -3.950000e+02
75%            0.000000e+00            0.000000e+00       -3.300000e+01
max            4.705600e+06            3.756681e+06        3.720000e+02

        AMT_ANNUITY
count  4.896370e+05
```

```
mean   1.571276e+04
std    3.258269e+05
min    0.000000e+00
25%    0.000000e+00
50%    0.000000e+00
75%    1.350000e+04
max    1.184534e+08
================================================


================================================
Data Frame: Correlation Statistics
------------------------------------------------
<ipython-input-264-21e78107dac0>:83: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  print(df.corr())
```

	SK_ID_CURR	SK_ID_BUREAU	DAYS_CREDIT
SK_ID_CURR	1.000000	0.000135	0.000266
SK_ID_BUREAU	0.000135	1.000000	0.013015
DAYS_CREDIT	0.000266	0.013015	1.000000
CREDIT_DAY_OVERDUE	0.000283	-0.002628	-0.027266
DAYS_CREDIT_ENDDATE	0.000456	0.009107	0.225682
DAYS_ENDDATE_FACT	-0.000648	0.017890	0.875359
AMT_CREDIT_MAX_OVERDUE	0.001329	0.002290	-0.014724
CNT_CREDIT_PROLONG	-0.000388	-0.000740	-0.030460
AMT_CREDIT_SUM	0.001179	0.007962	0.050883
AMT_CREDIT_SUM_DEBT	-0.000790	0.005732	0.135397
AMT_CREDIT_SUM_LIMIT	-0.000304	-0.003986	0.025140
AMT_CREDIT_SUM_OVERDUE	-0.000014	-0.000499	-0.000383
DAYS_CREDIT_UPDATE	0.000510	0.019398	0.688771
AMT_ANNUITY	-0.002727	0.001799	0.005676

	CREDIT_DAY_OVERDUE	DAYS_CREDIT_ENDDATE
SK_ID_CURR	0.000283	0.000456
SK_ID_BUREAU	-0.002628	0.009107
DAYS_CREDIT	-0.027266	0.225682
CREDIT_DAY_OVERDUE	1.000000	-0.007352
DAYS_CREDIT_ENDDATE	-0.007352	1.000000
DAYS_ENDDATE_FACT	-0.008637	0.248825
AMT_CREDIT_MAX_OVERDUE	0.001249	0.000577
CNT_CREDIT_PROLONG	0.002756	0.113683
AMT_CREDIT_SUM	-0.003292	0.055424
AMT_CREDIT_SUM_DEBT	-0.002355	0.081298
AMT_CREDIT_SUM_LIMIT	-0.000345	0.095421
AMT_CREDIT_SUM_OVERDUE	0.090951	0.001077
DAYS_CREDIT_UPDATE	-0.018461	0.248525
AMT_ANNUITY	-0.000339	0.000475

	DAYS_ENDDATE_FACT	AMT_CREDIT_MAX_OVERDUE
SK_ID_CURR	-0.000648	0.001329
SK_ID_BUREAU	0.017890	0.002290
DAYS_CREDIT	0.875359	-0.014724
CREDIT_DAY_OVERDUE	-0.008637	0.001249
DAYS_CREDIT_ENDDATE	0.248825	0.000577
DAYS_ENDDATE_FACT	1.000000	0.000999
AMT_CREDIT_MAX_OVERDUE	0.000999	1.000000
CNT_CREDIT_PROLONG	0.012017	0.001523
AMT_CREDIT_SUM	0.059096	0.081663
AMT_CREDIT_SUM_DEBT	0.019609	0.014007
AMT_CREDIT_SUM_LIMIT	0.019476	-0.000112
AMT_CREDIT_SUM_OVERDUE	-0.000332	0.015036
DAYS_CREDIT_UPDATE	0.751294	-0.000749
AMT_ANNUITY	0.006274	0.001578

	CNT_CREDIT_PROLONG	AMT_CREDIT_SUM
SK_ID_CURR	-0.000388	0.001179
SK_ID_BUREAU	-0.000740	0.007962
DAYS_CREDIT	-0.030460	0.050883
CREDIT_DAY_OVERDUE	0.002756	-0.003292
DAYS_CREDIT_ENDDATE	0.113683	0.055424
DAYS_ENDDATE_FACT	0.012017	0.059096
AMT_CREDIT_MAX_OVERDUE	0.001523	0.081663
CNT_CREDIT_PROLONG	1.000000	-0.008345
AMT_CREDIT_SUM	-0.008345	1.000000
AMT_CREDIT_SUM_DEBT	-0.001366	0.683419
AMT_CREDIT_SUM_LIMIT	0.073805	0.003756

```
AMT_CREDIT_SUM_OVERDUE            0.000002        0.006342
DAYS_CREDIT_UPDATE                0.017864        0.104629
AMT_ANNUITY                      -0.000465        0.049146

                         AMT_CREDIT_SUM_DEBT  AMT_CREDIT_SUM_LIMIT  \
SK_ID_CURR                         -0.000790             -0.000304
SK_ID_BUREAU                        0.005732             -0.003986
DAYS_CREDIT                         0.135397              0.025140
CREDIT_DAY_OVERDUE                 -0.002355             -0.000345
DAYS_CREDIT_ENDDATE                 0.081298              0.095421
DAYS_ENDDATE_FACT                   0.019609              0.019476
AMT_CREDIT_MAX_OVERDUE              0.014007             -0.000112
CNT_CREDIT_PROLONG                 -0.001366              0.073805
AMT_CREDIT_SUM                      0.683419              0.003756
AMT_CREDIT_SUM_DEBT                 1.000000             -0.018215
AMT_CREDIT_SUM_LIMIT               -0.018215              1.000000
AMT_CREDIT_SUM_OVERDUE              0.008046             -0.000687
DAYS_CREDIT_UPDATE                  0.141235              0.046028
AMT_ANNUITY                         0.025507              0.004392

                         AMT_CREDIT_SUM_OVERDUE  DAYS_CREDIT_UPDATE  \
SK_ID_CURR                            -0.000014            0.000510
SK_ID_BUREAU                          -0.000499            0.019398
DAYS_CREDIT                           -0.000383            0.688771
CREDIT_DAY_OVERDUE                     0.090951           -0.018461
DAYS_CREDIT_ENDDATE                    0.001077            0.248525
DAYS_ENDDATE_FACT                     -0.000332            0.751294
AMT_CREDIT_MAX_OVERDUE                 0.015036           -0.000749
CNT_CREDIT_PROLONG                     0.000002            0.017864
AMT_CREDIT_SUM                         0.006342            0.104629
AMT_CREDIT_SUM_DEBT                    0.008046            0.141235
AMT_CREDIT_SUM_LIMIT                  -0.000687            0.046028
AMT_CREDIT_SUM_OVERDUE                 1.000000            0.003528
DAYS_CREDIT_UPDATE                     0.003528            1.000000
AMT_ANNUITY                            0.000344            0.008418

                         AMT_ANNUITY
SK_ID_CURR                 -0.002727
SK_ID_BUREAU                0.001799
DAYS_CREDIT                 0.005676
CREDIT_DAY_OVERDUE         -0.000339
DAYS_CREDIT_ENDDATE         0.000475
DAYS_ENDDATE_FACT           0.006274
AMT_CREDIT_MAX_OVERDUE      0.001578
CNT_CREDIT_PROLONG         -0.000465
AMT_CREDIT_SUM              0.049146
AMT_CREDIT_SUM_DEBT         0.025507
AMT_CREDIT_SUM_LIMIT        0.004392
AMT_CREDIT_SUM_OVERDUE      0.000344
DAYS_CREDIT_UPDATE          0.008418
AMT_ANNUITY                 1.000000
================================================


================================================
Data Frame: Additional Information
------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1716428 entries, 0 to 1716427
Data columns (total 17 columns):
```

```
 #   Column                  Dtype
---  ------                  -----
 0   SK_ID_CURR              int64
 1   SK_ID_BUREAU            int64
 2   CREDIT_ACTIVE           object
 3   CREDIT_CURRENCY         object
 4   DAYS_CREDIT             int64
 5   CREDIT_DAY_OVERDUE      int64
 6   DAYS_CREDIT_ENDDATE     float64
 7   DAYS_ENDDATE_FACT       float64
 8   AMT_CREDIT_MAX_OVERDUE  float64
 9   CNT_CREDIT_PROLONG      int64
 10  AMT_CREDIT_SUM          float64
 11  AMT_CREDIT_SUM_DEBT     float64
 12  AMT_CREDIT_SUM_LIMIT    float64
 13  AMT_CREDIT_SUM_OVERDUE  float64
 14  CREDIT_TYPE             object
 15  DAYS_CREDIT_UPDATE      int64
 16  AMT_ANNUITY             float64
dtypes: float64(8), int64(6), object(3)
memory usage: 222.6+ MB
None
================================================
```

EDA Data Dictionary: bureau_balance.csv

In [ ]:

```
# Entering information to call the EDA Method
eda_info_bureau_bal = ['Bureau Balance', df_bureau_bal]

# Calling EDA Method
EDA(eda_info_bureau_bal)
```

```
**********************************************

          DATAFRAME: Bureau Balance

**********************************************


================================================
Data Frame: Size, Shape & Total Missing Values
------------------------------------------------
Number of Rows: 27299925
Number of Columns: 3
Number of Total Missing Values: 0
Data Frame Shape: (27299925, 3)
================================================


================================================
Data Frame: Missing Values by Feature
------------------------------------------------
Number of Missing Values by Feature: SK_ID_BUREAU      0
MONTHS_BALANCE    0
STATUS            0
dtype: int64
================================================


================================================
Data Frame: Data Types
------------------------------------------------
SK_ID_BUREAU       int64
MONTHS_BALANCE     int64
STATUS            object
dtype: object
================================================


================================================
Data Frame: Data Types
------------------------------------------------
int64     2
object    1
dtype: int64
================================================


================================================
Data Frame: Summary Statistics
------------------------------------------------
       SK_ID_BUREAU  MONTHS_BALANCE
count  2.729992e+07    2.729992e+07
mean   6.036297e+06   -3.074169e+01
std    4.923489e+05    2.386451e+01
min    5.001709e+06   -9.600000e+01
25%    5.730933e+06   -4.600000e+01
50%    6.070821e+06   -2.500000e+01
75%    6.431951e+06   -1.100000e+01
max    6.842888e+06    0.000000e+00
================================================
```

```
================================================
Data Frame: Correlation Statistics
------------------------------------------------
<ipython-input-264-21e78107dac0>:83: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  print(df.corr())
                SK_ID_BUREAU  MONTHS_BALANCE
SK_ID_BUREAU        1.000000        0.011873
MONTHS_BALANCE      0.011873        1.000000
================================================


================================================
Data Frame: Additional Information
------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27299925 entries, 0 to 27299924
Data columns (total 3 columns):
 #   Column          Dtype
---  ------          -----
 0   SK_ID_BUREAU    int64
 1   MONTHS_BALANCE  int64
 2   STATUS          object
dtypes: int64(2), object(1)
memory usage: 624.8+ MB
None
================================================
```

EDA Data Dictionary: POS_CASH_balance.csv

In [ ]:

```
# Entering information to call the EDA Method
eda_info_pos_cash_bal = ['POS_CASH Balance', df_pos_cash_bal]

# Calling EDA Method
EDA(eda_info_pos_cash_bal)
```

```
**********************************************

           DATAFRAME: POS_CASH Balance

**********************************************


================================================
Data Frame: Size, Shape & Total Missing Values
------------------------------------------------
Number of Rows: 10001358
Number of Columns: 8
Number of Total Missing Values: 52158
Data Frame Shape: (10001358, 8)
================================================


================================================
Data Frame: Missing Values by Feature
------------------------------------------------
Number of Missing Values by Feature: SK_ID_PREV                    0
SK_ID_CURR                    0
MONTHS_BALANCE                0
CNT_INSTALMENT            26071
CNT_INSTALMENT_FUTURE     26087
NAME_CONTRACT_STATUS          0
SK_DPD                        0
SK_DPD_DEF                    0
dtype: int64
================================================


================================================
Data Frame: Data Types
------------------------------------------------
SK_ID_PREV                 int64
SK_ID_CURR                 int64
MONTHS_BALANCE             int64
CNT_INSTALMENT           float64
CNT_INSTALMENT_FUTURE    float64
NAME_CONTRACT_STATUS      object
SK_DPD                     int64
SK_DPD_DEF                 int64
dtype: object
================================================


================================================
Data Frame: Data Types
------------------------------------------------
int64      5
float64    2
object     1
dtype: int64
================================================


================================================
Data Frame: Summary Statistics
------------------------------------------------
```

```
         SK_ID_PREV    SK_ID_CURR  MONTHS_BALANCE  CNT_INSTALMENT  \
count  1.000136e+07  1.000136e+07    1.000136e+07    9.975287e+06
mean   1.903217e+06  2.784039e+05   -3.501259e+01    1.708965e+01
std    5.358465e+05  1.027637e+05    2.606657e+01    1.199506e+01
min    1.000001e+06  1.000010e+05   -9.600000e+01    1.000000e+00
25%    1.434405e+06  1.895500e+05   -5.400000e+01    1.000000e+01
50%    1.896565e+06  2.786540e+05   -2.800000e+01    1.200000e+01
75%    2.368963e+06  3.674290e+05   -1.300000e+01    2.400000e+01
max    2.843499e+06  4.562550e+05   -1.000000e+00    9.200000e+01

       CNT_INSTALMENT_FUTURE        SK_DPD    SK_DPD_DEF
count           9.975271e+06  1.000136e+07  1.000136e+07
mean            1.048384e+01  1.160693e+01  6.544684e-01
std             1.110906e+01  1.327140e+02  3.276249e+01
min             0.000000e+00  0.000000e+00  0.000000e+00
25%             3.000000e+00  0.000000e+00  0.000000e+00
50%             7.000000e+00  0.000000e+00  0.000000e+00
75%             1.400000e+01  0.000000e+00  0.000000e+00
max             8.500000e+01  4.231000e+03  3.595000e+03
================================================


================================================
Data Frame: Correlation Statistics
------------------------------------------------
<ipython-input-264-21e78107dac0>:83: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  print(df.corr())
```

```
                        SK_ID_PREV  SK_ID_CURR  MONTHS_BALANCE  CNT_INSTALMENT  \
SK_ID_PREV                1.000000   -0.000336        0.001835        0.003820
SK_ID_CURR               -0.000336    1.000000        0.000404        0.000144
MONTHS_BALANCE            0.001835    0.000404        1.000000        0.336163
CNT_INSTALMENT            0.003820    0.000144        0.336163        1.000000
CNT_INSTALMENT_FUTURE     0.003679   -0.000559        0.271595        0.871276
SK_DPD                   -0.000487    0.003118       -0.018939       -0.060803
SK_DPD_DEF                0.004848    0.001948       -0.000381       -0.014154

                        CNT_INSTALMENT_FUTURE    SK_DPD  SK_DPD_DEF
SK_ID_PREV                           0.003679 -0.000487    0.004848
SK_ID_CURR                          -0.000559  0.003118    0.001948
MONTHS_BALANCE                       0.271595 -0.018939   -0.000381
CNT_INSTALMENT                       0.871276 -0.060803   -0.014154
CNT_INSTALMENT_FUTURE                1.000000 -0.082004   -0.017436
SK_DPD                              -0.082004  1.000000    0.245782
SK_DPD_DEF                          -0.017436  0.245782    1.000000
================================================


================================================
Data Frame: Additional Information
------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001358 entries, 0 to 10001357
Data columns (total 8 columns):
 #   Column                 Dtype
---  ------                 -----
 0   SK_ID_PREV             int64
 1   SK_ID_CURR             int64
 2   MONTHS_BALANCE         int64
 3   CNT_INSTALMENT         float64
 4   CNT_INSTALMENT_FUTURE  float64
 5   NAME_CONTRACT_STATUS   object
 6   SK_DPD                 int64
 7   SK_DPD_DEF             int64
dtypes: float64(2), int64(5), object(1)
memory usage: 610.4+ MB
None
================================================
```

EDA Data Dictionary: credit_card_balance.csv

In [ ]:

```
# Entering information to call the EDA Method
eda_info_credit_card_bal = ['Credit Card Balance', df_credit_card_bal]

# Calling EDA Method
EDA(eda_info_credit_card_bal)
```

```
**********************************************

          DATAFRAME: Credit Card Balance

**********************************************


================================================
Data Frame: Size, Shape & Total Missing Values
------------------------------------------------
Number of Rows: 3840312
Number of Columns: 23
Number of Total Missing Values: 5877356
Data Frame Shape: (3840312, 23)
================================================


================================================
Data Frame: Missing Values by Feature
------------------------------------------------
Number of Missing Values by Feature: SK_ID_PREV                          0
SK_ID_CURR                          0
MONTHS_BALANCE                      0
AMT_BALANCE                         0
AMT_CREDIT_LIMIT_ACTUAL             0
AMT_DRAWINGS_ATM_CURRENT       749816
AMT_DRAWINGS_CURRENT                0
AMT_DRAWINGS_OTHER_CURRENT     749816
AMT_DRAWINGS_POS_CURRENT       749816
AMT_INST_MIN_REGULARITY        305236
AMT_PAYMENT_CURRENT            767988
AMT_PAYMENT_TOTAL_CURRENT           0
AMT_RECEIVABLE_PRINCIPAL            0
AMT_RECIVABLE                       0
AMT_TOTAL_RECEIVABLE                0
CNT_DRAWINGS_ATM_CURRENT       749816
CNT_DRAWINGS_CURRENT                0
CNT_DRAWINGS_OTHER_CURRENT     749816
CNT_DRAWINGS_POS_CURRENT       749816
CNT_INSTALMENT_MATURE_CUM      305236
NAME_CONTRACT_STATUS                0
SK_DPD                              0
SK_DPD_DEF                          0
dtype: int64
================================================


================================================
Data Frame: Data Types
------------------------------------------------
SK_ID_PREV                      int64
SK_ID_CURR                      int64
MONTHS_BALANCE                  int64
AMT_BALANCE                   float64
AMT_CREDIT_LIMIT_ACTUAL         int64
AMT_DRAWINGS_ATM_CURRENT      float64
AMT_DRAWINGS_CURRENT          float64
AMT_DRAWINGS_OTHER_CURRENT    float64
AMT_DRAWINGS_POS_CURRENT      float64
AMT_INST_MIN_REGULARITY       float64
```

```
AMT_PAYMENT_CURRENT           float64
AMT_PAYMENT_TOTAL_CURRENT     float64
AMT_RECEIVABLE_PRINCIPAL      float64
AMT_RECIVABLE                 float64
AMT_TOTAL_RECEIVABLE          float64
CNT_DRAWINGS_ATM_CURRENT      float64
CNT_DRAWINGS_CURRENT            int64
CNT_DRAWINGS_OTHER_CURRENT    float64
CNT_DRAWINGS_POS_CURRENT      float64
CNT_INSTALMENT_MATURE_CUM     float64
NAME_CONTRACT_STATUS           object
SK_DPD                          int64
SK_DPD_DEF                      int64
dtype: object
================================================


================================================
Data Frame: Data Types
------------------------------------------------
float64    15
int64       7
object      1
dtype: int64
================================================


================================================
Data Frame: Summary Statistics
------------------------------------------------
         SK_ID_PREV    SK_ID_CURR  MONTHS_BALANCE   AMT_BALANCE  \
count  3.840312e+06  3.840312e+06    3.840312e+06  3.840312e+06
mean   1.904504e+06  2.783242e+05   -3.452192e+01  5.830016e+04
std    5.364695e+05  1.027045e+05    2.666775e+01  1.063070e+05
min    1.000018e+06  1.000060e+05   -9.600000e+01 -4.202502e+05
25%    1.434385e+06  1.895170e+05   -5.500000e+01  0.000000e+00
50%    1.897122e+06  2.783960e+05   -2.800000e+01  0.000000e+00
75%    2.369328e+06  3.675800e+05   -1.100000e+01  8.904669e+04
max    2.843496e+06  4.562500e+05   -1.000000e+00  1.505902e+06

       AMT_CREDIT_LIMIT_ACTUAL  AMT_DRAWINGS_ATM_CURRENT  \
count             3.840312e+06              3.090496e+06
mean              1.538080e+05              5.961325e+03
std               1.651457e+05              2.822569e+04
min               0.000000e+00             -6.827310e+03
25%               4.500000e+04              0.000000e+00
50%               1.125000e+05              0.000000e+00
75%               1.800000e+05              0.000000e+00
max               1.350000e+06              2.115000e+06

       AMT_DRAWINGS_CURRENT  AMT_DRAWINGS_OTHER_CURRENT  \
count          3.840312e+06                3.090496e+06
mean           7.433388e+03                2.881696e+02
std            3.384608e+04                8.201989e+03
min           -6.211620e+03                0.000000e+00
25%            0.000000e+00                0.000000e+00
50%            0.000000e+00                0.000000e+00
75%            0.000000e+00                0.000000e+00
max            2.287098e+06                1.529847e+06
```

	AMT_DRAWINGS_POS_CURRENT	AMT_INST_MIN_REGULARITY	...	\
count	3.090496e+06	3.535076e+06	...	
mean	2.968805e+03	3.540204e+03	...	
std	2.079689e+04	5.600154e+03	...	
min	0.000000e+00	0.000000e+00	...	
25%	0.000000e+00	0.000000e+00	...	
50%	0.000000e+00	0.000000e+00	...	
75%	0.000000e+00	6.633911e+03	...	
max	2.239274e+06	2.028820e+05	...	

	AMT_RECEIVABLE_PRINCIPAL	AMT_RECIVABLE	AMT_TOTAL_RECEIVABLE	\
count	3.840312e+06	3.840312e+06	3.840312e+06	
mean	5.596588e+04	5.808881e+04	5.809829e+04	
std	1.025336e+05	1.059654e+05	1.059718e+05	
min	-4.233058e+05	-4.202502e+05	-4.202502e+05	
25%	0.000000e+00	0.000000e+00	0.000000e+00	
50%	0.000000e+00	0.000000e+00	0.000000e+00	
75%	8.535924e+04	8.889949e+04	8.891451e+04	
max	1.472317e+06	1.493338e+06	1.493338e+06	

	CNT_DRAWINGS_ATM_CURRENT	CNT_DRAWINGS_CURRENT	\
count	3.090496e+06	3.840312e+06	
mean	3.094490e-01	7.031439e-01	
std	1.100401e+00	3.190347e+00	
min	0.000000e+00	0.000000e+00	
25%	0.000000e+00	0.000000e+00	
50%	0.000000e+00	0.000000e+00	
75%	0.000000e+00	0.000000e+00	
max	5.100000e+01	1.650000e+02	

	CNT_DRAWINGS_OTHER_CURRENT	CNT_DRAWINGS_POS_CURRENT	\
count	3.090496e+06	3.090496e+06	
mean	4.812496e-03	5.594791e-01	
std	8.263861e-02	3.240649e+00	
min	0.000000e+00	0.000000e+00	
25%	0.000000e+00	0.000000e+00	
50%	0.000000e+00	0.000000e+00	
75%	0.000000e+00	0.000000e+00	
max	1.200000e+01	1.650000e+02	

	CNT_INSTALMENT_MATURE_CUM	SK_DPD	SK_DPD_DEF
count	3.535076e+06	3.840312e+06	3.840312e+06
mean	2.082508e+01	9.283667e+00	3.316220e-01
std	2.005149e+01	9.751570e+01	2.147923e+01
min	0.000000e+00	0.000000e+00	0.000000e+00
25%	4.000000e+00	0.000000e+00	0.000000e+00
50%	1.500000e+01	0.000000e+00	0.000000e+00
75%	3.200000e+01	0.000000e+00	0.000000e+00
max	1.200000e+02	3.260000e+03	3.260000e+03

[8 rows x 22 columns]

==========

==========

Data Frame: Correlation Statistics

```
<ipython-input-264-21e78107dac0>:83: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  print(df.corr())
```

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE
SK_ID_PREV	1.000000	0.004723	0.003670
SK_ID_CURR	0.004723	1.000000	0.001696
MONTHS_BALANCE	0.003670	0.001696	1.000000
AMT_BALANCE	0.005046	0.003510	0.014558
AMT_CREDIT_LIMIT_ACTUAL	0.006631	0.005991	0.199900
AMT_DRAWINGS_ATM_CURRENT	0.004342	0.000814	0.036802
AMT_DRAWINGS_CURRENT	0.002624	0.000708	0.065527
AMT_DRAWINGS_OTHER_CURRENT	-0.000160	0.000958	0.000405
AMT_DRAWINGS_POS_CURRENT	0.001721	-0.000786	0.118146
AMT_INST_MIN_REGULARITY	0.006460	0.003300	-0.087529
AMT_PAYMENT_CURRENT	0.003472	0.000127	0.076355
AMT_PAYMENT_TOTAL_CURRENT	0.001641	0.000784	0.035614
AMT_RECEIVABLE_PRINCIPAL	0.005140	0.003589	0.016266
AMT_RECIVABLE	0.005035	0.003518	0.013172
AMT_TOTAL_RECEIVABLE	0.005032	0.003524	0.013084
CNT_DRAWINGS_ATM_CURRENT	0.002821	0.002082	0.002536
CNT_DRAWINGS_CURRENT	0.000367	0.002654	0.113321
CNT_DRAWINGS_OTHER_CURRENT	-0.001412	-0.000131	-0.026192
CNT_DRAWINGS_POS_CURRENT	0.000809	0.002135	0.160207
CNT_INSTALMENT_MATURE_CUM	-0.007219	-0.000581	-0.008620
SK_DPD	-0.001786	-0.000962	0.039434
SK_DPD_DEF	0.001973	0.001519	0.001659

	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL
SK_ID_PREV	0.005046	0.006631
SK_ID_CURR	0.003510	0.005991
MONTHS_BALANCE	0.014558	0.199900
AMT_BALANCE	1.000000	0.489386
AMT_CREDIT_LIMIT_ACTUAL	0.489386	1.000000
AMT_DRAWINGS_ATM_CURRENT	0.283551	0.247219
AMT_DRAWINGS_CURRENT	0.336965	0.263093
AMT_DRAWINGS_OTHER_CURRENT	0.065366	0.050579
AMT_DRAWINGS_POS_CURRENT	0.169449	0.234976
AMT_INST_MIN_REGULARITY	0.896728	0.467620
AMT_PAYMENT_CURRENT	0.143934	0.308294
AMT_PAYMENT_TOTAL_CURRENT	0.151349	0.226570
AMT_RECEIVABLE_PRINCIPAL	0.999720	0.490445
AMT_RECIVABLE	0.999917	0.488641
AMT_TOTAL_RECEIVABLE	0.999897	0.488598
CNT_DRAWINGS_ATM_CURRENT	0.309968	0.221808
CNT_DRAWINGS_CURRENT	0.259184	0.204237
CNT_DRAWINGS_OTHER_CURRENT	0.046563	0.030051
CNT_DRAWINGS_POS_CURRENT	0.155553	0.202868
CNT_INSTALMENT_MATURE_CUM	0.005009	-0.157269
SK_DPD	-0.046988	-0.038791
SK_DPD_DEF	0.013009	-0.002236

	AMT_DRAWINGS_ATM_CURRENT	AMT_DRAWINGS_CURRENT
SK_ID_PREV	0.004342	0.002624
SK_ID_CURR	0.000814	0.000708
MONTHS_BALANCE	0.036802	0.065527
AMT_BALANCE	0.283551	0.336965
AMT_CREDIT_LIMIT_ACTUAL	0.247219	0.263093
AMT_DRAWINGS_ATM_CURRENT	1.000000	0.800190
AMT_DRAWINGS_CURRENT	0.800190	1.000000
AMT_DRAWINGS_OTHER_CURRENT	0.017899	0.236297
AMT_DRAWINGS_POS_CURRENT	0.078971	0.615591
AMT_INST_MIN_REGULARITY	0.094824	0.124469
AMT_PAYMENT_CURRENT	0.189075	0.337343

AMT_PAYMENT_TOTAL_CURRENT	0.159186	0.305726
AMT_RECEIVABLE_PRINCIPAL	0.280402	0.337117
AMT_RECIVABLE	0.278290	0.332831
AMT_TOTAL_RECEIVABLE	0.278260	0.332796
CNT_DRAWINGS_ATM_CURRENT	0.732907	0.594361
CNT_DRAWINGS_CURRENT	0.298173	0.523016
CNT_DRAWINGS_OTHER_CURRENT	0.013254	0.140032
CNT_DRAWINGS_POS_CURRENT	0.076083	0.359001
CNT_INSTALMENT_MATURE_CUM	-0.103721	-0.093491
SK_DPD	-0.022044	-0.020606
SK_DPD_DEF	-0.003360	-0.003137

	AMT_DRAWINGS_OTHER_CURRENT \
SK_ID_PREV	-0.000160
SK_ID_CURR	0.000958
MONTHS_BALANCE	0.000405
AMT_BALANCE	0.065366
AMT_CREDIT_LIMIT_ACTUAL	0.050579
AMT_DRAWINGS_ATM_CURRENT	0.017899
AMT_DRAWINGS_CURRENT	0.236297
AMT_DRAWINGS_OTHER_CURRENT	1.000000
AMT_DRAWINGS_POS_CURRENT	0.007382
AMT_INST_MIN_REGULARITY	0.002158
AMT_PAYMENT_CURRENT	0.034577
AMT_PAYMENT_TOTAL_CURRENT	0.025123
AMT_RECEIVABLE_PRINCIPAL	0.066108
AMT_RECIVABLE	0.064929
AMT_TOTAL_RECEIVABLE	0.064923
CNT_DRAWINGS_ATM_CURRENT	0.012008
CNT_DRAWINGS_CURRENT	0.021271
CNT_DRAWINGS_OTHER_CURRENT	0.575295
CNT_DRAWINGS_POS_CURRENT	0.004458
CNT_INSTALMENT_MATURE_CUM	-0.023013
SK_DPD	-0.003693
SK_DPD_DEF	-0.000568

	AMT_DRAWINGS_POS_CURRENT	AMT_INST_MIN_REGULARITY \
SK_ID_PREV	0.001721	0.006460
SK_ID_CURR	-0.000786	0.003300
MONTHS_BALANCE	0.118146	-0.087529
AMT_BALANCE	0.169449	0.896728
AMT_CREDIT_LIMIT_ACTUAL	0.234976	0.467620
AMT_DRAWINGS_ATM_CURRENT	0.078971	0.094824
AMT_DRAWINGS_CURRENT	0.615591	0.124469
AMT_DRAWINGS_OTHER_CURRENT	0.007382	0.002158
AMT_DRAWINGS_POS_CURRENT	1.000000	0.063562
AMT_INST_MIN_REGULARITY	0.063562	1.000000
AMT_PAYMENT_CURRENT	0.321055	0.333909
AMT_PAYMENT_TOTAL_CURRENT	0.301760	0.335201
AMT_RECEIVABLE_PRINCIPAL	0.173745	0.896030
AMT_RECIVABLE	0.168974	0.897617
AMT_TOTAL_RECEIVABLE	0.168950	0.897587
CNT_DRAWINGS_ATM_CURRENT	0.072658	0.170616
CNT_DRAWINGS_CURRENT	0.520123	0.148262
CNT_DRAWINGS_OTHER_CURRENT	0.007620	0.014360
CNT_DRAWINGS_POS_CURRENT	0.542556	0.086729
CNT_INSTALMENT_MATURE_CUM	-0.106813	0.064320
SK_DPD	-0.015040	-0.061484
SK_DPD_DEF	-0.002384	-0.005715

	...	AMT_RECEIVABLE_PRINCIPAL	AMT_RECIVABLE
SK_ID_PREV	...	0.005140	0.005035
SK_ID_CURR	...	0.003589	0.003518
MONTHS_BALANCE	...	0.016266	0.013172
AMT_BALANCE	...	0.999720	0.999917
AMT_CREDIT_LIMIT_ACTUAL	...	0.490445	0.488641
AMT_DRAWINGS_ATM_CURRENT	...	0.280402	0.278290
AMT_DRAWINGS_CURRENT	...	0.337117	0.332831
AMT_DRAWINGS_OTHER_CURRENT	...	0.066108	0.064929
AMT_DRAWINGS_POS_CURRENT	...	0.173745	0.168974
AMT_INST_MIN_REGULARITY	...	0.896030	0.897617
AMT_PAYMENT_CURRENT	...	0.143162	0.142389
AMT_PAYMENT_TOTAL_CURRENT	...	0.149936	0.149926
AMT_RECEIVABLE_PRINCIPAL	...	1.000000	0.999727
AMT_RECIVABLE	...	0.999727	1.000000
AMT_TOTAL_RECEIVABLE	...	0.999702	0.999995
CNT_DRAWINGS_ATM_CURRENT	...	0.302627	0.303571
CNT_DRAWINGS_CURRENT	...	0.258848	0.256347
CNT_DRAWINGS_OTHER_CURRENT	...	0.046543	0.046118
CNT_DRAWINGS_POS_CURRENT	...	0.157723	0.154507
CNT_INSTALMENT_MATURE_CUM	...	0.003664	0.005935
SK_DPD	...	-0.048290	-0.046434
SK_DPD_DEF	...	0.006780	0.015466

	AMT_TOTAL_RECEIVABLE	CNT_DRAWINGS_ATM_CURRENT
SK_ID_PREV	0.005032	0.002821
SK_ID_CURR	0.003524	0.002082
MONTHS_BALANCE	0.013084	0.002536
AMT_BALANCE	0.999897	0.309968
AMT_CREDIT_LIMIT_ACTUAL	0.488598	0.221808
AMT_DRAWINGS_ATM_CURRENT	0.278260	0.732907
AMT_DRAWINGS_CURRENT	0.332796	0.594361
AMT_DRAWINGS_OTHER_CURRENT	0.064923	0.012008
AMT_DRAWINGS_POS_CURRENT	0.168950	0.072658
AMT_INST_MIN_REGULARITY	0.897587	0.170616
AMT_PAYMENT_CURRENT	0.142371	0.142935
AMT_PAYMENT_TOTAL_CURRENT	0.149914	0.125655
AMT_RECEIVABLE_PRINCIPAL	0.999702	0.302627
AMT_RECIVABLE	0.999995	0.303571
AMT_TOTAL_RECEIVABLE	1.000000	0.303542
CNT_DRAWINGS_ATM_CURRENT	0.303542	1.000000
CNT_DRAWINGS_CURRENT	0.256317	0.410907
CNT_DRAWINGS_OTHER_CURRENT	0.046113	0.012730
CNT_DRAWINGS_POS_CURRENT	0.154481	0.108388
CNT_INSTALMENT_MATURE_CUM	0.005959	-0.103403
SK_DPD	-0.046047	-0.029395
SK_DPD_DEF	0.017243	-0.004277

	CNT_DRAWINGS_CURRENT	CNT_DRAWINGS_OTHER_CURRENT
SK_ID_PREV	0.000367	-0.001412
SK_ID_CURR	0.002654	-0.000131
MONTHS_BALANCE	0.113321	-0.026192
AMT_BALANCE	0.259184	0.046563
AMT_CREDIT_LIMIT_ACTUAL	0.204237	0.030051
AMT_DRAWINGS_ATM_CURRENT	0.298173	0.013254
AMT_DRAWINGS_CURRENT	0.523016	0.140032
AMT_DRAWINGS_OTHER_CURRENT	0.021271	0.575295
AMT_DRAWINGS_POS_CURRENT	0.520123	0.007620
AMT_INST_MIN_REGULARITY	0.148262	0.014360
AMT_PAYMENT_CURRENT	0.223483	0.017246

AMT_PAYMENT_TOTAL_CURRENT	0.217857	0.014041
AMT_RECEIVABLE_PRINCIPAL	0.258848	0.046543
AMT_RECIVABLE	0.256347	0.046118
AMT_TOTAL_RECEIVABLE	0.256317	0.046113
CNT_DRAWINGS_ATM_CURRENT	0.410907	0.012730
CNT_DRAWINGS_CURRENT	1.000000	0.033940
CNT_DRAWINGS_OTHER_CURRENT	0.033940	1.000000
CNT_DRAWINGS_POS_CURRENT	0.950546	0.007203
CNT_INSTALMENT_MATURE_CUM	-0.099186	-0.021632
SK_DPD	-0.020786	-0.006083
SK_DPD_DEF	-0.003106	-0.000895

	CNT_DRAWINGS_POS_CURRENT \
SK_ID_PREV	0.000809
SK_ID_CURR	0.002135
MONTHS_BALANCE	0.160207
AMT_BALANCE	0.155553
AMT_CREDIT_LIMIT_ACTUAL	0.202868
AMT_DRAWINGS_ATM_CURRENT	0.076083
AMT_DRAWINGS_CURRENT	0.359001
AMT_DRAWINGS_OTHER_CURRENT	0.004458
AMT_DRAWINGS_POS_CURRENT	0.542556
AMT_INST_MIN_REGULARITY	0.086729
AMT_PAYMENT_CURRENT	0.195074
AMT_PAYMENT_TOTAL_CURRENT	0.183973
AMT_RECEIVABLE_PRINCIPAL	0.157723
AMT_RECIVABLE	0.154507
AMT_TOTAL_RECEIVABLE	0.154481
CNT_DRAWINGS_ATM_CURRENT	0.108388
CNT_DRAWINGS_CURRENT	0.950546
CNT_DRAWINGS_OTHER_CURRENT	0.007203
CNT_DRAWINGS_POS_CURRENT	1.000000
CNT_INSTALMENT_MATURE_CUM	-0.129338
SK_DPD	-0.018212
SK_DPD_DEF	-0.002840

	CNT_INSTALMENT_MATURE_CUM	SK_DPD	SK_DPD_DEF
SK_ID_PREV	-0.007219	-0.001786	0.001973
SK_ID_CURR	-0.000581	-0.000962	0.001519
MONTHS_BALANCE	-0.008620	0.039434	0.001659
AMT_BALANCE	0.005009	-0.046988	0.013009
AMT_CREDIT_LIMIT_ACTUAL	-0.157269	-0.038791	-0.002236
AMT_DRAWINGS_ATM_CURRENT	-0.103721	-0.022044	-0.003360
AMT_DRAWINGS_CURRENT	-0.093491	-0.020606	-0.003137
AMT_DRAWINGS_OTHER_CURRENT	-0.023013	-0.003693	-0.000568
AMT_DRAWINGS_POS_CURRENT	-0.106813	-0.015040	-0.002384
AMT_INST_MIN_REGULARITY	0.064320	-0.061484	-0.005715
AMT_PAYMENT_CURRENT	-0.079266	-0.030222	-0.004340
AMT_PAYMENT_TOTAL_CURRENT	-0.023156	-0.022475	-0.003443
AMT_RECEIVABLE_PRINCIPAL	0.003664	-0.048290	0.006780
AMT_RECIVABLE	0.005935	-0.046434	0.015466
AMT_TOTAL_RECEIVABLE	0.005959	-0.046047	0.017243
CNT_DRAWINGS_ATM_CURRENT	-0.103403	-0.029395	-0.004277
CNT_DRAWINGS_CURRENT	-0.099186	-0.020786	-0.003106
CNT_DRAWINGS_OTHER_CURRENT	-0.021632	-0.006083	-0.000895
CNT_DRAWINGS_POS_CURRENT	-0.129338	-0.018212	-0.002840
CNT_INSTALMENT_MATURE_CUM	1.000000	0.059654	0.002156
SK_DPD	0.059654	1.000000	0.218950
SK_DPD_DEF	0.002156	0.218950	1.000000

```
[22 rows x 22 columns]
================================================


================================================
Data Frame: Additional Information
------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3840312 entries, 0 to 3840311
Data columns (total 23 columns):
 #   Column                      Dtype
---  ------                      -----
 0   SK_ID_PREV                  int64
 1   SK_ID_CURR                  int64
 2   MONTHS_BALANCE              int64
 3   AMT_BALANCE                 float64
 4   AMT_CREDIT_LIMIT_ACTUAL     int64
 5   AMT_DRAWINGS_ATM_CURRENT    float64
 6   AMT_DRAWINGS_CURRENT        float64
 7   AMT_DRAWINGS_OTHER_CURRENT  float64
 8   AMT_DRAWINGS_POS_CURRENT    float64
 9   AMT_INST_MIN_REGULARITY     float64
 10  AMT_PAYMENT_CURRENT         float64
 11  AMT_PAYMENT_TOTAL_CURRENT   float64
 12  AMT_RECEIVABLE_PRINCIPAL    float64
 13  AMT_RECIVABLE               float64
 14  AMT_TOTAL_RECEIVABLE        float64
 15  CNT_DRAWINGS_ATM_CURRENT    float64
 16  CNT_DRAWINGS_CURRENT        int64
 17  CNT_DRAWINGS_OTHER_CURRENT  float64
 18  CNT_DRAWINGS_POS_CURRENT    float64
 19  CNT_INSTALMENT_MATURE_CUM   float64
 20  NAME_CONTRACT_STATUS        object
 21  SK_DPD                      int64
 22  SK_DPD_DEF                  int64
dtypes: float64(15), int64(7), object(1)
memory usage: 673.9+ MB
None
================================================
```

Data Dictionary: previous_application.csv

In [ ]:

```
# Entering information to call the EDA Method
eda_info_pre_app = ['Previous Application', df_pre_app]

# Calling EDA Method
EDA(eda_info_pre_app)
```

```
**********************************************

           DATAFRAME: Previous Application

**********************************************


================================================
Data Frame: Size, Shape & Total Missing Values
------------------------------------------------
Number of Rows: 1670214
Number of Columns: 37
Number of Total Missing Values: 11109336
Data Frame Shape: (1670214, 37)
================================================


================================================
Data Frame: Missing Values by Feature
------------------------------------------------
Number of Missing Values by Feature: SK_ID_PREV                           0
SK_ID_CURR                             0
NAME_CONTRACT_TYPE                     0
AMT_ANNUITY                       372235
AMT_APPLICATION                        0
AMT_CREDIT                             1
AMT_DOWN_PAYMENT                  895844
AMT_GOODS_PRICE                   385515
WEEKDAY_APPR_PROCESS_START             0
HOUR_APPR_PROCESS_START                0
FLAG_LAST_APPL_PER_CONTRACT            0
NFLAG_LAST_APPL_IN_DAY                 0
RATE_DOWN_PAYMENT                 895844
RATE_INTEREST_PRIMARY            1664263
RATE_INTEREST_PRIVILEGED         1664263
NAME_CASH_LOAN_PURPOSE                 0
NAME_CONTRACT_STATUS                   0
DAYS_DECISION                          0
NAME_PAYMENT_TYPE                      0
CODE_REJECT_REASON                     0
NAME_TYPE_SUITE                   820405
NAME_CLIENT_TYPE                       0
NAME_GOODS_CATEGORY                    0
NAME_PORTFOLIO                         0
NAME_PRODUCT_TYPE                      0
CHANNEL_TYPE                           0
SELLERPLACE_AREA                       0
NAME_SELLER_INDUSTRY                   0
CNT_PAYMENT                       372230
NAME_YIELD_GROUP                       0
PRODUCT_COMBINATION                  346
DAYS_FIRST_DRAWING                673065
DAYS_FIRST_DUE                    673065
DAYS_LAST_DUE_1ST_VERSION         673065
DAYS_LAST_DUE                     673065
DAYS_TERMINATION                  673065
NFLAG_INSURED_ON_APPROVAL         673065
dtype: int64
================================================
```

```
================================================
Data Frame: Data Types
------------------------------------------------
SK_ID_PREV                      int64
SK_ID_CURR                      int64
NAME_CONTRACT_TYPE             object
AMT_ANNUITY                   float64
AMT_APPLICATION               float64
AMT_CREDIT                    float64
AMT_DOWN_PAYMENT              float64
AMT_GOODS_PRICE               float64
WEEKDAY_APPR_PROCESS_START     object
HOUR_APPR_PROCESS_START         int64
FLAG_LAST_APPL_PER_CONTRACT    object
NFLAG_LAST_APPL_IN_DAY          int64
RATE_DOWN_PAYMENT             float64
RATE_INTEREST_PRIMARY         float64
RATE_INTEREST_PRIVILEGED      float64
NAME_CASH_LOAN_PURPOSE         object
NAME_CONTRACT_STATUS           object
DAYS_DECISION                   int64
NAME_PAYMENT_TYPE              object
CODE_REJECT_REASON             object
NAME_TYPE_SUITE                object
NAME_CLIENT_TYPE               object
NAME_GOODS_CATEGORY            object
NAME_PORTFOLIO                 object
NAME_PRODUCT_TYPE              object
CHANNEL_TYPE                   object
SELLERPLACE_AREA                int64
NAME_SELLER_INDUSTRY           object
CNT_PAYMENT                   float64
NAME_YIELD_GROUP               object
PRODUCT_COMBINATION            object
DAYS_FIRST_DRAWING            float64
DAYS_FIRST_DUE                float64
DAYS_LAST_DUE_1ST_VERSION     float64
DAYS_LAST_DUE                 float64
DAYS_TERMINATION              float64
NFLAG_INSURED_ON_APPROVAL     float64
dtype: object
================================================


================================================
Data Frame: Data Types
------------------------------------------------
object     16
float64    15
int64       6
dtype: int64
================================================


================================================
Data Frame: Summary Statistics
------------------------------------------------
         SK_ID_PREV    SK_ID_CURR   AMT_ANNUITY  AMT_APPLICATION  \
count  1.670214e+06  1.670214e+06  1.297979e+06     1.670214e+06
```

mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06

	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	\
count	1.670213e+06	7.743700e+05	1.284699e+06	
mean	1.961140e+05	6.697402e+03	2.278473e+05	
std	3.185746e+05	2.092150e+04	3.153966e+05	
min	0.000000e+00	-9.000000e-01	0.000000e+00	
25%	2.416050e+04	0.000000e+00	5.084100e+04	
50%	8.054100e+04	1.638000e+03	1.123200e+05	
75%	2.164185e+05	7.740000e+03	2.340000e+05	
max	6.905160e+06	3.060045e+06	6.905160e+06	

	HOUR_APPR_PROCESS_START	NFLAG_LAST_APPL_IN_DAY	RATE_DOWN_PAYMENT	\
count	1.670214e+06	1.670214e+06	774370.000000	
mean	1.248418e+01	9.964675e-01	0.079637	
std	3.334028e+00	5.932963e-02	0.107823	
min	0.000000e+00	0.000000e+00	-0.000015	
25%	1.000000e+01	1.000000e+00	0.000000	
50%	1.200000e+01	1.000000e+00	0.051605	
75%	1.500000e+01	1.000000e+00	0.108909	
max	2.300000e+01	1.000000e+00	1.000000	

	...	RATE_INTEREST_PRIVILEGED	DAYS_DECISION	SELLERPLACE_AREA	\
count	...	5951.000000	1.670214e+06	1.670214e+06	
mean	...	0.773503	-8.806797e+02	3.139511e+02	
std	...	0.100879	7.790997e+02	7.127443e+03	
min	...	0.373150	-2.922000e+03	-1.000000e+00	
25%	...	0.715645	-1.300000e+03	-1.000000e+00	
50%	...	0.835095	-5.810000e+02	3.000000e+00	
75%	...	0.852537	-2.800000e+02	8.200000e+01	
max	...	1.000000	-1.000000e+00	4.000000e+06	

	CNT_PAYMENT	DAYS_FIRST_DRAWING	DAYS_FIRST_DUE	\
count	1.297984e+06	997149.000000	997149.000000	
mean	1.605408e+01	342209.855039	13826.269337	
std	1.456729e+01	88916.115833	72444.869708	
min	0.000000e+00	-2922.000000	-2892.000000	
25%	6.000000e+00	365243.000000	-1628.000000	
50%	1.200000e+01	365243.000000	-831.000000	
75%	2.400000e+01	365243.000000	-411.000000	
max	8.400000e+01	365243.000000	365243.000000	

	DAYS_LAST_DUE_1ST_VERSION	DAYS_LAST_DUE	DAYS_TERMINATION	\
count	997149.000000	997149.000000	997149.000000	
mean	33767.774054	76582.403064	81992.343838	
std	106857.034789	149647.415123	153303.516729	
min	-2801.000000	-2889.000000	-2874.000000	
25%	-1242.000000	-1314.000000	-1270.000000	
50%	-361.000000	-537.000000	-499.000000	
75%	129.000000	-74.000000	-44.000000	
max	365243.000000	365243.000000	365243.000000	

	NFLAG_INSURED_ON_APPROVAL
count	997149.000000

```
mean                    0.332570
std                     0.471134
min                     0.000000
25%                     0.000000
50%                     0.000000
75%                     1.000000
max                     1.000000

[8 rows x 21 columns]
================================================


================================================
Data Frame: Correlation Statistics
------------------------------------------------
<ipython-input-264-21e78107dac0>:83: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  print(df.corr())
```

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY
SK_ID_PREV	1.000000	-0.000321	0.011459
SK_ID_CURR	-0.000321	1.000000	0.000577
AMT_ANNUITY	0.011459	0.000577	1.000000
AMT_APPLICATION	0.003302	0.000280	0.808872
AMT_CREDIT	0.003659	0.000195	0.816429
AMT_DOWN_PAYMENT	-0.001313	-0.000063	0.267694
AMT_GOODS_PRICE	0.015293	0.000369	0.820895
HOUR_APPR_PROCESS_START	-0.002652	0.002842	-0.036201
NFLAG_LAST_APPL_IN_DAY	-0.002828	0.000098	0.020639
RATE_DOWN_PAYMENT	-0.004051	0.001158	-0.103878
RATE_INTEREST_PRIMARY	0.012969	0.033197	0.141823
RATE_INTEREST_PRIVILEGED	-0.022312	-0.016757	-0.202335
DAYS_DECISION	0.019100	-0.000637	0.279051
SELLERPLACE_AREA	-0.001079	0.001265	-0.015027
CNT_PAYMENT	0.015589	0.000031	0.394535
DAYS_FIRST_DRAWING	-0.001478	-0.001329	0.052839
DAYS_FIRST_DUE	-0.000071	-0.000757	-0.053295
DAYS_LAST_DUE_1ST_VERSION	0.001222	0.000252	-0.068877
DAYS_LAST_DUE	0.001915	-0.000318	0.082659
DAYS_TERMINATION	0.001781	-0.000020	0.068022
NFLAG_INSURED_ON_APPROVAL	0.003986	0.000876	0.283080

	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT
SK_ID_PREV	0.003302	0.003659	-0.001313
SK_ID_CURR	0.000280	0.000195	-0.000063
AMT_ANNUITY	0.808872	0.816429	0.267694
AMT_APPLICATION	1.000000	0.975824	0.482776
AMT_CREDIT	0.975824	1.000000	0.301284
AMT_DOWN_PAYMENT	0.482776	0.301284	1.000000
AMT_GOODS_PRICE	0.999884	0.993087	0.482776
HOUR_APPR_PROCESS_START	-0.014415	-0.021039	0.016776
NFLAG_LAST_APPL_IN_DAY	0.004310	-0.025179	0.001597
RATE_DOWN_PAYMENT	-0.072479	-0.188128	0.473935
RATE_INTEREST_PRIMARY	0.110001	0.125106	0.016323
RATE_INTEREST_PRIVILEGED	-0.199733	-0.205158	-0.115343
DAYS_DECISION	0.133660	0.133763	-0.024536
SELLERPLACE_AREA	-0.007649	-0.009567	0.003533
CNT_PAYMENT	0.680630	0.674278	0.031659
DAYS_FIRST_DRAWING	0.074544	-0.036813	-0.001773
DAYS_FIRST_DUE	-0.049532	0.002881	-0.013586
DAYS_LAST_DUE_1ST_VERSION	-0.084905	0.044031	-0.000869
DAYS_LAST_DUE	0.172627	0.224829	-0.031425
DAYS_TERMINATION	0.148618	0.214320	-0.030702
NFLAG_INSURED_ON_APPROVAL	0.259219	0.263932	-0.042585

	AMT_GOODS_PRICE	HOUR_APPR_PROCESS_START
SK_ID_PREV	0.015293	-0.002652
SK_ID_CURR	0.000369	0.002842
AMT_ANNUITY	0.820895	-0.036201
AMT_APPLICATION	0.999884	-0.014415
AMT_CREDIT	0.993087	-0.021039
AMT_DOWN_PAYMENT	0.482776	0.016776
AMT_GOODS_PRICE	1.000000	-0.045267
HOUR_APPR_PROCESS_START	-0.045267	1.000000
NFLAG_LAST_APPL_IN_DAY	-0.017100	0.005789
RATE_DOWN_PAYMENT	-0.072479	0.025930
RATE_INTEREST_PRIMARY	0.110001	-0.027172
RATE_INTEREST_PRIVILEGED	-0.199733	-0.045720
DAYS_DECISION	0.290422	-0.039962

SELLERPLACE_AREA	-0.015842	0.015671
CNT_PAYMENT	0.672129	-0.055511
DAYS_FIRST_DRAWING	-0.024445	0.014321
DAYS_FIRST_DUE	-0.021062	-0.002797
DAYS_LAST_DUE_1ST_VERSION	0.016883	-0.016567
DAYS_LAST_DUE	0.211696	-0.018018
DAYS_TERMINATION	0.209296	-0.018254
NFLAG_INSURED_ON_APPROVAL	0.243400	-0.117318

	NFLAG_LAST_APPL_IN_DAY	RATE_DOWN_PAYMENT	... \
SK_ID_PREV	-0.002828	-0.004051	...
SK_ID_CURR	0.000098	0.001158	...
AMT_ANNUITY	0.020639	-0.103878	...
AMT_APPLICATION	0.004310	-0.072479	...
AMT_CREDIT	-0.025179	-0.188128	...
AMT_DOWN_PAYMENT	0.001597	0.473935	...
AMT_GOODS_PRICE	-0.017100	-0.072479	...
HOUR_APPR_PROCESS_START	0.005789	0.025930	...
NFLAG_LAST_APPL_IN_DAY	1.000000	0.004554	...
RATE_DOWN_PAYMENT	0.004554	1.000000	...
RATE_INTEREST_PRIMARY	0.009604	-0.103373	...
RATE_INTEREST_PRIVILEGED	0.024640	-0.106143	...
DAYS_DECISION	0.016555	-0.208742	...
SELLERPLACE_AREA	0.000912	-0.006489	...
CNT_PAYMENT	0.063347	-0.278875	...
DAYS_FIRST_DRAWING	-0.000409	-0.007969	...
DAYS_FIRST_DUE	-0.002288	-0.039178	...
DAYS_LAST_DUE_1ST_VERSION	-0.001981	-0.010934	...
DAYS_LAST_DUE	-0.002277	-0.147562	...
DAYS_TERMINATION	-0.000744	-0.145461	...
NFLAG_INSURED_ON_APPROVAL	-0.007124	-0.021633	...

	RATE_INTEREST_PRIVILEGED	DAYS_DECISION \
SK_ID_PREV	-0.022312	0.019100
SK_ID_CURR	-0.016757	-0.000637
AMT_ANNUITY	-0.202335	0.279051
AMT_APPLICATION	-0.199733	0.133660
AMT_CREDIT	-0.205158	0.133763
AMT_DOWN_PAYMENT	-0.115343	-0.024536
AMT_GOODS_PRICE	-0.199733	0.290422
HOUR_APPR_PROCESS_START	-0.045720	-0.039962
NFLAG_LAST_APPL_IN_DAY	0.024640	0.016555
RATE_DOWN_PAYMENT	-0.106143	-0.208742
RATE_INTEREST_PRIMARY	-0.001937	0.014037
RATE_INTEREST_PRIVILEGED	1.000000	0.631940
DAYS_DECISION	0.631940	1.000000
SELLERPLACE_AREA	-0.066316	-0.018382
CNT_PAYMENT	-0.057150	0.246453
DAYS_FIRST_DRAWING	NaN	-0.012007
DAYS_FIRST_DUE	0.150904	0.176711
DAYS_LAST_DUE_1ST_VERSION	0.030513	0.089167
DAYS_LAST_DUE	0.372214	0.448549
DAYS_TERMINATION	0.378671	0.400179
NFLAG_INSURED_ON_APPROVAL	-0.067157	-0.028905

	SELLERPLACE_AREA	CNT_PAYMENT	DAYS_FIRST_DRAWING \
SK_ID_PREV	-0.001079	0.015589	-0.001478
SK_ID_CURR	0.001265	0.000031	-0.001329
AMT_ANNUITY	-0.015027	0.394535	0.052839
AMT_APPLICATION	-0.007649	0.680630	0.074544

AMT_CREDIT	-0.009567	0.674278	-0.036813
AMT_DOWN_PAYMENT	0.003533	0.031659	-0.001773
AMT_GOODS_PRICE	-0.015842	0.672129	-0.024445
HOUR_APPR_PROCESS_START	0.015671	-0.055511	0.014321
NFLAG_LAST_APPL_IN_DAY	0.000912	0.063347	-0.000409
RATE_DOWN_PAYMENT	-0.006489	-0.278875	-0.007969
RATE_INTEREST_PRIMARY	0.159182	-0.019030	NaN
RATE_INTEREST_PRIVILEGED	-0.066316	-0.057150	NaN
DAYS_DECISION	-0.018382	0.246453	-0.012007
SELLERPLACE_AREA	1.000000	-0.010646	0.007401
CNT_PAYMENT	-0.010646	1.000000	0.309900
DAYS_FIRST_DRAWING	0.007401	0.309900	1.000000
DAYS_FIRST_DUE	-0.002166	-0.204907	0.004710
DAYS_LAST_DUE_1ST_VERSION	-0.007510	-0.381013	-0.803494
DAYS_LAST_DUE	-0.006291	0.088903	-0.257466
DAYS_TERMINATION	-0.006675	0.055121	-0.396284
NFLAG_INSURED_ON_APPROVAL	-0.018280	0.320520	0.177652

	DAYS_FIRST_DUE	DAYS_LAST_DUE_1ST_VERSION	\
SK_ID_PREV	-0.000071	0.001222	
SK_ID_CURR	-0.000757	0.000252	
AMT_ANNUITY	-0.053295	-0.068877	
AMT_APPLICATION	-0.049532	-0.084905	
AMT_CREDIT	0.002881	0.044031	
AMT_DOWN_PAYMENT	-0.013586	-0.000869	
AMT_GOODS_PRICE	-0.021062	0.016883	
HOUR_APPR_PROCESS_START	-0.002797	-0.016567	
NFLAG_LAST_APPL_IN_DAY	-0.002288	-0.001981	
RATE_DOWN_PAYMENT	-0.039178	-0.010934	
RATE_INTEREST_PRIMARY	-0.017171	-0.000933	
RATE_INTEREST_PRIVILEGED	0.150904	0.030513	
DAYS_DECISION	0.176711	0.089167	
SELLERPLACE_AREA	-0.002166	-0.007510	
CNT_PAYMENT	-0.204907	-0.381013	
DAYS_FIRST_DRAWING	0.004710	-0.803494	
DAYS_FIRST_DUE	1.000000	0.513949	
DAYS_LAST_DUE_1ST_VERSION	0.513949	1.000000	
DAYS_LAST_DUE	0.401838	0.423462	
DAYS_TERMINATION	0.323608	0.493174	
NFLAG_INSURED_ON_APPROVAL	-0.119048	-0.221947	

	DAYS_LAST_DUE	DAYS_TERMINATION	\
SK_ID_PREV	0.001915	0.001781	
SK_ID_CURR	-0.000318	-0.000020	
AMT_ANNUITY	0.082659	0.068022	
AMT_APPLICATION	0.172627	0.148618	
AMT_CREDIT	0.224829	0.214320	
AMT_DOWN_PAYMENT	-0.031425	-0.030702	
AMT_GOODS_PRICE	0.211696	0.209296	
HOUR_APPR_PROCESS_START	-0.018018	-0.018254	
NFLAG_LAST_APPL_IN_DAY	-0.002277	-0.000744	
RATE_DOWN_PAYMENT	-0.147562	-0.145461	
RATE_INTEREST_PRIMARY	-0.010677	-0.011099	
RATE_INTEREST_PRIVILEGED	0.372214	0.378671	
DAYS_DECISION	0.448549	0.400179	
SELLERPLACE_AREA	-0.006291	-0.006675	
CNT_PAYMENT	0.088903	0.055121	
DAYS_FIRST_DRAWING	-0.257466	-0.396284	
DAYS_FIRST_DUE	0.401838	0.323608	
DAYS_LAST_DUE_1ST_VERSION	0.423462	0.493174	

```
DAYS_LAST_DUE                   1.000000          0.927990
DAYS_TERMINATION                0.927990          1.000000
NFLAG_INSURED_ON_APPROVAL       0.012560         -0.003065

                            NFLAG_INSURED_ON_APPROVAL
SK_ID_PREV                                   0.003986
SK_ID_CURR                                   0.000876
AMT_ANNUITY                                  0.283080
AMT_APPLICATION                              0.259219
AMT_CREDIT                                   0.263932
AMT_DOWN_PAYMENT                            -0.042585
AMT_GOODS_PRICE                              0.243400
HOUR_APPR_PROCESS_START                     -0.117318
NFLAG_LAST_APPL_IN_DAY                      -0.007124
RATE_DOWN_PAYMENT                           -0.021633
RATE_INTEREST_PRIMARY                        0.311938
RATE_INTEREST_PRIVILEGED                    -0.067157
DAYS_DECISION                               -0.028905
SELLERPLACE_AREA                            -0.018280
CNT_PAYMENT                                  0.320520
DAYS_FIRST_DRAWING                           0.177652
DAYS_FIRST_DUE                              -0.119048
DAYS_LAST_DUE_1ST_VERSION                   -0.221947
DAYS_LAST_DUE                                0.012560
DAYS_TERMINATION                            -0.003065
NFLAG_INSURED_ON_APPROVAL                    1.000000

[21 rows x 21 columns]
================================================


================================================
Data Frame: Additional Information
------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   1670214 non-null  int64
 1   SK_ID_CURR                   1670214 non-null  int64
 2   NAME_CONTRACT_TYPE           1670214 non-null  object
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object
 16  NAME_CONTRACT_STATUS         1670214 non-null  object
 17  DAYS_DECISION                1670214 non-null  int64
 18  NAME_PAYMENT_TYPE            1670214 non-null  object
 19  CODE_REJECT_REASON           1670214 non-null  object
 20  NAME_TYPE_SUITE              849809 non-null   object
```

```
 21  NAME_CLIENT_TYPE             1670214 non-null  object
 22  NAME_GOODS_CATEGORY          1670214 non-null  object
 23  NAME_PORTFOLIO               1670214 non-null  object
 24  NAME_PRODUCT_TYPE            1670214 non-null  object
 25  CHANNEL_TYPE                 1670214 non-null  object
 26  SELLERPLACE_AREA             1670214 non-null  int64
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object
 30  PRODUCT_COMBINATION          1669868 non-null  object
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
None
================================================
```

Data Dictionary: installment_payments.csv

In [ ]:

```
# Entering information to call the EDA Method
eda_info_installments_payments = ['Installment Payments', df_installments_payments]

# Calling EDA Method
EDA(eda_info_installments_payments)
```

```
**********************************************

          DATAFRAME: Installment Payments

**********************************************


================================================
Data Frame: Size, Shape & Total Missing Values
------------------------------------------------
Number of Rows: 13605401
Number of Columns: 8
Number of Total Missing Values: 5810
Data Frame Shape: (13605401, 8)
================================================


================================================
Data Frame: Missing Values by Feature
------------------------------------------------
Number of Missing Values by Feature: SK_ID_PREV                   0
SK_ID_CURR                    0
NUM_INSTALMENT_VERSION        0
NUM_INSTALMENT_NUMBER         0
DAYS_INSTALMENT               0
DAYS_ENTRY_PAYMENT         2905
AMT_INSTALMENT                0
AMT_PAYMENT                2905
dtype: int64
================================================


================================================
Data Frame: Data Types
------------------------------------------------
SK_ID_PREV                   int64
SK_ID_CURR                   int64
NUM_INSTALMENT_VERSION     float64
NUM_INSTALMENT_NUMBER        int64
DAYS_INSTALMENT            float64
DAYS_ENTRY_PAYMENT         float64
AMT_INSTALMENT             float64
AMT_PAYMENT                float64
dtype: object
================================================


================================================
Data Frame: Data Types
------------------------------------------------
float64    5
int64      3
dtype: int64
================================================


================================================
Data Frame: Summary Statistics
------------------------------------------------
         SK_ID_PREV    SK_ID_CURR  NUM_INSTALMENT_VERSION  \
```

count	1.360540e+07	1.360540e+07	1.360540e+07
mean	1.903365e+06	2.784449e+05	8.566373e-01
std	5.362029e+05	1.027183e+05	1.035216e+00
min	1.000001e+06	1.000010e+05	0.000000e+00
25%	1.434191e+06	1.896390e+05	0.000000e+00
50%	1.896520e+06	2.786850e+05	1.000000e+00
75%	2.369094e+06	3.675300e+05	1.000000e+00
max	2.843499e+06	4.562550e+05	1.780000e+02

	NUM_INSTALMENT_NUMBER	DAYS_INSTALMENT	DAYS_ENTRY_PAYMENT
count	1.360540e+07	1.360540e+07	1.360250e+07
mean	1.887090e+01	-1.042270e+03	-1.051114e+03
std	2.666407e+01	8.009463e+02	8.005859e+02
min	1.000000e+00	-2.922000e+03	-4.921000e+03
25%	4.000000e+00	-1.654000e+03	-1.662000e+03
50%	8.000000e+00	-8.180000e+02	-8.270000e+02
75%	1.900000e+01	-3.610000e+02	-3.700000e+02
max	2.770000e+02	-1.000000e+00	-1.000000e+00

	AMT_INSTALMENT	AMT_PAYMENT
count	1.360540e+07	1.360250e+07
mean	1.705091e+04	1.723822e+04
std	5.057025e+04	5.473578e+04
min	0.000000e+00	0.000000e+00
25%	4.226085e+03	3.398265e+03
50%	8.884080e+03	8.125515e+03
75%	1.671021e+04	1.610842e+04
max	3.771488e+06	3.771488e+06

==========

==========
Data Frame: Correlation Statistics

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION
SK_ID_PREV	1.000000	0.002132	0.000685
SK_ID_CURR	0.002132	1.000000	0.000480
NUM_INSTALMENT_VERSION	0.000685	0.000480	1.000000
NUM_INSTALMENT_NUMBER	-0.002095	-0.000548	-0.323414
DAYS_INSTALMENT	0.003748	0.001191	0.130244
DAYS_ENTRY_PAYMENT	0.003734	0.001215	0.128124
AMT_INSTALMENT	0.002042	-0.000226	0.168109
AMT_PAYMENT	0.001887	-0.000124	0.177176

	NUM_INSTALMENT_NUMBER	DAYS_INSTALMENT
SK_ID_PREV	-0.002095	0.003748
SK_ID_CURR	-0.000548	0.001191
NUM_INSTALMENT_VERSION	-0.323414	0.130244
NUM_INSTALMENT_NUMBER	1.000000	0.090286
DAYS_INSTALMENT	0.090286	1.000000
DAYS_ENTRY_PAYMENT	0.094305	0.999491
AMT_INSTALMENT	-0.089640	0.125985
AMT_PAYMENT	-0.087664	0.127018

	DAYS_ENTRY_PAYMENT	AMT_INSTALMENT	AMT_PAYMENT
SK_ID_PREV	0.003734	0.002042	0.001887
SK_ID_CURR	0.001215	-0.000226	-0.000124
NUM_INSTALMENT_VERSION	0.128124	0.168109	0.177176
NUM_INSTALMENT_NUMBER	0.094305	-0.089640	-0.087664
DAYS_INSTALMENT	0.999491	0.125985	0.127018

```
DAYS_ENTRY_PAYMENT                1.000000        0.125555     0.126602
AMT_INSTALMENT                    0.125555        1.000000     0.937191
AMT_PAYMENT                       0.126602        0.937191     1.000000
================================================


================================================
Data Frame: Additional Information
------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13605401 entries, 0 to 13605400
Data columns (total 8 columns):
 #   Column                  Dtype
---  ------                  -----
 0   SK_ID_PREV              int64
 1   SK_ID_CURR              int64
 2   NUM_INSTALMENT_VERSION  float64
 3   NUM_INSTALMENT_NUMBER   int64
 4   DAYS_INSTALMENT         float64
 5   DAYS_ENTRY_PAYMENT      float64
 6   AMT_INSTALMENT          float64
 7   AMT_PAYMENT             float64
dtypes: float64(5), int64(3)
memory usage: 830.4 MB
None
================================================
```

Visual Exploritory Data Analysis (VEDA)

VEDA: Input & Target Features

In [ ]:
```
# Import Libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

VEDA: Target Feature Visualiaztion

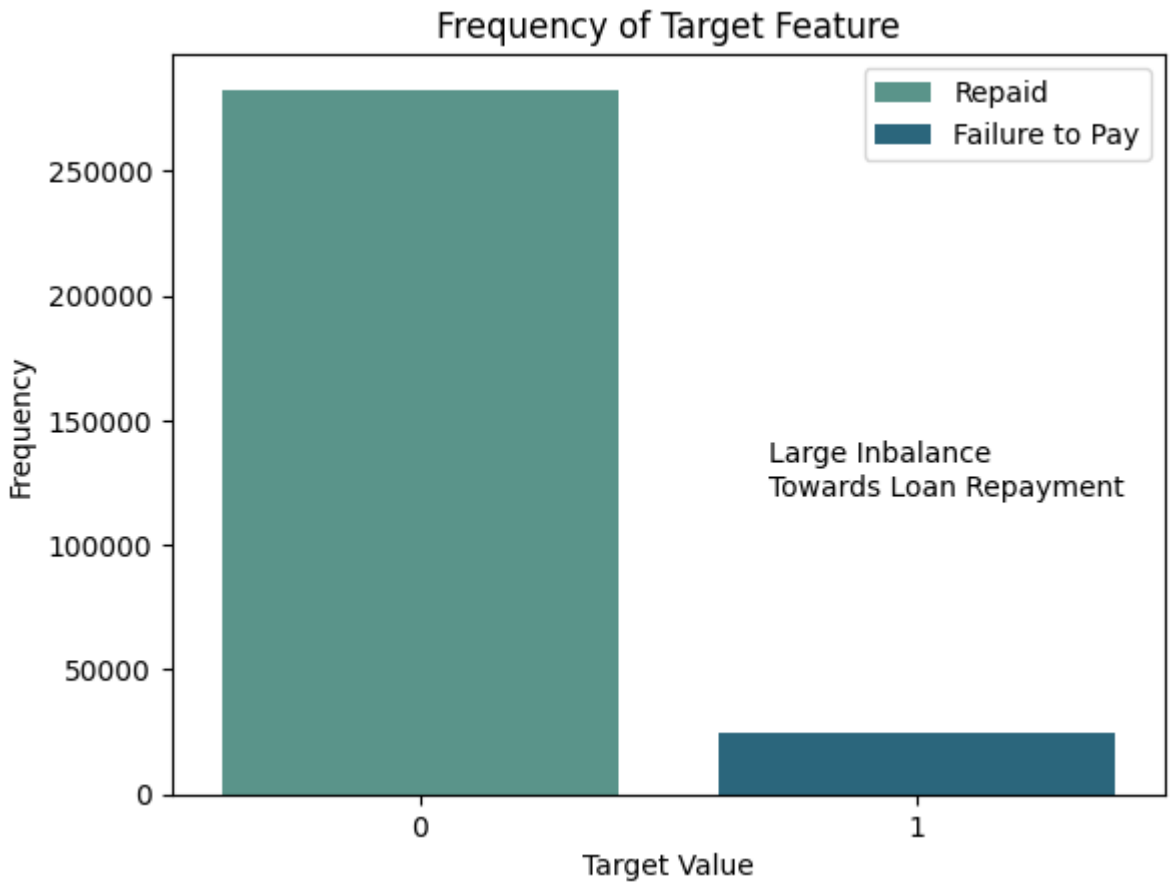
In [ ]:
```
# First lets see numerically the distribution of targets
df_app_train["TARGET"].value_counts()
```

Out[ ]:
```
0    282686
1     24825
Name: TARGET, dtype: int64
```

We can see that there is a large imbalance between the targets, with most customers lent to repaying the loan.

In [ ]:
```
# Lets visualize this

# Bar Plot
g = sns.countplot(data = df_app_train, x = "TARGET", palette="crest", hue="TARGET", d
g.legend(loc="upper right", labels=["Repaid", "Failure to Pay"])
g.set_title("Frequency of Target Feature")
g.set_ylabel("Frequency")
g.set_xlabel("Target Value")
g.annotate("Large Inbalance \nTowards Loan Repayment", xy = (0.7, 120000))
```

Out[ ]:
```
Text(0.7, 120000, 'Large Inbalance \nTowards Loan Repayment')
```



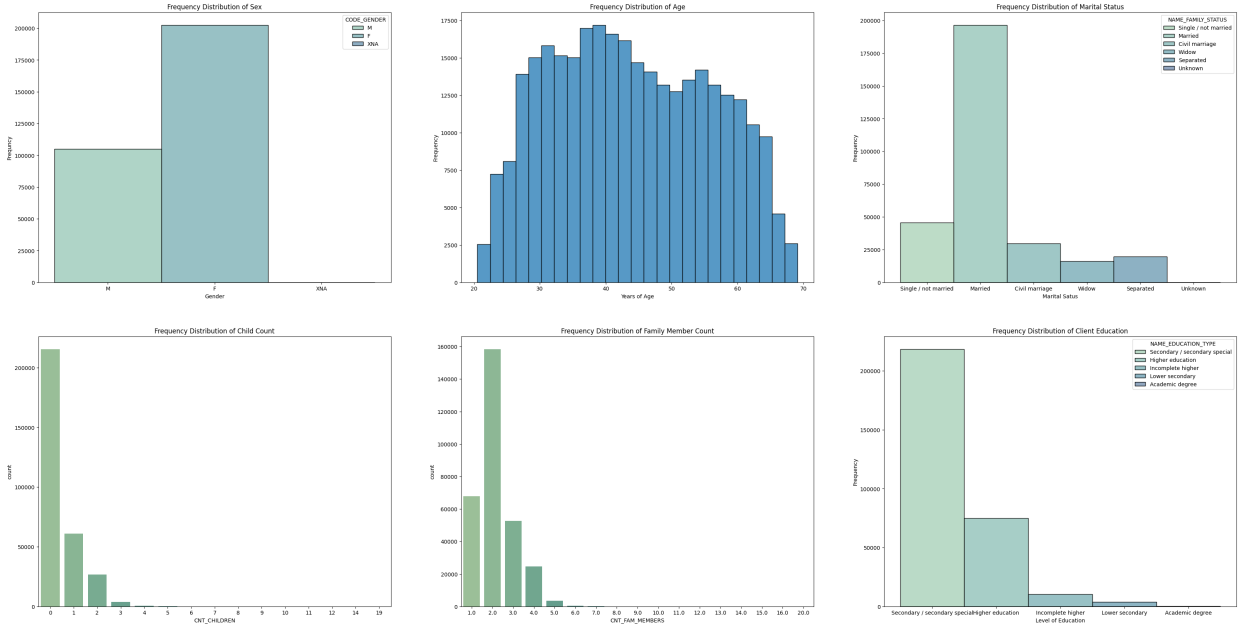
VEDA: Input Feature Visualiaztion (application_train.csv)

VEDA: Input Feature Visualiaztion: Demographics

In [ ]:
```
#pre-vis processing
df_app_train_age = df_app_train['DAYS_BIRTH'] / 365 * -1

# set up fig
fig, ax = plt.subplots(2,3, sharex=False, figsize=(40,20))

# Set Figure Labels
ax[0,0].set_title('Frequency Distribution of Sex')
ax[0,1].set_title('Frequency Distribution of Age')
```

```
ax[0,2].set_title('Frequency Distribution of Marital Status')
ax[1,0].set_title('Frequency Distribution of Child Count')
ax[1,1].set_title('Frequency Distribution of Family Member Count')
ax[1,2].set_title('Frequency Distribution of Client Education')

# Set Lables
ax[0,0].set_ylabel('Frequncy')
ax[0,1].set_ylabel('Frequency')
ax[0,2].set_ylabel('Frequency')
ax[1,0].set_ylabel('Frequency')
ax[1,1].set_ylabel('Frequency')
ax[1,2].set_ylabel('Frequency')

# Set Lables
ax[0,0].set_xlabel('Gender')
ax[0,1].set_xlabel('Years of Age')
ax[0,2].set_xlabel('Marital Satus')
ax[1,0].set_xlabel('Number of Children')
ax[1,1].set_xlabel('Number of Family Members')
ax[1,2].set_xlabel('Level of Education')

# Set histogram
sns.histplot(ax = ax[0,0], data = df_app_train, palette="crest", x = "CODE_GENDER", h
sns.histplot(ax = ax[0,1], data = df_app_train_age, bins=25)
sns.histplot(ax = ax[0,2], data = df_app_train, palette="crest", x = "NAME_FAMILY_STA
sns.countplot(ax = ax[1,0], data = df_app_train, palette="crest", x = "CNT_CHILDREN")
sns.countplot(ax = ax[1,1], data = df_app_train, palette="crest", x = "CNT_FAM_MEMBER
sns.histplot(ax = ax[1,2], data = df_app_train, palette="crest", x = "NAME_EDUCATION_
```

Out[ ]:
```
<Axes: title={'center': 'Frequency Distribution of Client Education'}, xlabel='Level
of Education', ylabel='Frequency'>
```



These Demographic distributions only give us a sense of the body of clients. We see that they are most commonly:

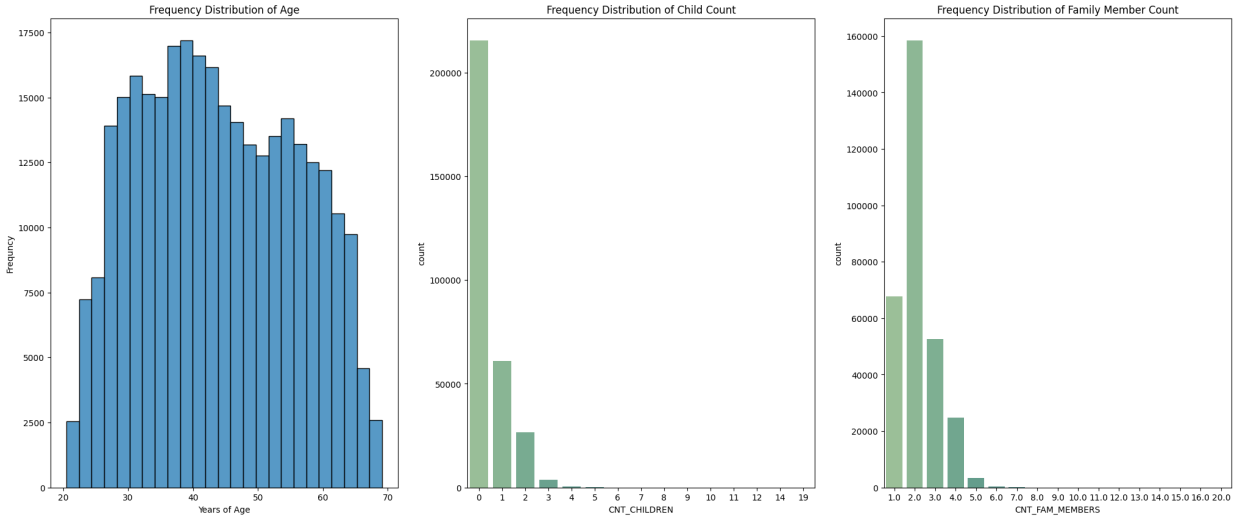
- Married
- Females
- From the ages of 30 to 60
- No Children

- And 2 Family Members

Lets apply the target variable to the distributions to see if there are any obvious trends that we can look further into. Lets first look into the numerical features.

In [ ]:
```
# set up fig
fig, ax = plt.subplots(1,3, sharex=False, figsize=(25,10))

# Set Figure Labels
ax[0].set_title('Frequency Distribution of Age')
ax[1].set_title('Frequency Distribution of Child Count')
ax[2].set_title('Frequency Distribution of Family Member Count')

# Set Lables
ax[0].set_ylabel('Frequncy')
ax[1].set_ylabel('Frequency')
ax[2].set_ylabel('Frequency')

# Set Lables
ax[0].set_xlabel('Years of Age')
ax[1].set_xlabel('Number of Children')
ax[2].set_xlabel('Number of Family Members')

# Set histogram
sns.histplot(ax = ax[0], data = df_app_train_age, bins=25)
sns.countplot(ax = ax[1], data = df_app_train, palette="crest", x = "CNT_CHILDREN")
sns.countplot(ax = ax[2], data = df_app_train, palette="crest", x = "CNT_FAM_MEMBERS"
```

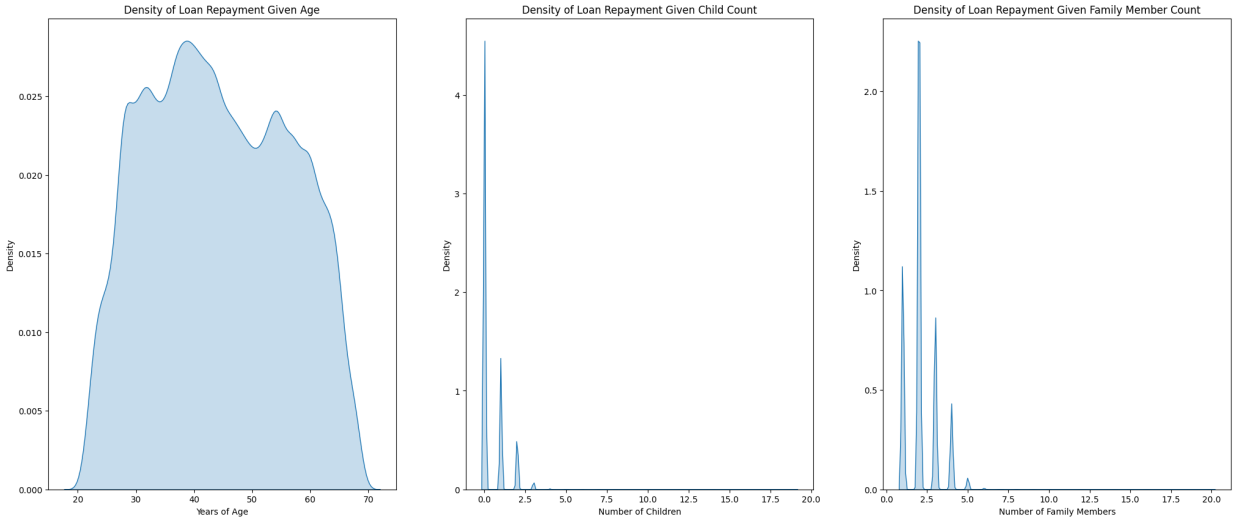
Out[ ]:
```
<Axes: title={'center': 'Frequency Distribution of Family Member Count'}, xlabel='CNT
_FAM_MEMBERS', ylabel='count'>
```



In [ ]:
```
# set up fig
fig, ax = plt.subplots(1,3, sharex=False, figsize=(25,10))

# Set Figure Labels
ax[0].set_title('Density of Loan Repayment Given Age')
ax[1].set_title('Density of Loan Repayment Given Child Count')
ax[2].set_title('Density of Loan Repayment Given Family Member Count')

# Set Lables
ax[0].set_ylabel('Density')
ax[1].set_ylabel('Density')
```

```
ax[2].set_ylabel('Density')

# Set Lables
ax[0].set_xlabel('Years of Age')
ax[1].set_xlabel('Number of Children')
ax[2].set_xlabel('Number of Family Members')

# Set KDE
sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'DAYS_BIRTH'] / 365 * -1, l
sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'CNT_CHILDREN'], label = 't
sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'CNT_FAM_MEMBERS'], label =
```

Out[ ]:
```
<Axes: title={'center': 'Density of Loan Repayment Given Family Member Count'}, xlabe
l='Number of Family Members', ylabel='Density'>
```



DISCUSSION </br> These visualizations give us some great insight into the general trends of where loan repayment is most common in these data. We can see that repayment is most common in individuals around 40 years old, with no children, and a family size around 2.

IMPORTANT </br> Since we are using a KDE or Kernal Density Estimate, this just shows where the highest amount of occurences happen, not who is most likely to do so. This instead will give us insight on where we might be able to reduce features to understand where people are not repaying their loans.

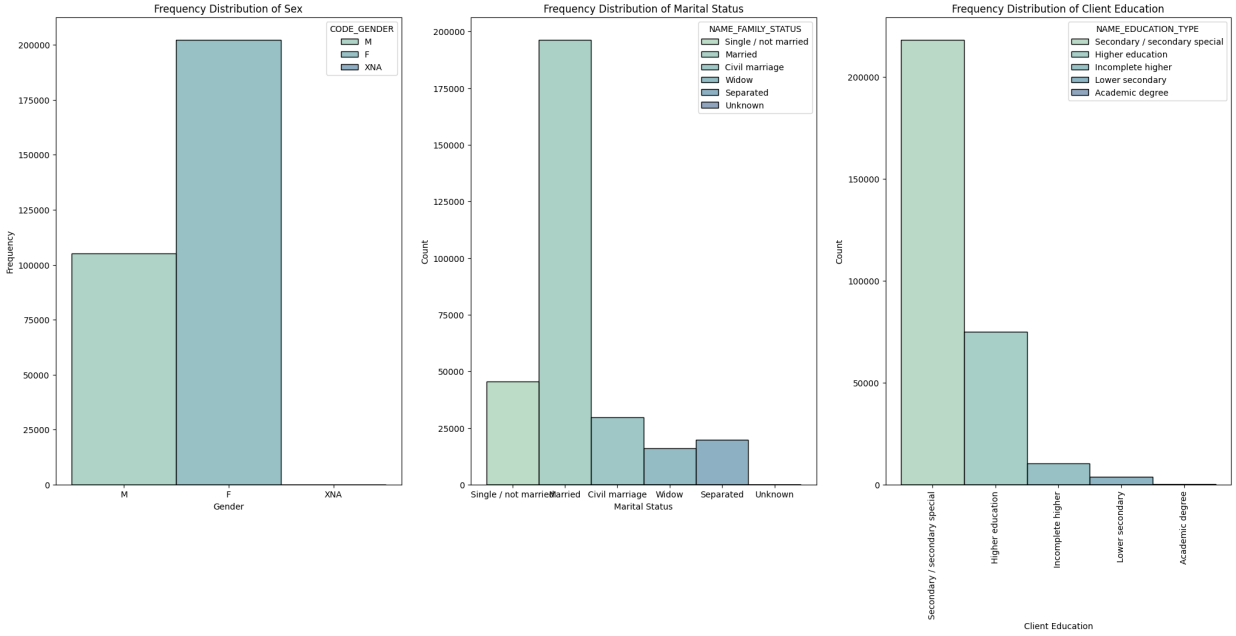
Lets now Take a look at the categorical side

In [ ]:
```
# set up fig
fig, ax = plt.subplots(1,3, sharex=False, figsize=(25,10))

# Set Figure Labels
ax[0].set_title('Frequency Distribution of Sex')
ax[1].set_title('Frequency Distribution of Marital Status')
ax[2].set_title('Frequency Distribution of Client Education')

# Set Lables
ax[0].set_ylabel('Frequncy')
ax[0].set_ylabel('Frequency')
ax[0].set_ylabel('Frequency')

# Set Lables
ax[0].set_xlabel('Gender')
```

```
ax[1].set_xlabel('Marital Status')
ax[2].set_xlabel('Client Education')

# Set histogram
sns.histplot(ax = ax[0], data = df_app_train, palette="crest", x = "CODE_GENDER", hue
sns.histplot(ax = ax[1], data = df_app_train, palette="crest", x = "NAME_FAMILY_STATU
sns.histplot(ax = ax[2], data = df_app_train, palette="crest", x = "NAME_EDUCATION_TY
plt.xticks(rotation=90)
```

Out[ ]:

```
([0, 1, 2, 3, 4],
 [Text(0, 0, 'Secondary / secondary special'),
  Text(1, 0, 'Higher education'),
  Text(2, 0, 'Incomplete higher'),
  Text(3, 0, 'Lower secondary'),
  Text(4, 0, 'Academic degree')])
```

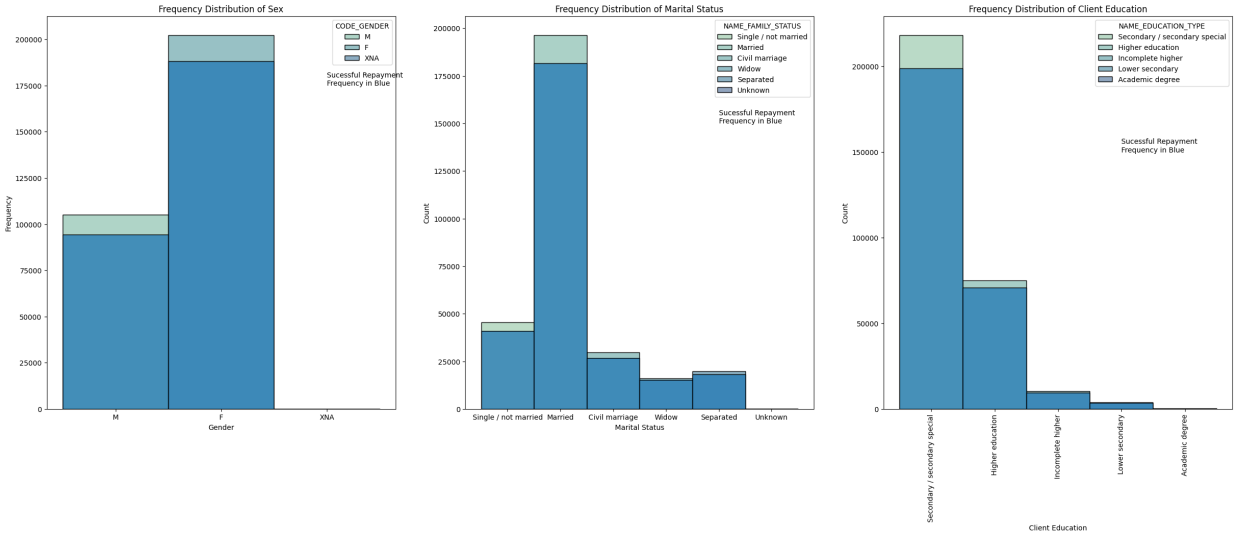


In [ ]:

```
# set up fig
fig, ax = plt.subplots(1,3, sharex=False, figsize=(30,10))

# Set Figure Labels
ax[0].set_title('Frequency Distribution of Sex')
ax[1].set_title('Frequency Distribution of Marital Status')
ax[2].set_title('Frequency Distribution of Client Education')

# Set Lables
ax[0].set_ylabel('Frequncy')
ax[0].set_ylabel('Frequency')
ax[0].set_ylabel('Frequency')

# Set Lables
ax[0].set_xlabel('Gender')
ax[1].set_xlabel('Marital Status')
ax[2].set_xlabel('Client Education')

sns.histplot(ax = ax[0], data = df_app_train, palette="crest", x = "CODE_GENDER", hue
sns.histplot(ax = ax[1], data = df_app_train, palette="crest", x = "NAME_FAMILY_STATU
sns.histplot(ax = ax[2], data = df_app_train, palette="crest", x = "NAME_EDUCATION_TY
ax1 = sns.histplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'CODE_GENDER'], labe
ax2 = sns.histplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'NAME_FAMILY_STATUS'
ax3 = sns.histplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'NAME_EDUCATION_TYPE
```

```
plt.xticks(rotation=90)

ax1.annotate("Sucessful Repayment\nFrequency in Blue", xy=('XNA', 175000))
ax2.annotate("Sucessful Repayment\nFrequency in Blue", xy=('Separated', 150000))
ax3.annotate("Sucessful Repayment\nFrequency in Blue", xy=('Lower secondary', 150000)
```

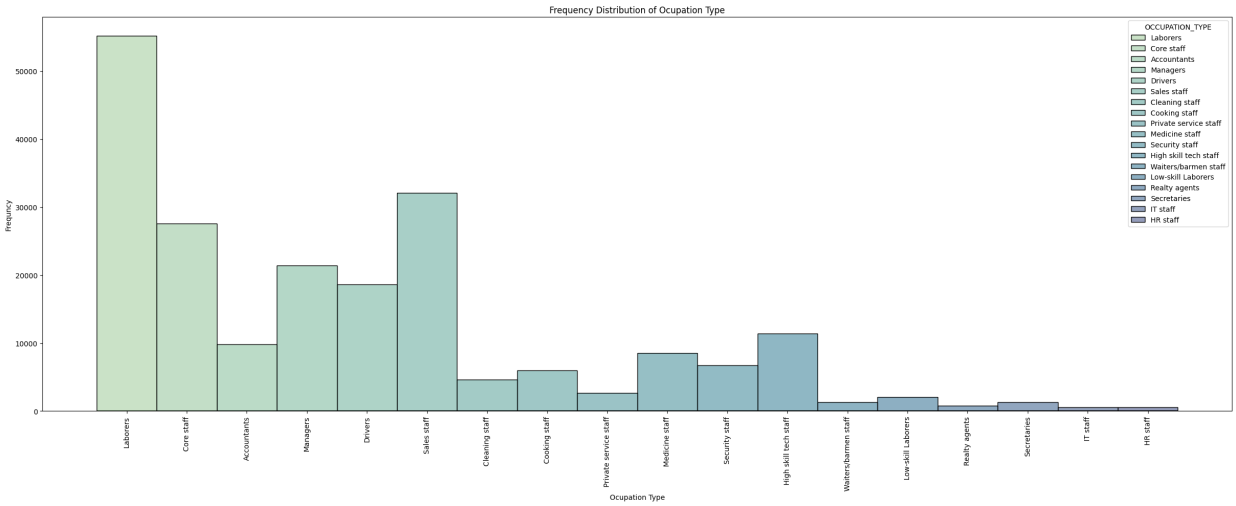
Out[ ]:
```
Text(Lower secondary, 150000, 'Sucessful Repayment\nFrequency in Blue')
```



VEDA: Input Feature Visualiaztion: Occupation

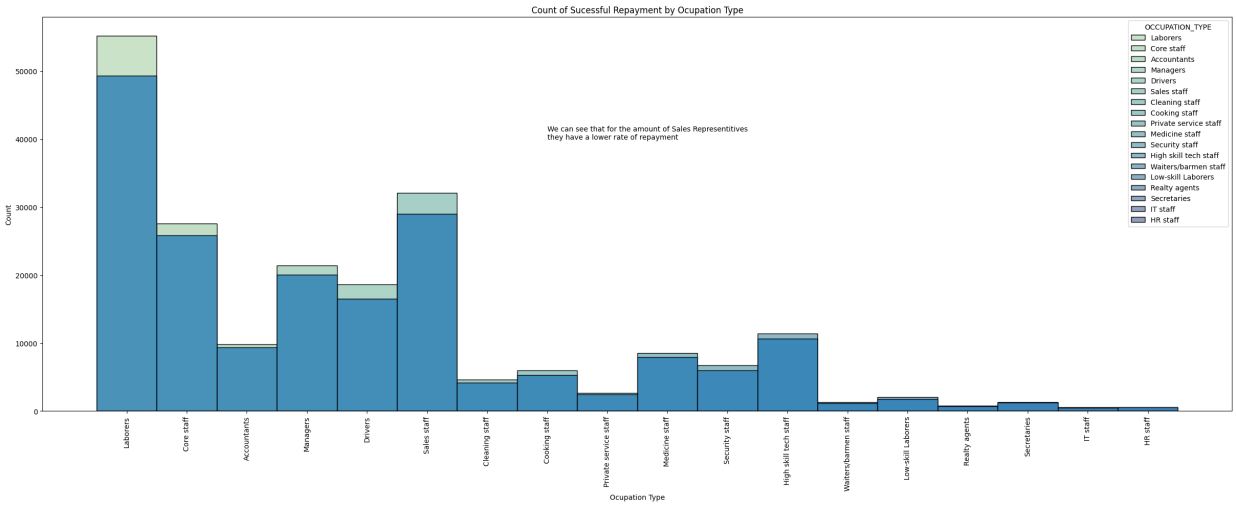
In [ ]:
```
# set up fig
fig, ax = plt.subplots(1,1, sharex=False, figsize=(30,10))

# Set Figure Labels
ax.set_title('Frequency Distribution of Ocupation Type')

# Set Lables
ax.set_ylabel('Frequncy')

# Set Lables
ax.set_xlabel('Ocupation Type')

sns.histplot(ax = ax, data = df_app_train, palette="crest", x = "OCCUPATION_TYPE", hu
plt.xticks(rotation=90)
```

Out[ ]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17],
 [Text(0, 0, 'Laborers'),
  Text(1, 0, 'Core staff'),
  Text(2, 0, 'Accountants'),
  Text(3, 0, 'Managers'),
  Text(4, 0, 'Drivers'),
  Text(5, 0, 'Sales staff'),
  Text(6, 0, 'Cleaning staff'),
  Text(7, 0, 'Cooking staff'),
  Text(8, 0, 'Private service staff'),
  Text(9, 0, 'Medicine staff'),
  Text(10, 0, 'Security staff'),
  Text(11, 0, 'High skill tech staff'),
  Text(12, 0, 'Waiters/barmen staff'),
  Text(13, 0, 'Low-skill Laborers'),
  Text(14, 0, 'Realty agents'),
  Text(15, 0, 'Secretaries'),
  Text(16, 0, 'IT staff'),
  Text(17, 0, 'HR staff')])
```



In [ ]:
```
# set up fig
fig, ax = plt.subplots(1,1, sharex=False, figsize=(30,10))

# Set Figure Labels
ax.set_title('Count of Sucessful Repayment by Ocupation Type')

# Set Lables
ax.set_ylabel('Count')

# Set Lables
ax.set_xlabel('Ocupation Type')

ax = sns.histplot(ax = ax, data = df_app_train, palette="crest", x = "OCCUPATION_TYPE
ax = sns.histplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'OCCUPATION_TYPE'], l
ax.annotate("We can see that for the amount of Sales Representitives \nthey have a lo
plt.xticks(rotation=90)
```

Out[ ]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17],
 [Text(0, 0, 'Laborers'),
  Text(1, 0, 'Core staff'),
  Text(2, 0, 'Accountants'),
  Text(3, 0, 'Managers'),
  Text(4, 0, 'Drivers'),
  Text(5, 0, 'Sales staff'),
  Text(6, 0, 'Cleaning staff'),
  Text(7, 0, 'Cooking staff'),
  Text(8, 0, 'Private service staff'),
  Text(9, 0, 'Medicine staff'),
  Text(10, 0, 'Security staff'),
  Text(11, 0, 'High skill tech staff'),
  Text(12, 0, 'Waiters/barmen staff'),
  Text(13, 0, 'Low-skill Laborers'),
  Text(14, 0, 'Realty agents'),
  Text(15, 0, 'Secretaries'),
  Text(16, 0, 'IT staff'),
  Text(17, 0, 'HR staff')])
```

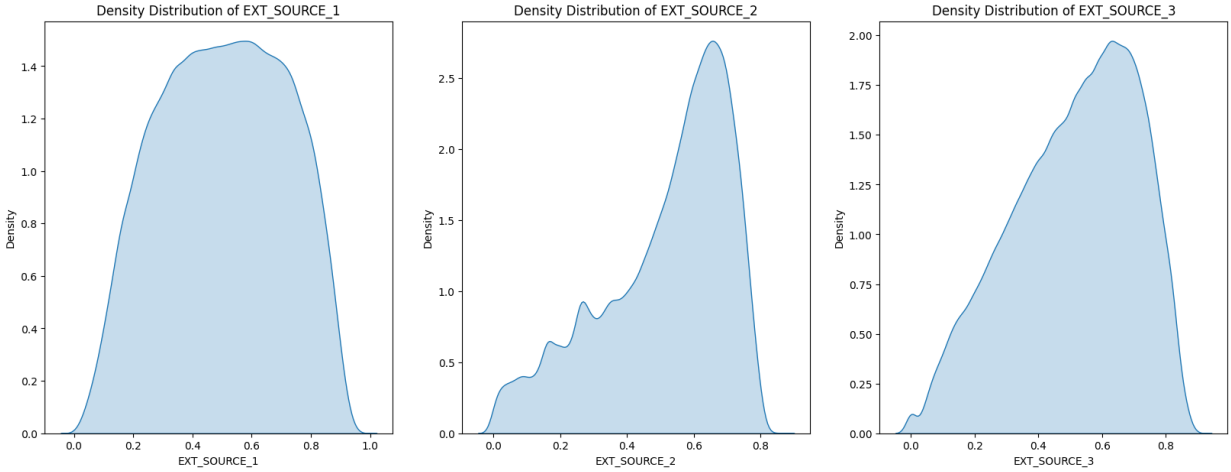


VEDA: Input Feature Visualiaztion: EXTERNAL SOURCE

In [ ]:
```
# set up fig
fig, ax = plt.subplots(1,3, sharex=False, figsize=(20,7))

# Set Figure Labels
ax[0].set_title('Density Distribution of EXT_SOURCE_1')
ax[1].set_title('Density Distribution of EXT_SOURCE_2')
ax[2].set_title('Density Distribution of EXT_SOURCE_3')

# Set Lables
ax[0].set_ylabel('Density')
ax[1].set_ylabel('Density')
ax[2].set_ylabel('Density')

# Set Lables
ax[0].set_xlabel('EXT_SOURCE_1')
ax[1].set_xlabel('EXT_SOURCE_2')
ax[2].set_xlabel('EXT_SOURCE_3')

# Set histogram
sns.kdeplot(df_app_train["EXT_SOURCE_1"], ax=ax[0],fill=True)
```

```
sns.kdeplot(df_app_train["EXT_SOURCE_2"], ax=ax[1],fill=True)
sns.kdeplot(df_app_train["EXT_SOURCE_3"], ax=ax[2],fill=True)
```

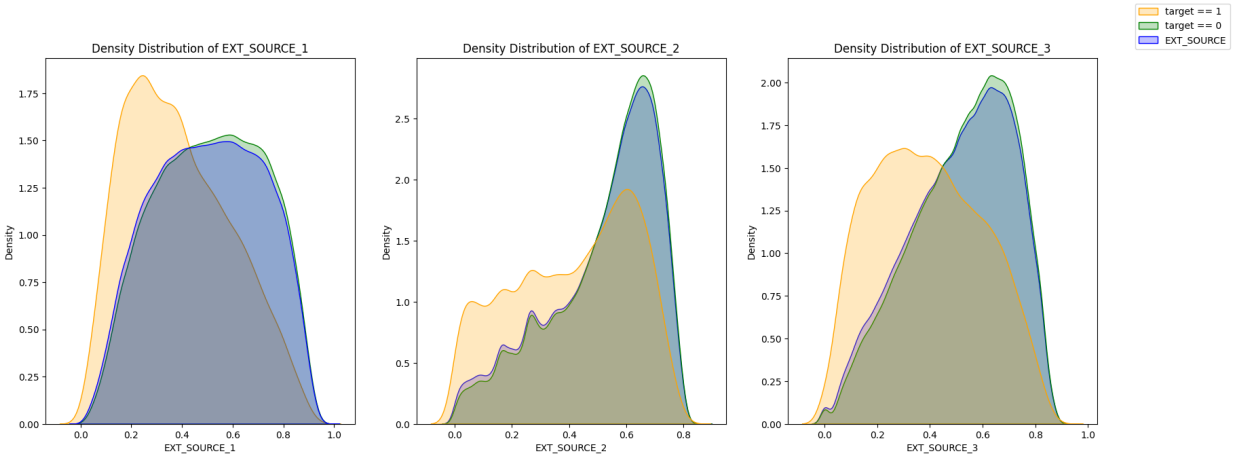
Out[ ]:
```
<Axes: title={'center': 'Density Distribution of EXT_SOURCE_3'}, xlabel='EXT_SOURCE_
3', ylabel='Density'>
```



Lets now observe the target density over these external data sources to see if there may be any interesting distributions or trends.

In [ ]:
```
# set up fig
fig, ax = plt.subplots(1,3, sharex=False, figsize=(20,7))

# Set Figure Labels
ax[0].set_title('Density Distribution of EXT_SOURCE_1')
ax[1].set_title('Density Distribution of EXT_SOURCE_2')
ax[2].set_title('Density Distribution of EXT_SOURCE_3')

# Set Lables
ax[0].set_ylabel('Density')
ax[1].set_ylabel('Density')
ax[2].set_ylabel('Density')

# Set Lables
ax[0].set_xlabel('EXT_SOURCE_1')
ax[1].set_xlabel('EXT_SOURCE_2')
ax[2].set_xlabel('EXT_SOURCE_3')

# Set kdeplot of targets

sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 1, 'EXT_SOURCE_1'], label = 't
sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'EXT_SOURCE_1'], label = 't
sns.kdeplot(df_app_train["EXT_SOURCE_1"],label="EXT_SOURCE", ax=ax[0], fill=True, col
fig.legend()

sns.kdeplot(df_app_train["EXT_SOURCE_2"],label="EXT_SOURCE_2", ax=ax[1], fill=True, c
sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'EXT_SOURCE_2'], label = 't
sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 1, 'EXT_SOURCE_2'], label = 't

sns.kdeplot(df_app_train["EXT_SOURCE_3"],label="EXT_SOURCE_3", ax=ax[2], fill=True, c
sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 0, 'EXT_SOURCE_3'], label = 't
sns.kdeplot(df_app_train.loc[df_app_train['TARGET'] == 1, 'EXT_SOURCE_3'], label = 't
```

Out[ ]:
```
<Axes: title={'center': 'Density Distribution of EXT_SOURCE_3'}, xlabel='EXT_SOURCE_
3', ylabel='Density'>
```



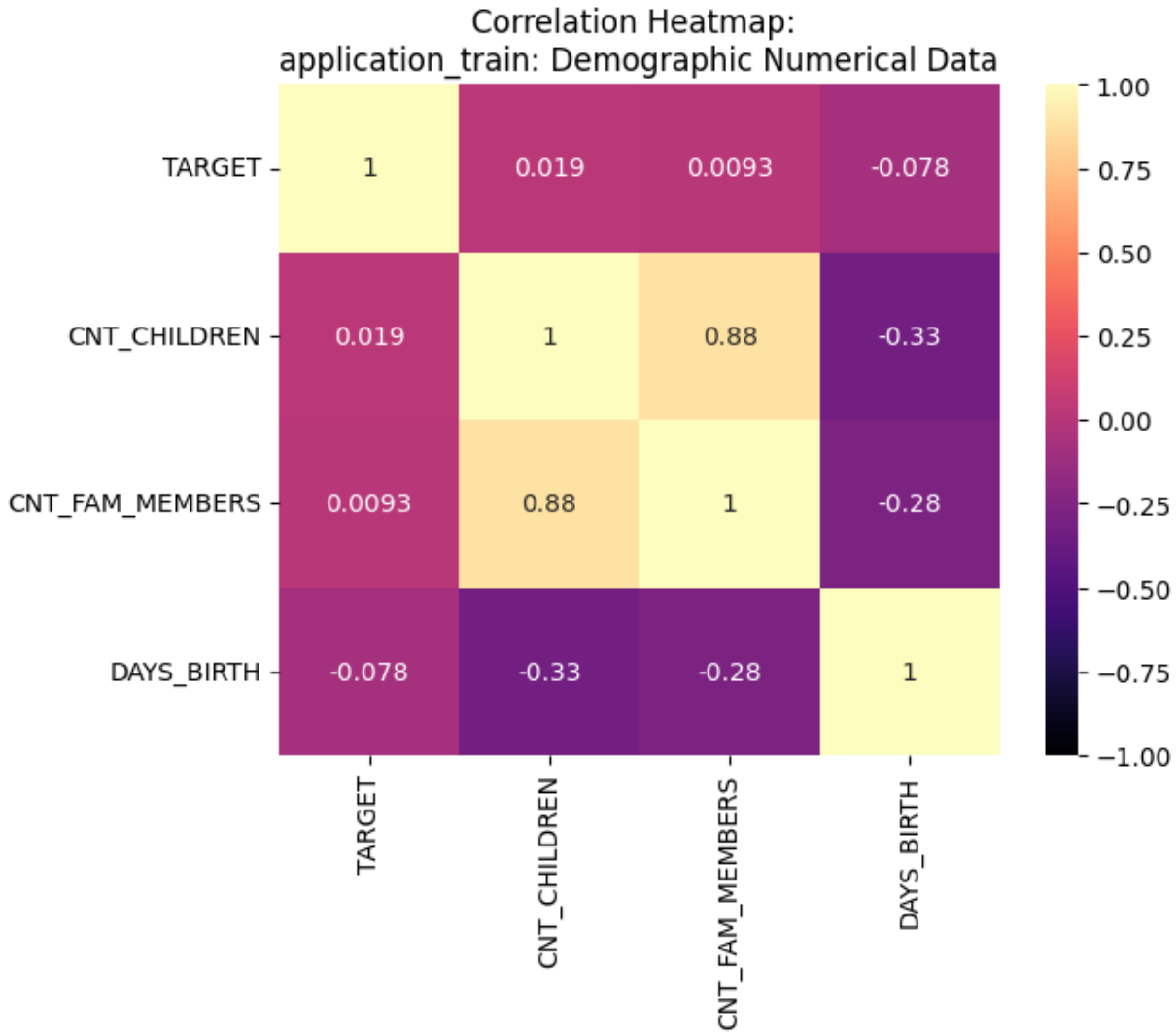
DISCUSION </br> We can see that in each of teh EXT_SOURCE visualizations that EXT_SOURCE seems to follow the same density as target 0. However, the largest seperation between these two features and the target value of 1 is shown in EXT_SOURCE_3. Although slight, this may give some insight into how these data could be use later.

VEDA: Correlation Visualiaztion: Demographic Numerical

In [ ]:
```
# Lets see the correlation map for the numerical demographics
corr_occ_data = df_app_train[["TARGET", "CNT_CHILDREN", "CNT_FAM_MEMBERS", "DAYS_BIRT
corr_occ_data["DAYS_BIRTH"] = abs(corr_occ_data["DAYS_BIRTH"])
corr_occ_data = corr_occ_data.corr()
sns.heatmap(corr_occ_data, cmap="magma", annot=True, vmin= -1.0, vmax=1.0)
plt.title("Correlation Heatmap: \napplication_train: Demographic Numerical Data")
```

```
<ipython-input-285-f80d2e77f988>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  corr_occ_data["DAYS_BIRTH"] = abs(corr_occ_data["DAYS_BIRTH"])
```

Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap: \napplication_train: Demographic Numerical Dat
a')
```

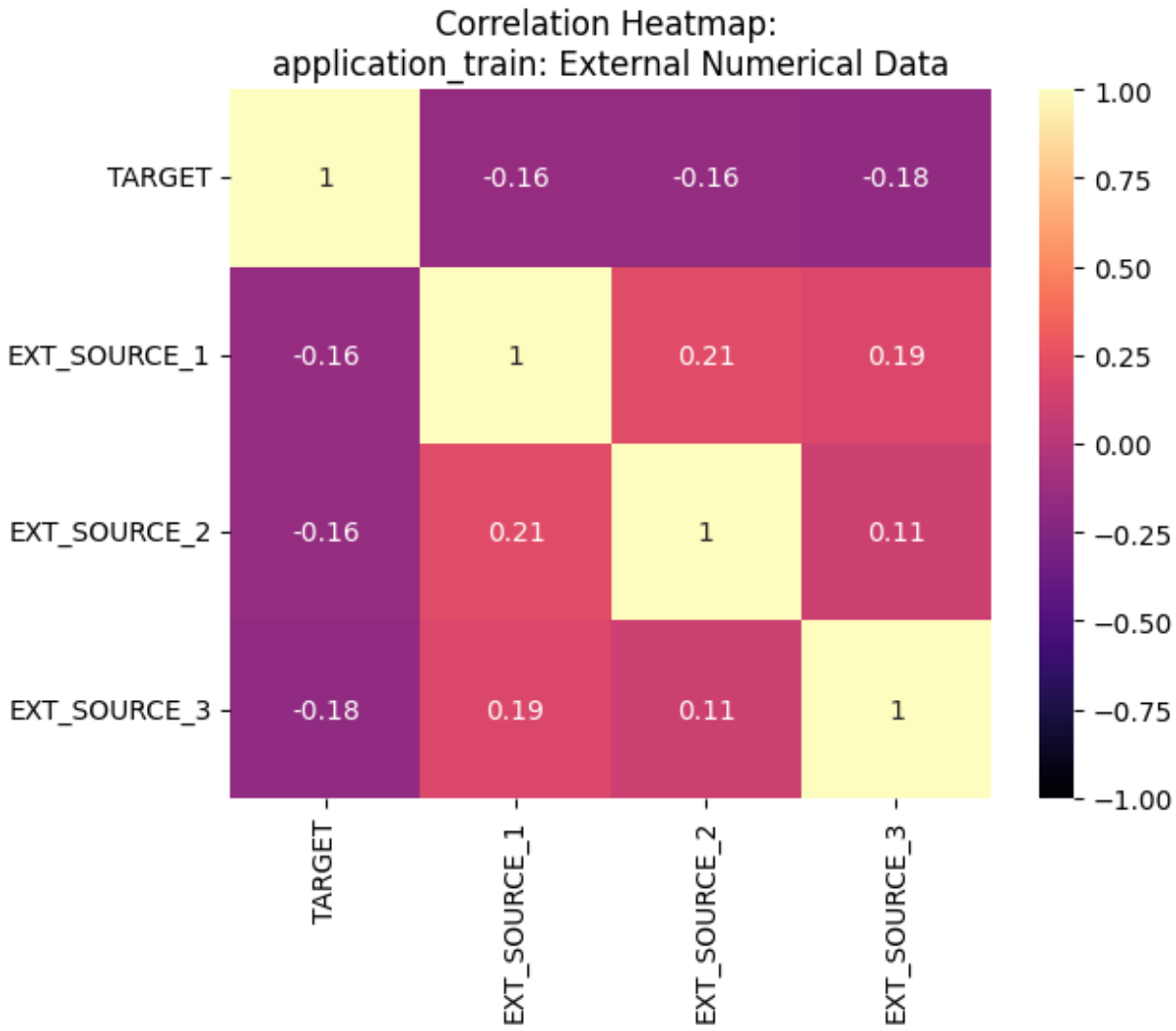


DISCUSSION </br> Unfortunatly there are not many insights we can pull from these data correlation coefficients. The largest correlation coefficent to show is DAYS_BIRTH. This negitive correlation shows that the older the client the more likely they are to succesfully repay since a positive correlation would be increasing with target == 1, failure to repay.

VEDA: Correlation Visualiaztion: EXTERNAL Numerical

In [ ]:
```
# Lets see the correlation map for the numerical external data
corr_extern_data = df_app_train[["TARGET", "EXT_SOURCE_1", "EXT_SOURCE_2", "EXT_SOURC
corr_extern_data = corr_extern_data.corr()
sns.heatmap(corr_extern_data, cmap="magma", annot=True, vmin= -1.0, vmax=1.0)
plt.title("Correlation Heatmap: \napplication_train: External Numerical Data")
```

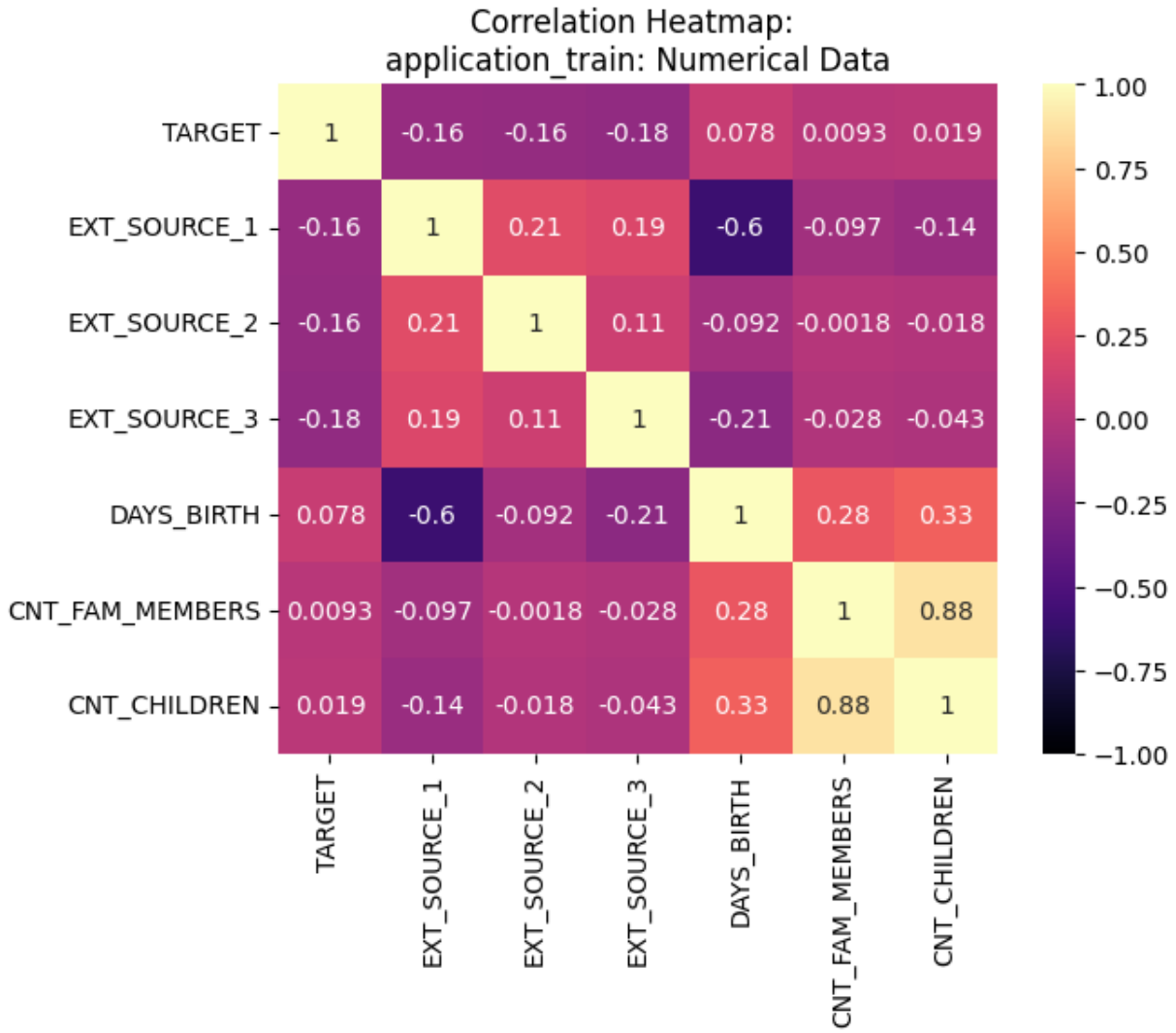
Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap: \napplication_train: External Numerical Data')
```



DISCUSSION </br> This heatmap provides some insight into the correlations of these data to the target value. The most evident insight about this visualization is that EXT_SOURCE_3 has the largest negitive correlation to the TARGET. This means that this feature, has a positive correlation to repayment of the loan, since it has a negitive coefficient and a positive coefficient would be positively correlated to target == 1 which is failure to pay.

In [ ]:
```
# Lets put these 2 heatmaps together for a better summary
corr_full_data = df_app_train[["TARGET", "EXT_SOURCE_1", "EXT_SOURCE_2", "EXT_SOURCE_
corr_full_data = corr_full_data.corr()
sns.heatmap(corr_full_data, cmap="magma", annot=True, vmin= -1.0, vmax=1.0)
plt.title("Correlation Heatmap: \napplication_train: Numerical Data")
```

Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap: \napplication_train: Numerical Data')
```



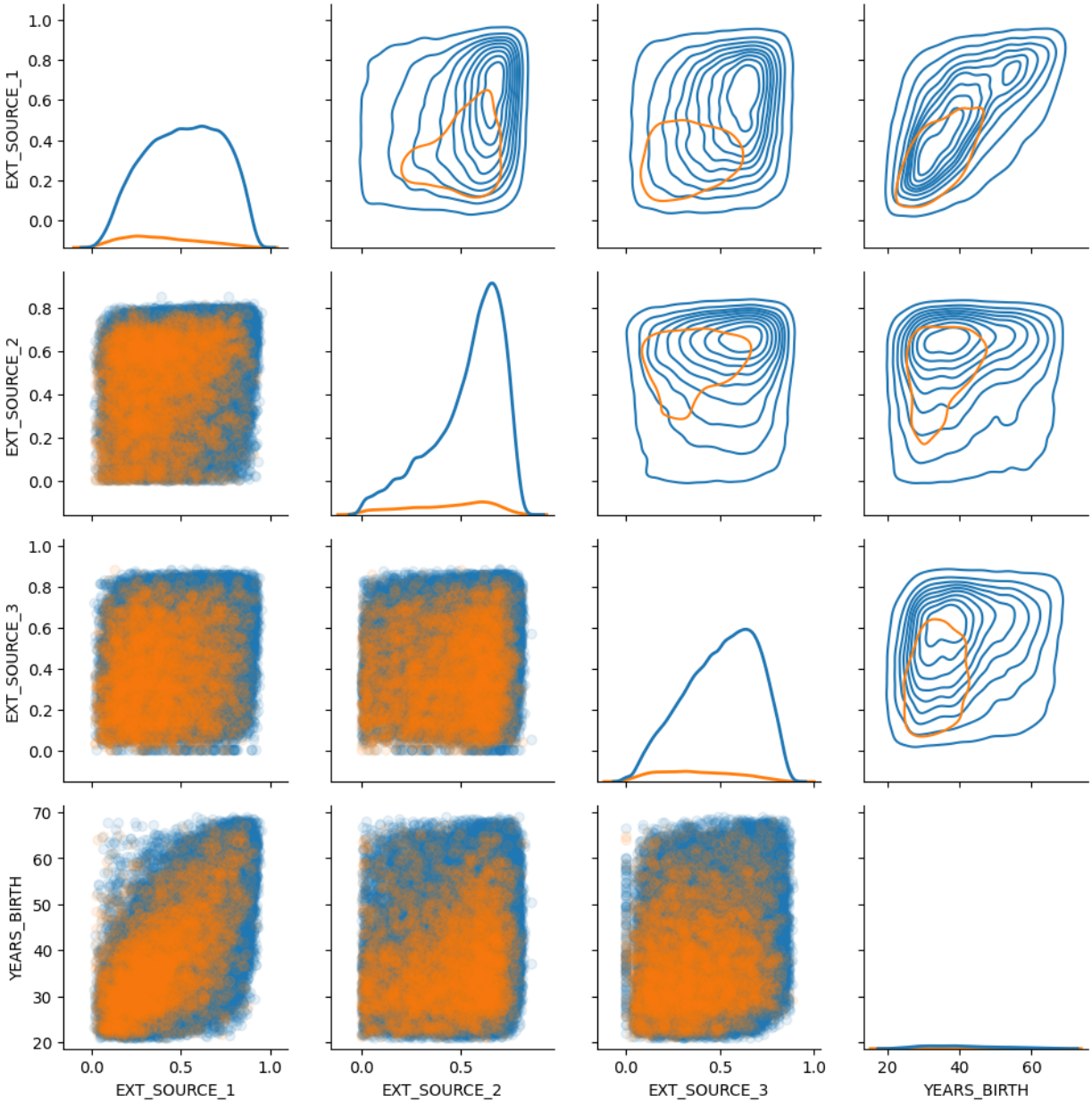
VEDA: Pair-based Input & Output

In [ ]:

```
# Lets create a pair plot of the numerical data used in some of these heat maps
df_pair_plot = df_app_train[["TARGET", "EXT_SOURCE_1", "EXT_SOURCE_2", "EXT_SOURCE_3"
df_pair_plot["YEARS_BIRTH"] = abs(df_app_train["DAYS_BIRTH"] / 365)

# Data size is too large for pair plot so we will reduce
df_pair_plot = df_pair_plot.dropna()
df_pair_plot = df_pair_plot.iloc[:50000]

g = sns.PairGrid(df_pair_plot, hue = "TARGET")
g.map_upper(sns.kdeplot)
g.map_diag(sns.kdeplot, lw=2)
g.map_lower(plt.scatter, alpha = 0.1)
g.legend()
```

```
<ipython-input-288-57ac1db8742d>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df_pair_plot["YEARS_BIRTH"] = abs(df_app_train["DAYS_BIRTH"] / 365)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-288-57ac1db8742d> in <cell line: 13>()
     11 g.map_diag(sns.kdeplot, lw=2)
     12 g.map_lower(plt.scatter, alpha = 0.1)
---> 13 g.legend()

TypeError: 'NoneType' object is not callable
```



DISCUSSION </br> The most interesting artifact of this visualization is the positive correlation that is seen at the intersection of YEARS_BIRTH and EXT_SOURCE_1. On both the scatter plot and the KDE plot, we are able to see that there a positive correlation between these 2 features. This is interesting becuase looking back at the KDE Density Distribution there is some seperation between target = 1 and EXT_SOURCE_1. This seems have the potential to yeilduseful insight in feature analysis and selection

---

VEDA: Missing Value Analysis

In [ ]:
```
# Import Libraries
import missingno as msno
```

In [ ]:
```
# Numerical Analysis

# CITATION: Parts of code taken from HCDR_baseline_submission_phase2 starter code

missing_percentage = (df_app_train.isnull().sum() / df_app_train.isnull().count() * 1
missing_count = df_app_train.isna().sum().sort_values(ascending = False)
missing_app_table = pd.concat([missing_percentage, missing_count], axis=1, keys=["Mis
missing_app_table.head(20)
```

Out[ ]:

	Missing (%)	Missing (Count)
COMMONAREA_MEDI	69.87	214865
COMMONAREA_AVG	69.87	214865
COMMONAREA_MODE	69.87	214865
NONLIVINGAPARTMENTS_MODE	69.43	213514
NONLIVINGAPARTMENTS_AVG	69.43	213514
NONLIVINGAPARTMENTS_MEDI	69.43	213514
FONDKAPREMONT_MODE	68.39	210295
LIVINGAPARTMENTS_MODE	68.35	210199
LIVINGAPARTMENTS_AVG	68.35	210199
LIVINGAPARTMENTS_MEDI	68.35	210199
FLOORSMIN_AVG	67.85	208642
FLOORSMIN_MODE	67.85	208642
FLOORSMIN_MEDI	67.85	208642
YEARS_BUILD_MEDI	66.50	204488
YEARS_BUILD_MODE	66.50	204488
YEARS_BUILD_AVG	66.50	204488
OWN_CAR_AGE	65.99	202929
LANDAREA_MEDI	59.38	182590
LANDAREA_MODE	59.38	182590
LANDAREA_AVG	59.38	182590

In [ ]:
```
# Visualization of these Misssing Values
fig, ax = plt.subplots(1,1, sharex=False, figsize=(20,10))
ax = msno.bar(df_app_train.drop("TARGET", axis='columns').sample(10000))
ax.set_title("Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more missi
```

Out[ ]: Text(0.5, 1.0, 'Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more missing values from sample]')


Bar Plot of Present Values: Sample of 10000
[smaller bar = more missing values from sample]
0.0
0.2
0.4
0.6
0.8
1.0
AMT_REQ_CREDIT_BUREAU_YEAR 8704
AMT_REQ_CREDIT_BUREAU_QRT 8704
AMT_REQ_CREDIT_BUREAU_MON 8704
AMT_REQ_CREDIT_BUREAU_WEEK 8704
AMT_REQ_CREDIT_BUREAU_DAY 8704
AMT_REQ_CREDIT_BUREAU_HOUR 8704
FLAG_DOCUMENT_21 10000
FLAG_DOCUMENT_20 10000
FLAG_DOCUMENT_19 10000
FLAG_DOCUMENT_18 10000
FLAG_DOCUMENT_17 10000
FLAG_DOCUMENT_16 10000
FLAG_DOCUMENT_15 10000
FLAG_DOCUMENT_14 10000
FLAG_DOCUMENT_13 10000
FLAG_DOCUMENT_12 10000
FLAG_DOCUMENT_11 10000
FLAG_DOCUMENT_10 10000
FLAG_DOCUMENT_9 10000
FLAG_DOCUMENT_8 10000
FLAG_DOCUMENT_7 10000
FLAG_DOCUMENT_6 10000
FLAG_DOCUMENT_5 10000
FLAG_DOCUMENT_4 10000
FLAG_DOCUMENT_3 10000
FLAG_DOCUMENT_2 10000
DAYS_LAST_PHONE_CHANGE 10000
DEF_60_CNT_SOCIAL_CIRCLE 9966
OBS_60_CNT_SOCIAL_CIRCLE 9966
DEF_30_CNT_SOCIAL_CIRCLE 9966
OBS_30_CNT_SOCIAL_CIRCLE 9966
EMERGENCYSTATE_MODE 5276
WALLSMATERIAL_MODE 4942
TOTALAREA_MODE 5176
HOUSETYPE_MODE 5006
FONDKAPREMONT_MODE 3204
NONLIVINGAREA_MEDI 4523
NONLIVINGAPARTMENTS_MEDI 3121
LIVINGAREA_MEDI 4989
LIVINGAPARTMENTS_MEDI 3213
LANDAREA_MEDI 4082
FLOORSMIN_MEDI 3246
FLOORSMAX_MEDI 5031
ENTRANCES_MEDI 4956
ELEVATORS_MEDI 4702
COMMONAREA_MEDI 3058
YEARS_BUILD_MEDI 3413
YEARS_BEGINEXPLUATATION_MEDI 5134
BASEMENTAREA_MEDI 4118
APARTMENTS_MEDI 4943
NONLIVINGAREA_MODE 4523
NONLIVINGAPARTMENTS_MODE 3121
LIVINGAREA_MODE 4989
LIVINGAPARTMENTS_MODE 3213
LANDAREA_MODE 4082
FLOORSMIN_MODE 3246
FLOORSMAX_MODE 5031
ENTRANCES_MODE 4956
ELEVATORS_MODE 4702
COMMONAREA_MODE 3058
YEARS_BUILD_MODE 3413
YEARS_BEGINEXPLUATATION_MODE 5134
BASEMENTAREA_MODE 4118
APARTMENTS_MODE 4943
NONLIVINGAREA_AVG 4523
NONLIVINGAPARTMENTS_AVG 3121
LIVINGAREA_AVG 4989
LIVINGAPARTMENTS_AVG 3213
LANDAREA_AVG 4082
FLOORSMIN_AVG 3246
FLOORSMAX_AVG 5031
ENTRANCES_AVG 4956
ELEVATORS_AVG 4702
COMMONAREA_AVG 3058
YEARS_BUILD_AVG 3413
YEARS_BEGINEXPLUATATION_AVG 5134
BASEMENTAREA_AVG 4118
APARTMENTS_AVG 4943
EXT_SOURCE_3 8072
EXT_SOURCE_2 9978
EXT_SOURCE_1 4417
ORGANIZATION_TYPE 10000
LIVE_CITY_NOT_WORK_CITY 10000
REG_CITY_NOT_WORK_CITY 10000
REG_CITY_NOT_LIVE_CITY 10000
LIVE_REGION_NOT_WORK_REGION 10000
REG_REGION_NOT_WORK_REGION 10000
REG_REGION_NOT_LIVE_REGION 10000
HOUR_APPR_PROCESS_START 10000
WEEKDAY_APPR_PROCESS_START 10000
REGION_RATING_CLIENT_W_CITY 10000
REGION_RATING_CLIENT 10000
CNT_FAM_MEMBERS 9999
OCCUPATION_TYPE 6898
FLAG_EMAIL 10000
FLAG_PHONE 10000
FLAG_CONT_MOBILE 10000
FLAG_WORK_PHONE 10000
FLAG_EMP_PHONE 10000
FLAG_MOBIL 10000
OWN_CAR_AGE 3395
DAYS_ID_PUBLISH 10000
DAYS_REGISTRATION 10000
DAYS_EMPLOYED 10000
DAYS_BIRTH 10000
REGION_POPULATION_RELATIVE 10000
NAME_HOUSING_TYPE 10000
NAME_FAMILY_STATUS 10000
NAME_EDUCATION_TYPE 10000
NAME_INCOME_TYPE 10000
NAME_TYPE_SUITE 9951
AMT_GOODS_PRICE 9988
AMT_ANNUITY 9999
AMT_CREDIT 10000
AMT_INCOME_TOTAL 10000
CNT_CHILDREN 10000
FLAG_OWN_REALTY 10000
FLAG_OWN_CAR 10000
CODE_GENDER 10000
NAME_CONTRACT_TYPE 10000
SK_ID_CURR 10000
0
2000
4000
6000
8000
10000

DISCUSSION </br> After reviewing the visualization and the numerical metrics of the missing values it seems that most of the missing values come from computational features. By computational, it is meant that features that have a mean, median, mode, or average tag on the features are more likely to have missing values. This may be helpful in feature reduction and selection.

VEDA: Input Feature Visualiaztion (bureau.csv)

VEDA: Correlation Anlaysis: beureau.csv

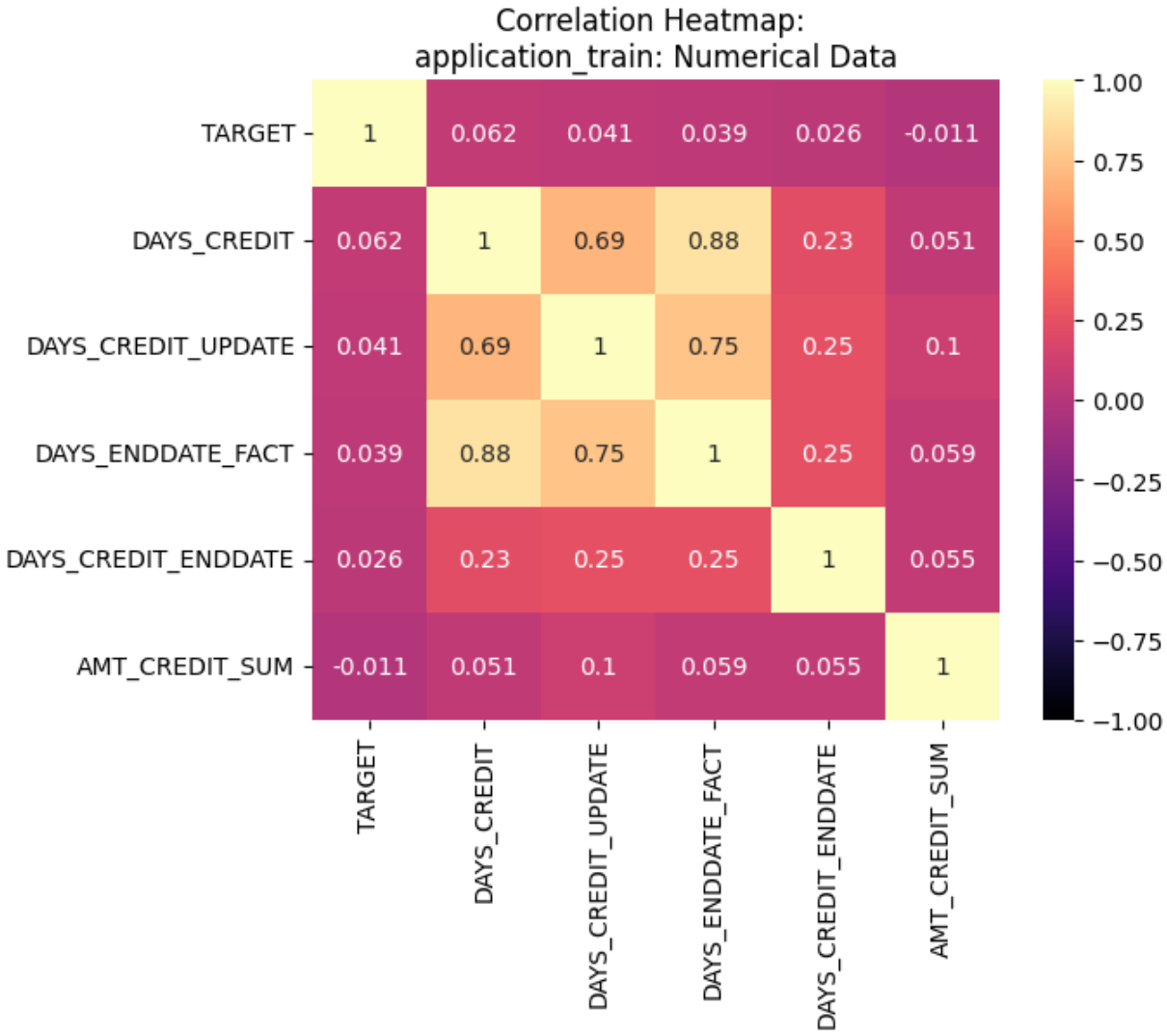
PREFACE </br> since the rest of the tables of the data set are vast in features and efficent extraction of these features are important, we should first look at the correlation of these features before finding the distributions and other visual exploritory data analysis for the sake of efficency.

In [ ]:
```
import pandas as pd

bureau_merg_targets = pd.merge(df_bureau, df_app_train[['SK_ID_CURR', 'TARGET']], on=
bureau_corr = bureau_merg_targets.corr()['TARGET']
bureau_corr_sorted = bureau_corr.abs().sort_values(ascending=False)

## Show the top correlated
bureau_corr_sorted.head(10)

## select the top 5 correlated features including the target
n=5
bureau_top_feat = bureau_corr_sorted[0:n+1].index.tolist()

## Lets put these features into a dataframe with thier original values with the targe
df_bureau_top_feat = bureau_merg_targets[bureau_top_feat]
```

```
<ipython-input-293-61ddce5ef669>:4: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  bureau_corr = bureau_merg_targets.corr()['TARGET']
```

In [ ]:
```
corr_data = df_bureau_top_feat
corr_data = corr_data.corr()
sns.heatmap(corr_data, cmap="magma", annot=True, vmin= -1.0, vmax=1.0)
plt.title("Correlation Heatmap: \napplication_train: Numerical Data")
```

Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap: \napplication_train: Numerical Data')
```

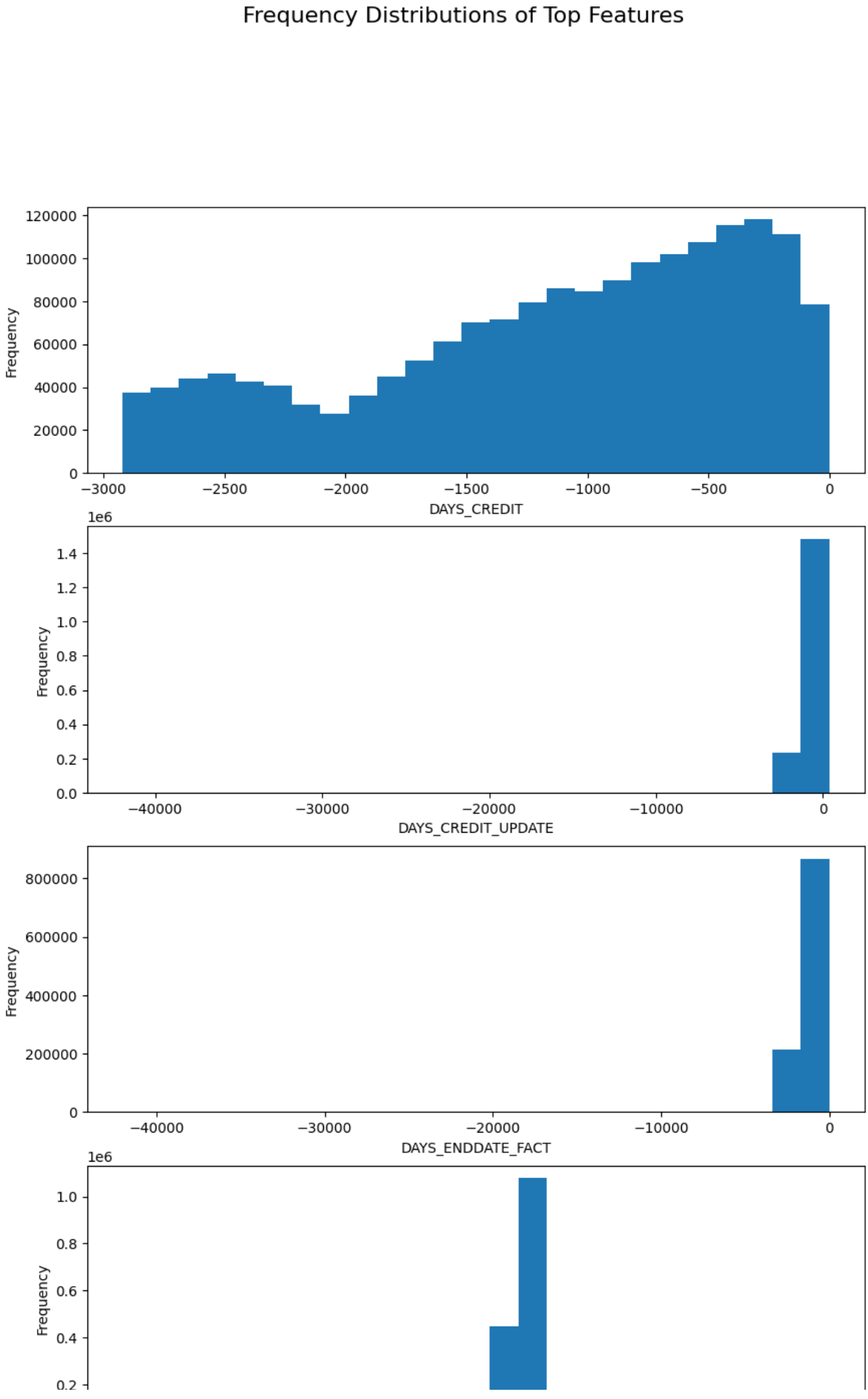


DISCUSSION </br> This heat map gives us a great sense of the inital features we will be interested in from the bureau.csv table, top amoung those are the DAYS_CREDIT, DAYS_CREDIT_UPDATE, DAYS_ENDATE_FACT, and DAYS_CREDIT_ENDATE. AMT_CREDIT_SUM has the lowest degree of correlation, so it could possibly be ignored.

VEDA: Distribution Anlaysis: beureau.csv

In [ ]:

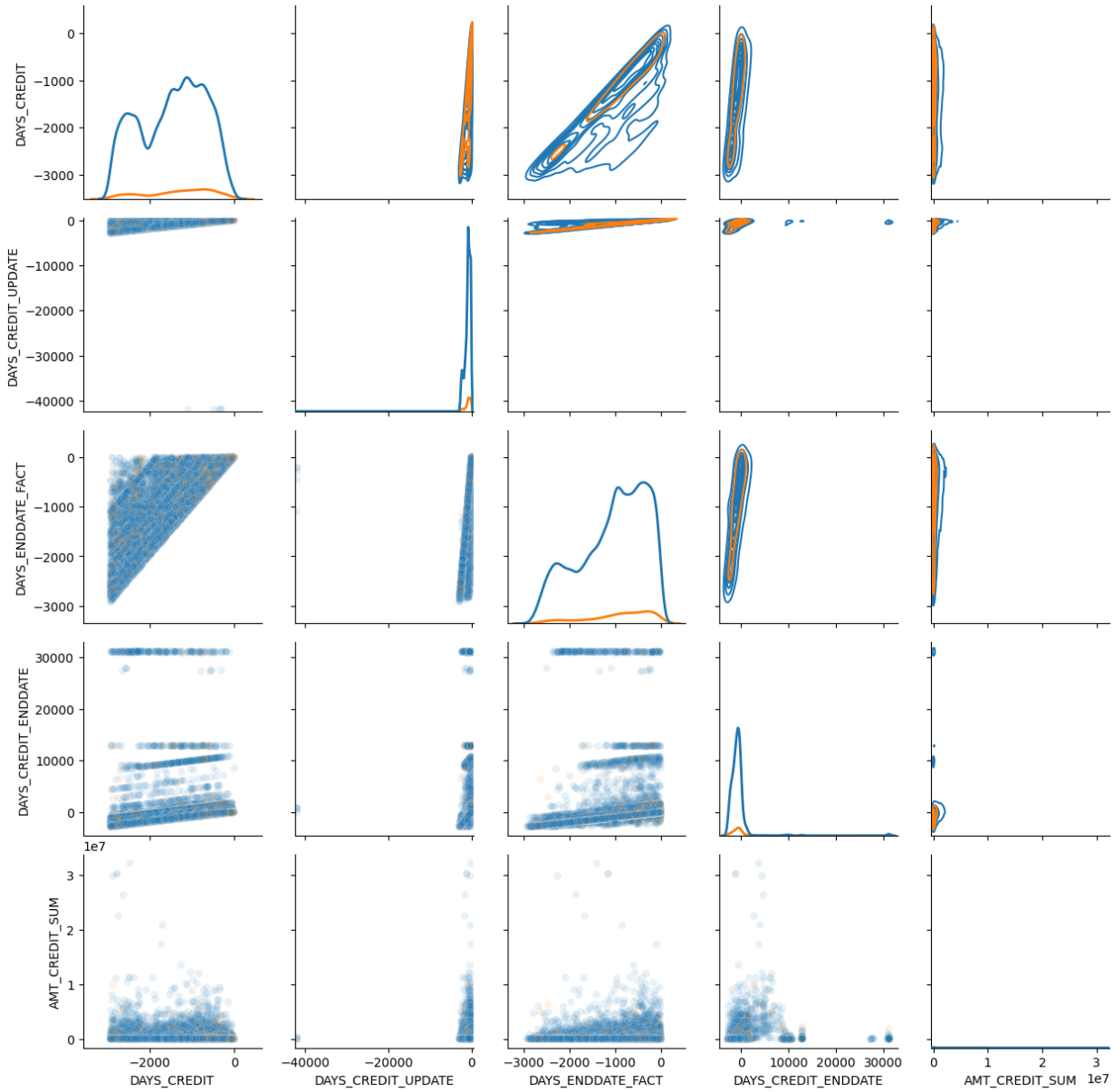
```
# Lets look at the distributions of these data
fig, axs = plt.subplots(nrows=n, figsize=(10,20))
fig.suptitle('Frequency Distributions of Top Features', fontsize=16)
for i, feature in enumerate(bureau_top_feat):
  axs[i - 1].hist(df_bureau_top_feat[feature], bins=25)
  axs[i - 1].set_xlabel(feature)
  axs[i - 1].set_ylabel("Frequency")

plt.show()
```


Frequency Distributions of Top Features
Frequency
DAYS_CREDIT
1e6
Frequency
DAYS_CREDIT_UPDATE
Frequency
DAYS_ENDDATE_FACT
1e6
Frequency

VEDA: Pairwise Anlaysis: beureau.csv

In [ ]:
```
 # Data size is too large for pair plot so we will reduce
df_pair_plot = df_bureau_top_feat.dropna()
df_pair_plot = df_pair_plot.iloc[:50000]

g = sns.PairGrid(df_pair_plot, hue = "TARGET")
g.map_upper(sns.kdeplot)
g.map_diag(sns.kdeplot, lw=2)
g.map_lower(sns.scatterplot, alpha = 0.1)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f92ab6eceb0>
```



VEDA: Missing Value Anlaysis: beureau.csv

In [ ]:

```
# Numerical Analysis

# CITATION: Parts of code taken from HCDR_baseline_submission_phase2 starter code

missing_percentage = (df_bureau.isnull().sum() / df_bureau.isnull().count() * 100).so
missing_count = df_bureau.isna().sum().sort_values(ascending = False)
missing_app_table = pd.concat([missing_percentage, missing_count], axis=1, keys=["Mis
missing_app_table.head(20)
```

Out[ ]:

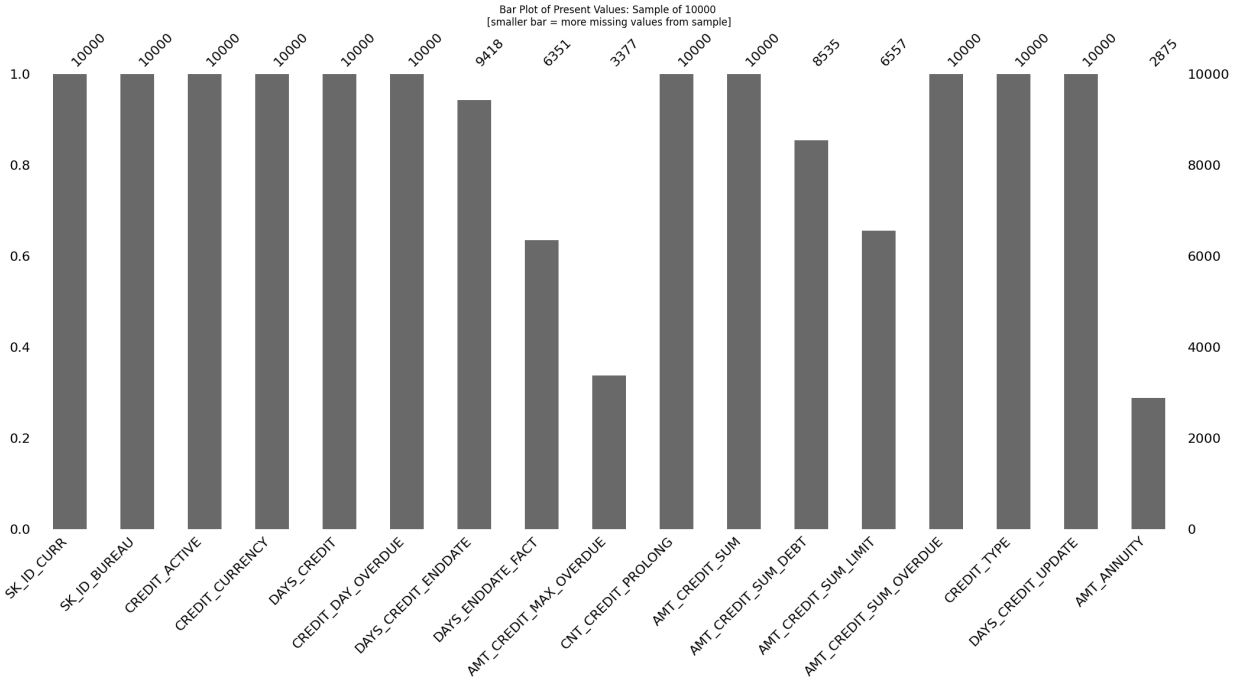
	Missing (%)	Missing (Count)
AMT_ANNUITY	71.47	1226791
AMT_CREDIT_MAX_OVERDUE	65.51	1124488
DAYS_ENDDATE_FACT	36.92	633653
AMT_CREDIT_SUM_LIMIT	34.48	591780
AMT_CREDIT_SUM_DEBT	15.01	257669
DAYS_CREDIT_ENDDATE	6.15	105553
AMT_CREDIT_SUM	0.00	13
CREDIT_ACTIVE	0.00	0
CREDIT_CURRENCY	0.00	0
DAYS_CREDIT	0.00	0
CREDIT_DAY_OVERDUE	0.00	0
SK_ID_BUREAU	0.00	0
CNT_CREDIT_PROLONG	0.00	0
AMT_CREDIT_SUM_OVERDUE	0.00	0
CREDIT_TYPE	0.00	0
DAYS_CREDIT_UPDATE	0.00	0
SK_ID_CURR	0.00	0

In [ ]:

```
# Visualization of these Misssing Values
fig, ax = plt.subplots(1,1, sharex=False, figsize=(20,10))
ax = msno.bar(df_bureau.sample(10000))
ax.set_title("Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more missi
```

Out[ ]:

```
Text(0.5, 1.0, 'Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more mis
sing values from sample]')
```



DISCUSSION </br> We can see from the sample that AMT_CREDIT_MAX_OVERDUE and AMT_ANNUITY both have the most missing values from the data table.

VEDA: Input Feature Visualiaztion (bureau_balance.csv)

VEDA: Correlation Anlaysis: beureau_balance.csv

In [ ]:
```
import pandas as pd
bur_bal_id_merge = pd.merge(df_bureau_bal, df_bureau[['SK_ID_BUREAU', 'SK_ID_CURR']],
bur_bal_target = pd.merge(bur_bal_id_merge, df_app_train[['SK_ID_CURR', 'TARGET']], o
bureau_bal_corr = bur_bal_target.corr()['TARGET']
bureau_bal_corr_sorted = bureau_bal_corr.abs().sort_values(ascending=False)

bureau_bal_top_feat = bureau_bal_corr_sorted[0:2].index.tolist()
bureau_bal_top_feat = bur_bal_target[bureau_bal_top_feat]
```

```
<ipython-input-299-9536630e7185>:4: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  bureau_bal_corr = bur_bal_target.corr()['TARGET']
```

In [ ]:
```
bureau_bal_top_feat
```

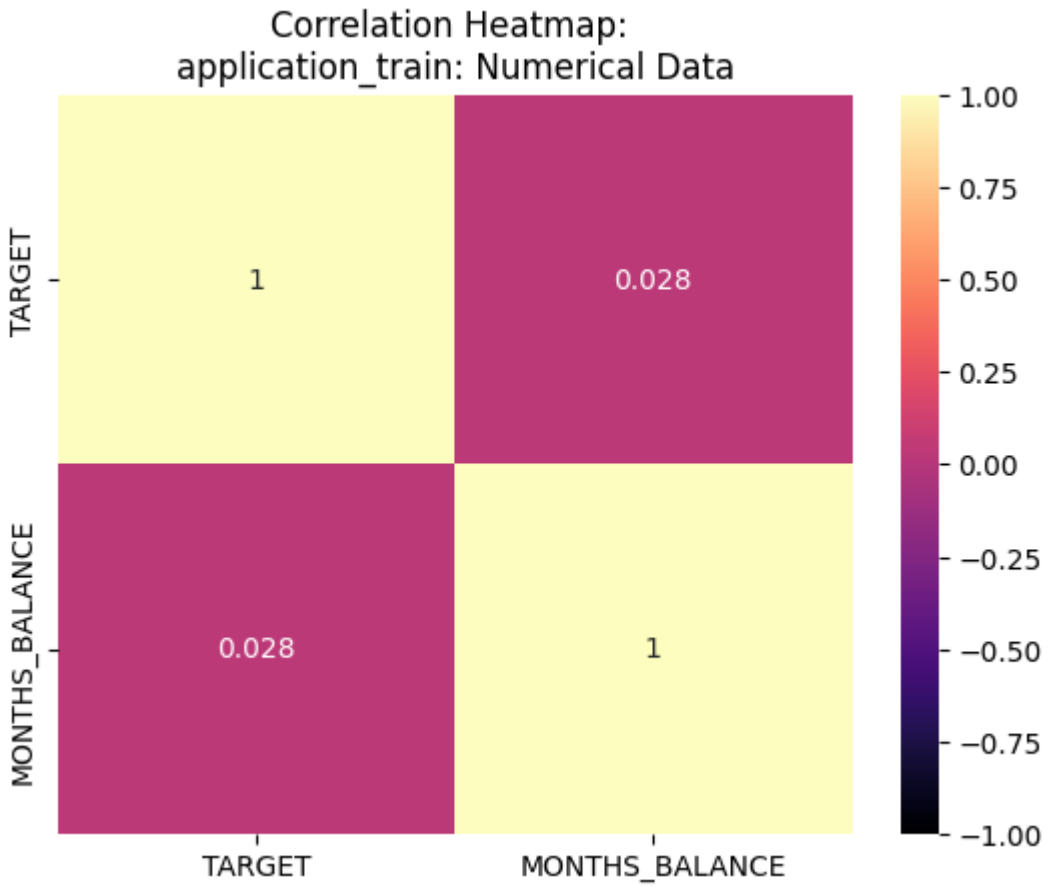
Out[ ]:

	TARGET	MONTHS_BALANCE
0	0.0	0
1	0.0	-1
2	0.0	-2
3	0.0	-3
4	0.0	-4
...	...	...
27299920	1.0	-47
27299921	1.0	-48
27299922	1.0	-49
27299923	1.0	-50
27299924	1.0	-51

27299925 rows × 2 columns

In [ ]:
```
corr_data = bureau_bal_top_feat
corr_data = corr_data.corr()
sns.heatmap(corr_data, cmap="magma", annot=True, vmin= -1.0, vmax=1.0)
plt.title("Correlation Heatmap: \napplication_train: Numerical Data")
```

Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap: \napplication_train: Numerical Data')
```



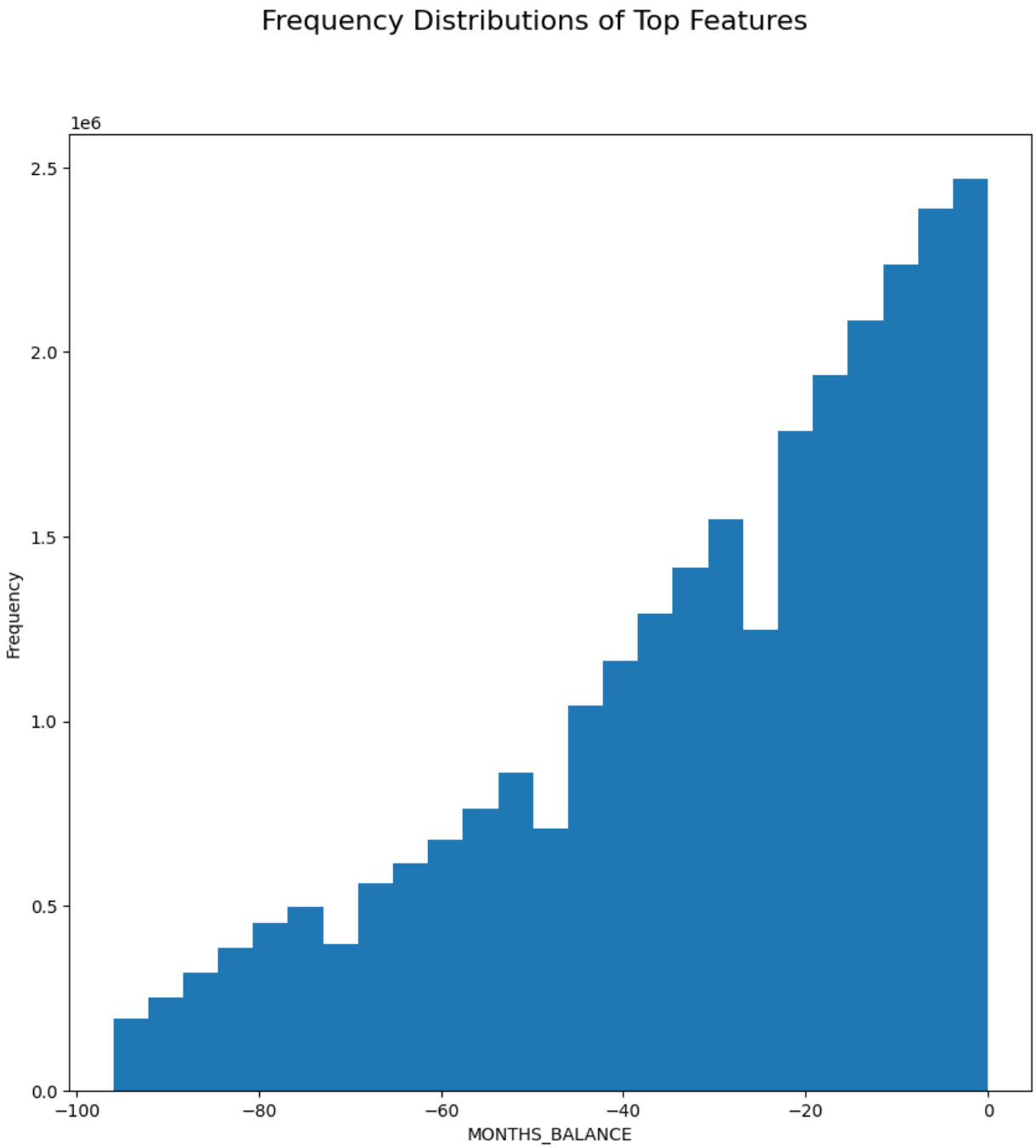
DISCUSSION </br> From this heat map and the previous correlation analysis, we can see that the only feasible feature from the buraeu_balance table is the MONTHS_BLANCE. After further analysis it can be said that this feature does have some correlation positivly with the target, meaning that as the MONTHS_BALANCE increases there is a decrease in the rate of repayment

---

VEDA: Distribution Anlaysis: beureau_balance.csv

In [ ]:

```
# Lets look at the distributions of these data
fig, axs = plt.subplots(1, figsize=(10,10))
fig.suptitle('Frequency Distributions of Top Features', fontsize=16)
axs.hist(df_bureau_bal['MONTHS_BALANCE'], bins=25)
axs.set_xlabel("MONTHS_BALANCE")
axs.set_ylabel("Frequency")

plt.show()
```



DISCUSSION </br> Observing the top feature of the buraeu_balance table, we can see that the distribution is unimodal, meaning that there seems to be a large grouping towards the 0 value. This imbalance in the distribution could help us later handle the values of this feauture upon implementation.

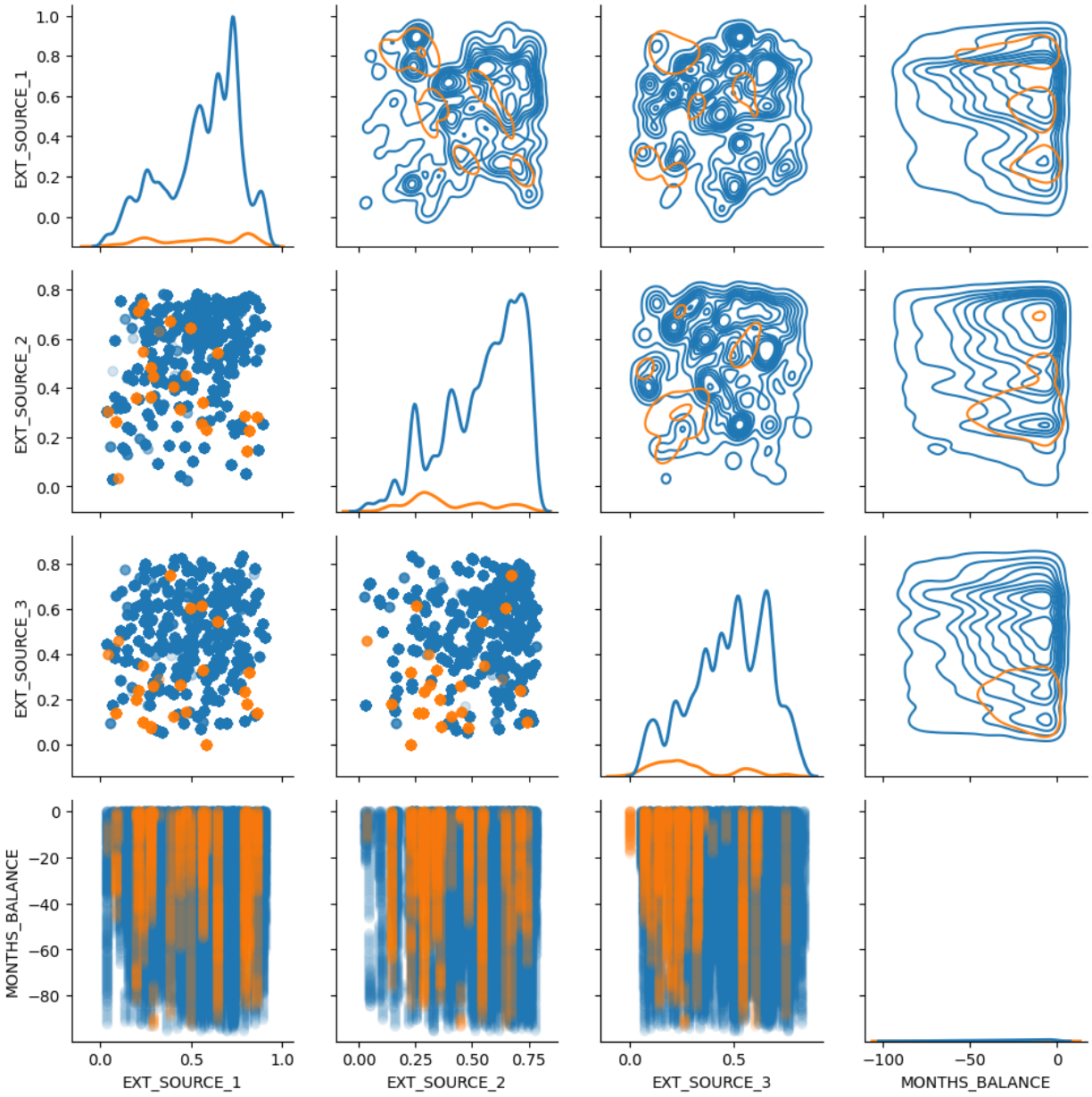
VEDA: Pairwise Anlaysis: beureau_balance.csv

DISCLAIMER </br> Since there was only one feature that was deemed to be of significance it is dificult to map this to any pairwise plot inside the same table, but we can use other tables to observe this. Lets compare this to the external data found in the application_train.csv

In [ ]:
```
# Lets create a pair plot of the numerical data used in some of these heat maps
df_pair_plot = df_app_train[["TARGET", "EXT_SOURCE_1", "EXT_SOURCE_2", "EXT_SOURCE_3"
```

```
#df_pair_plot["MONTHS_BALANCE"] =
df_pair_plot = pd.merge(df_pair_plot, df_bureau[['SK_ID_CURR', 'SK_ID_BUREAU']], on='
df_pair_plot = pd.merge(df_pair_plot, df_bureau_bal[['SK_ID_BUREAU', 'MONTHS_BALANCE'
# Data size is too large for pair plot so we will reduce
df_pair_plot = df_pair_plot.dropna()
df_pair_plot = df_pair_plot.drop(["SK_ID_CURR", 'SK_ID_BUREAU'], axis=1)
df_pair_plot = df_pair_plot.iloc[:50000]

g = sns.PairGrid(df_pair_plot, hue = "TARGET")
g.map_upper(sns.kdeplot)
g.map_diag(sns.kdeplot, lw=2)
g.map_lower(plt.scatter, alpha = 0.1)
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x7f92a5c6adc0>



DISCUSION </br> Looking at the KDE plots of the MONTHS_BALACNE, we can see that the target == 1 in oragnge is topologically aligned with the EXT_SOURCE_1. This is quite interesting because of the disimilarity in data tables they come from. This reitterated the fact that MONTHS_BALANCE is an effective feature to include.

VEDA: Missing Value Anlaysis: beureau_balance.csv

There are no missing values in this table

VEDA: Input Feature Visualiaztion (POS_CASH_balance.csv)

VEDA: Correlation Anlaysis: POS_CASH_balance.csv

In [ ]:
```
import pandas as pd

pos_cash_target_merge = pd.merge(df_pos_cash_bal, df_app_train[['SK_ID_CURR', 'TARGET
pos_cash_corr = pos_cash_target_merge.corr()['TARGET']
pos_cash_corr_sorted = pos_cash_corr.abs().sort_values(ascending=False)

## Show the top correlated
pos_cash_corr_sorted.head(10)

## select the top 5 correlated features including the target
n=4
pos_cash_top_feat_list = pos_cash_corr_sorted[0:n+1].index.tolist()

## Lets put these features into a dataframe with thier original values with the targe
pos_cash_top_feat = pos_cash_target_merge[pos_cash_top_feat_list]
```

```
<ipython-input-304-56a417d1a489>:4: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  pos_cash_corr = pos_cash_target_merge.corr()['TARGET']
```

In [ ]:
```
print(pos_cash_top_feat_list)
```

```
['TARGET', 'CNT_INSTALMENT_FUTURE', 'MONTHS_BALANCE', 'CNT_INSTALMENT', 'SK_DPD']
```

In [ ]:
```
corr_data = pos_cash_top_feat
corr_data = corr_data.corr()
sns.heatmap(corr_data, cmap="magma", annot=True, vmin= -1.0, vmax=1.0)
plt.title("Correlation Heatmap: \npos_cash_balance: Numerical Data")
```

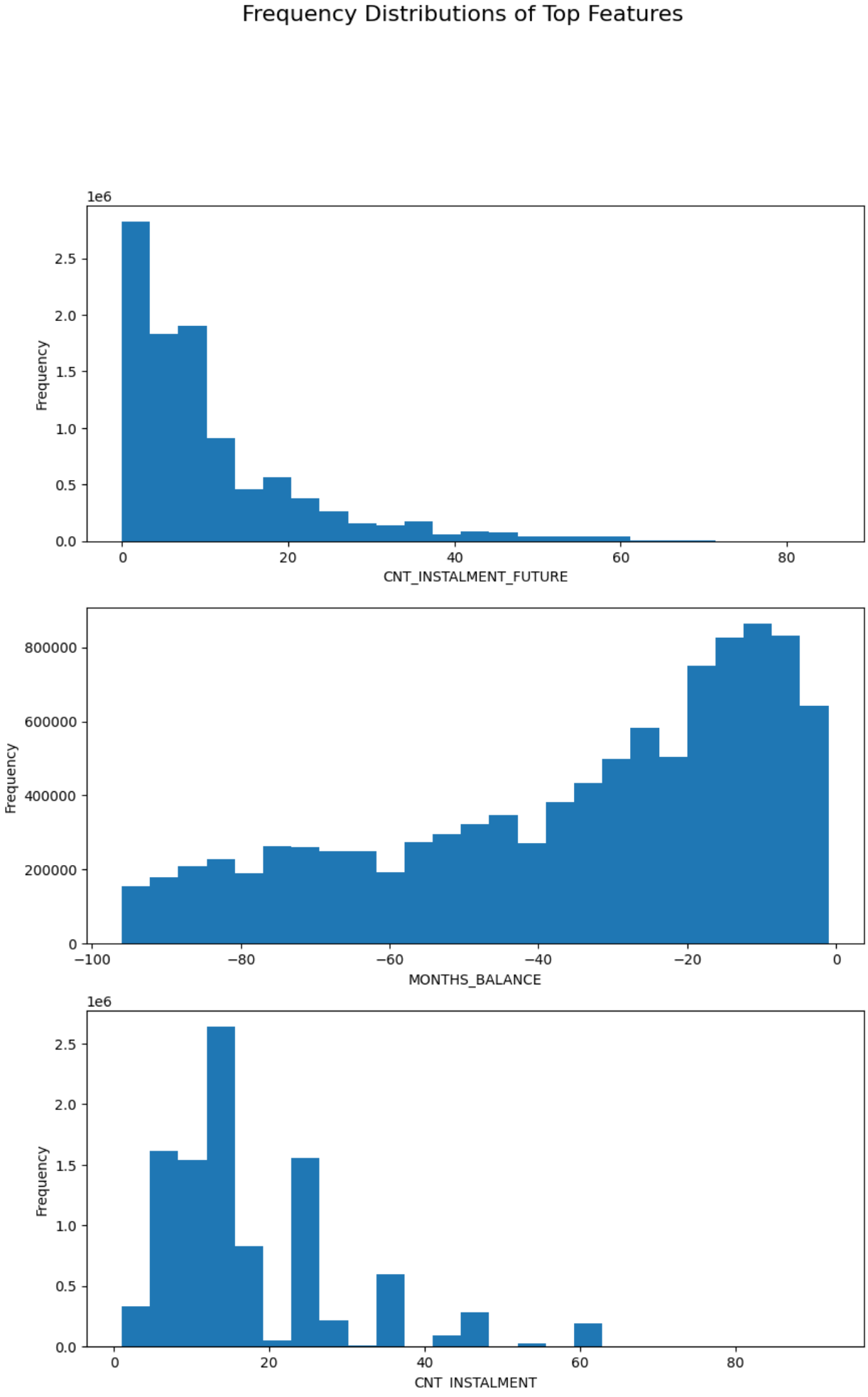
Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap: \npos_cash_balance: Numerical Data')
```

DISCUSSION </br> From this heat map we can see that CNT_INSTALMENT_FUTURE, MONTHS_BALANCE, and CNT_INSTALMENT are all features that have some correlation positivly to the target variable. This means that an increase in any one of these features should see a higher rate in the failure to repay the loan.

VEDA: Distribution Anlaysis: POS_CASH_balance.csv

In [ ]:

```
# Lets look at the distributions of these data
fig, axs = plt.subplots(nrows=n, figsize=(10,20))
fig.suptitle('Frequency Distributions of Top Features', fontsize=16)
for i, feature in enumerate(pos_cash_top_feat_list):
  axs[i - 1].hist(pos_cash_top_feat[feature], bins=25)
  axs[i - 1].set_xlabel(feature)
  axs[i - 1].set_ylabel("Frequency")

plt.show()
```

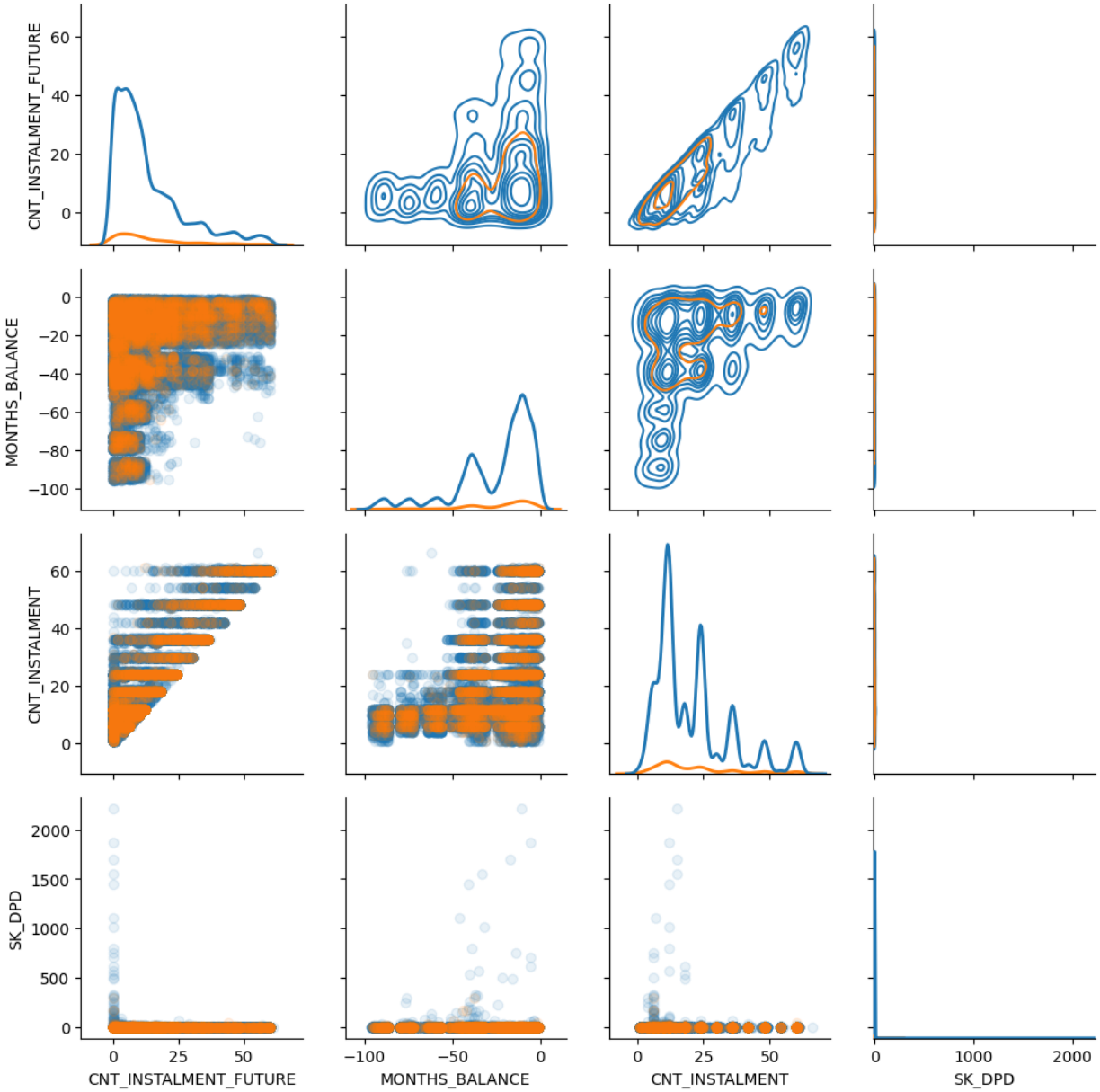

Frequency Distributions of Top Features
1e6
Frequency
CNT_INSTALMENT_FUTURE
MONTHS_BALANCE
1e6
Frequency
CNT_INSTALMENT

DISCUSSION </br> We can see from these distributions that all of these are imbalanced unimodal distributions. This is important to take into account when we move to the stage of handling these features in feature selection and preprocessing.

VEDA: Pairwise Anlaysis: POS_CASH_balance.csv

In [ ]:
```
df_pair_plot = pos_cash_top_feat
df_pair_plot = df_pair_plot.dropna()
df_pair_plot = df_pair_plot.iloc[:50000]

g = sns.PairGrid(df_pair_plot, hue = "TARGET")
g.map_upper(sns.kdeplot)
g.map_diag(sns.kdeplot, lw=2)
g.map_lower(plt.scatter, alpha = 0.1)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f92a65c4eb0>
```



DISCUSSION Forom this pairwise plot, we can observe again the topology of the target value = 1 in the KDE plot of CNT_INSTALMENT and MONTHS_BALANCE is overlapping at its most dense point. This again affirms that the correlation behind these features and the target value is correct.

VEDA: Missing Value Anlaysis: POS_CASH_balance.csv

In [ ]:

```
# Numerical Analysis

# CITATION: Parts of code taken from HCDR_baseline_submission_phase2 starter code

missing_percentage = (df_pos_cash_bal.isnull().sum() / df_pos_cash_bal.isnull().count
missing_count = df_pos_cash_bal.isna().sum().sort_values(ascending = False)
missing_app_table = pd.concat([missing_percentage, missing_count], axis=1, keys=["Mis
missing_app_table.head(20)
```

Out[ ]:

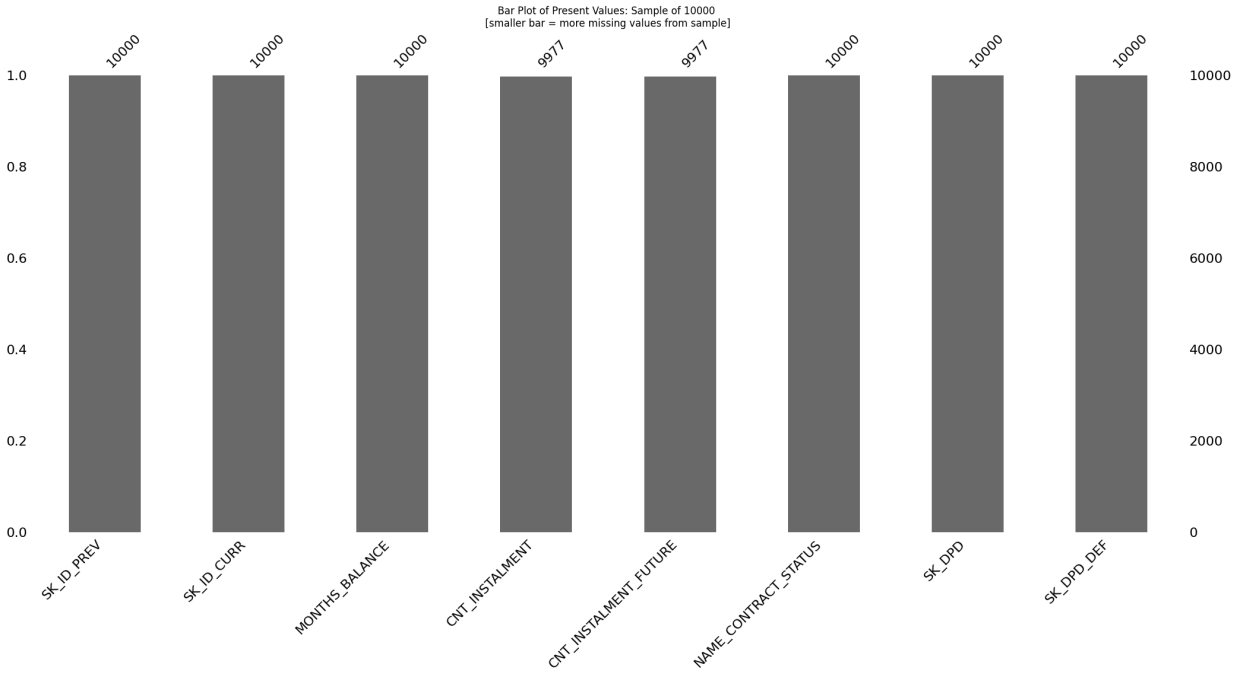
	Missing (%)	Missing (Count)
CNT_INSTALMENT_FUTURE	0.26	26087
CNT_INSTALMENT	0.26	26071
SK_ID_PREV	0.00	0
SK_ID_CURR	0.00	0
MONTHS_BALANCE	0.00	0
NAME_CONTRACT_STATUS	0.00	0
SK_DPD	0.00	0
SK_DPD_DEF	0.00	0

In [ ]:

```
# Visualization of these Misssing Values
fig, ax = plt.subplots(1,1, sharex=False, figsize=(20,10))
ax = msno.bar(df_pos_cash_bal.sample(10000))
ax.set_title("Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more missi
```

Out[ ]:

```
Text(0.5, 1.0, 'Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more mis
sing values from sample]')
```



DISCUSSION </br> From this graph and the previous numerical analysis, we can see the only features missing values are the CNT_INSTALMETNS_FUTURE and CNT_INSTALMENTS which are both features we are interested in, so imputing this later will be a worth while task if the percentage was higher.

VEDA: Input Feature Visualiaztion (credit_card_balance.csv)

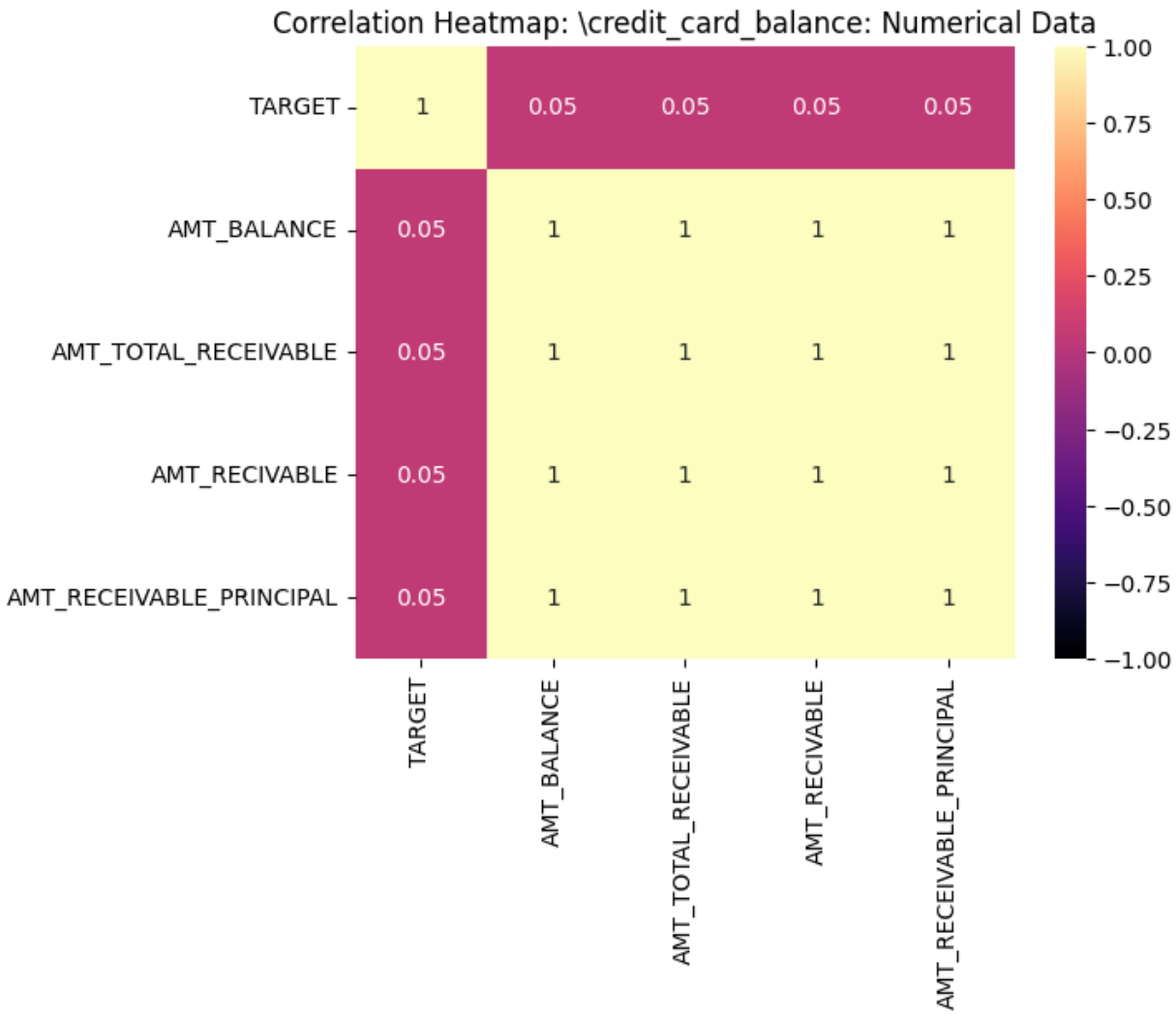
VEDA: Correlation Analysis: credict_card_balance.csv

In [ ]:
```
import pandas as pd

pos_credit_target_merge = pd.merge(df_credit_card_bal, df_app_train[['SK_ID_CURR', 'T
pos_credit_corr = pos_credit_target_merge.corr()['TARGET']
pos_credit_corr_sorted = pos_credit_corr.abs().sort_values(ascending=False)

## Show the top correlated
pos_credit_corr_sorted.head(10)

## select the top 5 correlated features including the target
n=4
pos_credit_top_feat_list = pos_credit_corr_sorted[0:n+1].index.tolist()

## Lets put these features into a dataframe with thier original values with the targe
pos_credit_top_feat = pos_credit_target_merge[pos_credit_top_feat_list]
```

```
<ipython-input-311-c9b13e518754>:4: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  pos_credit_corr = pos_credit_target_merge.corr()['TARGET']
```

In [ ]:
```
corr_data = pos_credit_top_feat
corr_data = corr_data.corr()
sns.heatmap(corr_data, cmap="magma", annot=True, vmin= -1.0, vmax=1.0)
plt.title("Correlation Heatmap: \credit_card_balance: Numerical Data")
```

Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap: \\credit_card_balance: Numerical Data')
```

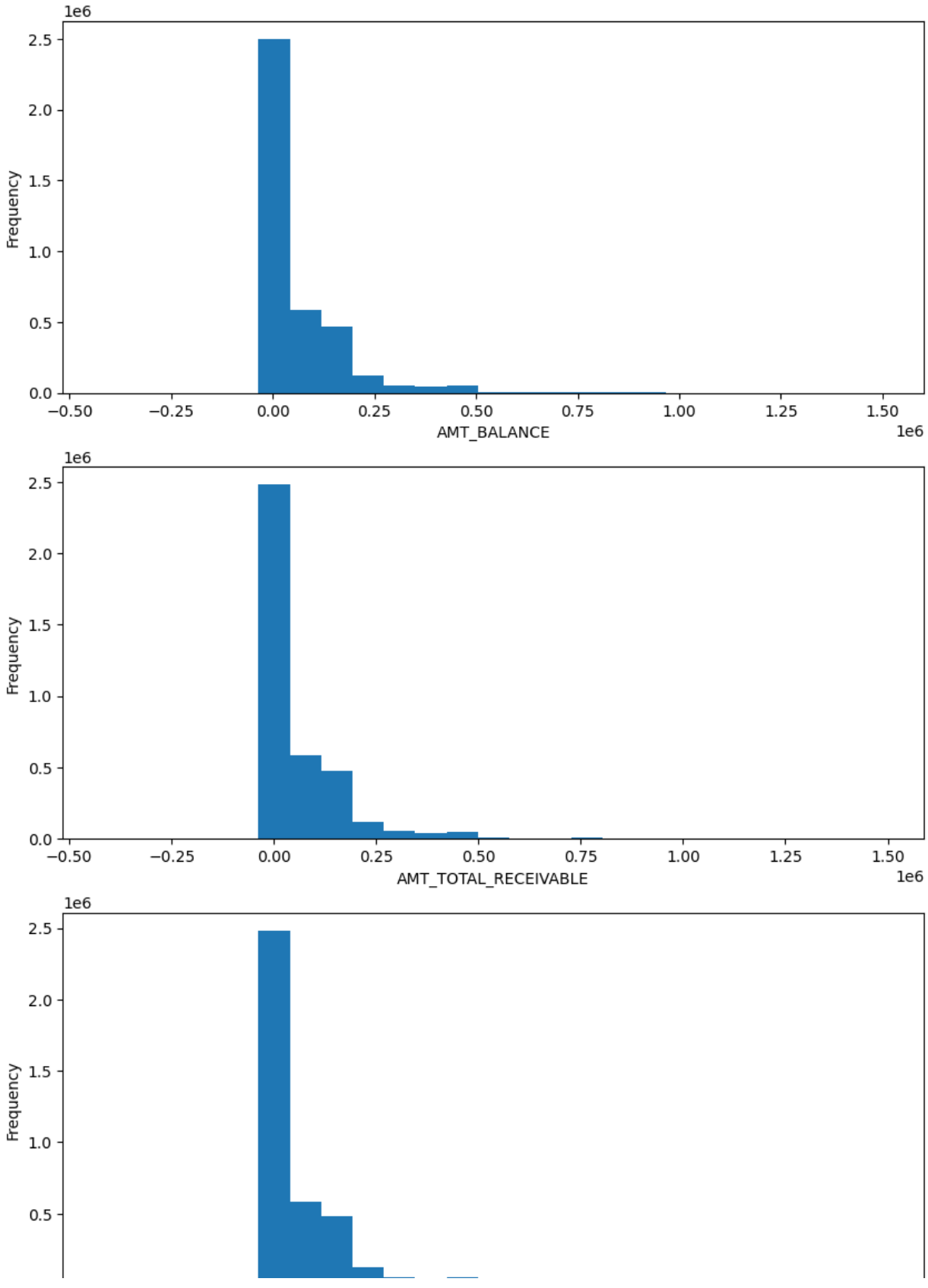


DISCUSSION </br> It seems that these data must have some artifacts or characteristics about them that throw off the heat map. Possibly they are having an interesting interaction with target variable.

VEDA: Distribution Analysis: credict_card_balance.csv

In [ ]:

```
# Lets look at the distributions of these data
fig, axs = plt.subplots(nrows=n, figsize=(10,20))
fig.suptitle('Frequency Distributions of Top Features', fontsize=16)
for i, feature in enumerate(pos_credit_top_feat_list):
  axs[i - 1].hist(pos_credit_top_feat[feature], bins=25)
  axs[i - 1].set_xlabel(feature)
  axs[i - 1].set_ylabel("Frequency")

plt.show()
```

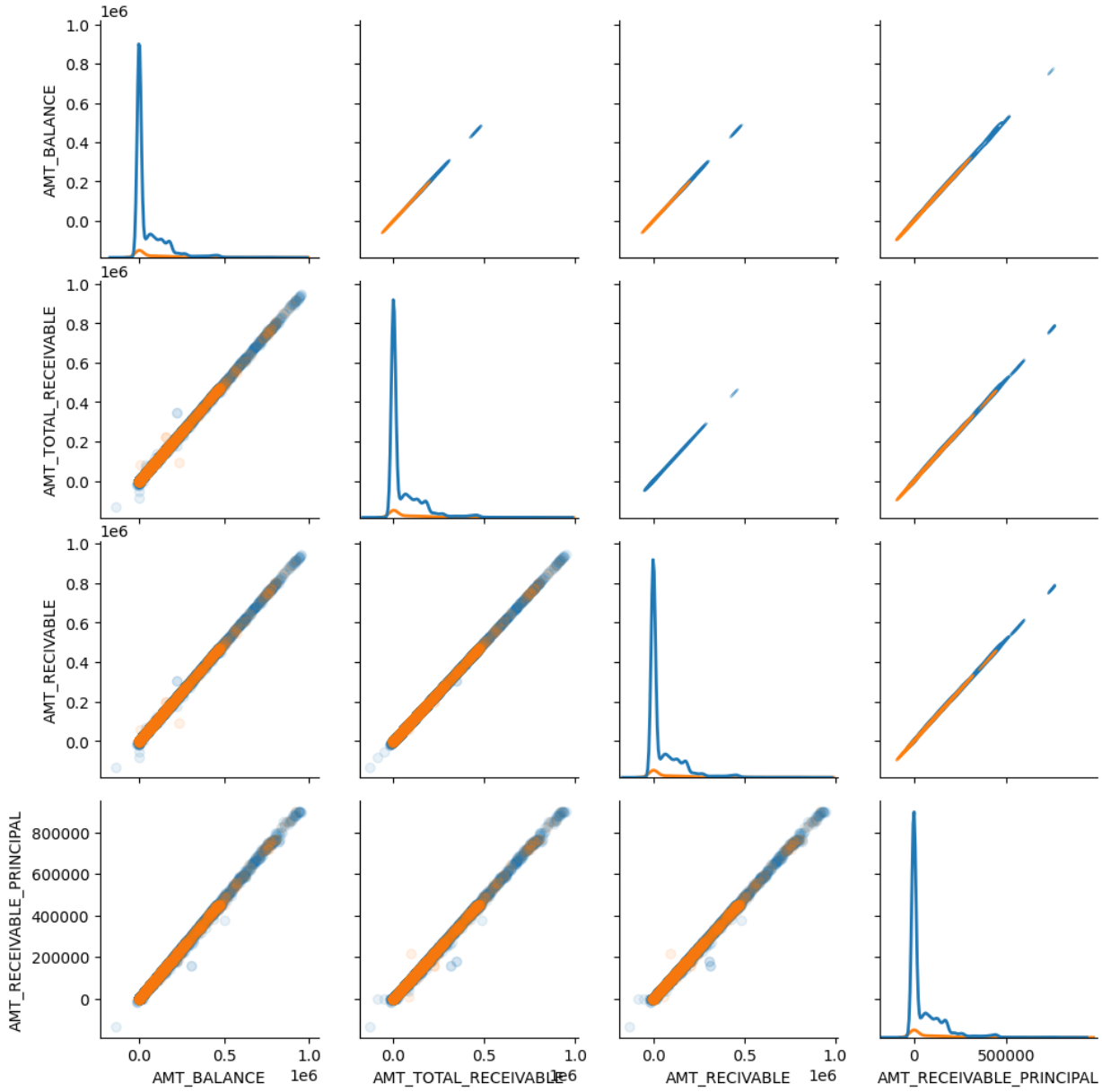

Frequency Distributions of Top Features
1e6
2.5
2.0
1.5
1.0
0.5
0.0
Frequency
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
AMT_BALANCE
1e6
1e6
AMT_TOTAL_RECEIVABLE
1e6
1e6

DISCUSSION This exaplains some of the artifacts that we saw on the heat map. It would seem that this data is normalized, having a sqewed distribution towards the mode at 0.

VEDA: Pairwise Analysis: credict_card_balance.csv

In [ ]:
```
df_pair_plot = pos_credit_top_feat
df_pair_plot = df_pair_plot.dropna()
df_pair_plot = df_pair_plot.iloc[:50000]

g = sns.PairGrid(df_pair_plot, hue = "TARGET")
g.map_upper(sns.kdeplot)
g.map_diag(sns.kdeplot, lw=2)
g.map_lower(plt.scatter, alpha = 0.1)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f92a4fca5b0>
```



DISCUSSION </br>

Again, we see this interesting artifact where these normalized features are linearlly aligning with the target value, this seems as though we should do some transformation of these data before using them in our data set.

VEDA: Missing Data Analysis: credict_card_balance.csv

In [ ]:

```
# Numerical Analysis

# CITATION: Parts of code taken from HCDR_baseline_submission_phase2 starter code

missing_percentage = (df_credit_card_bal.isnull().sum() / df_credit_card_bal.isnull()
missing_count = df_credit_card_bal.isna().sum().sort_values(ascending = False)
missing_app_table = pd.concat([missing_percentage, missing_count], axis=1, keys=["Mis
missing_app_table.head(20)
```

Out[ ]:

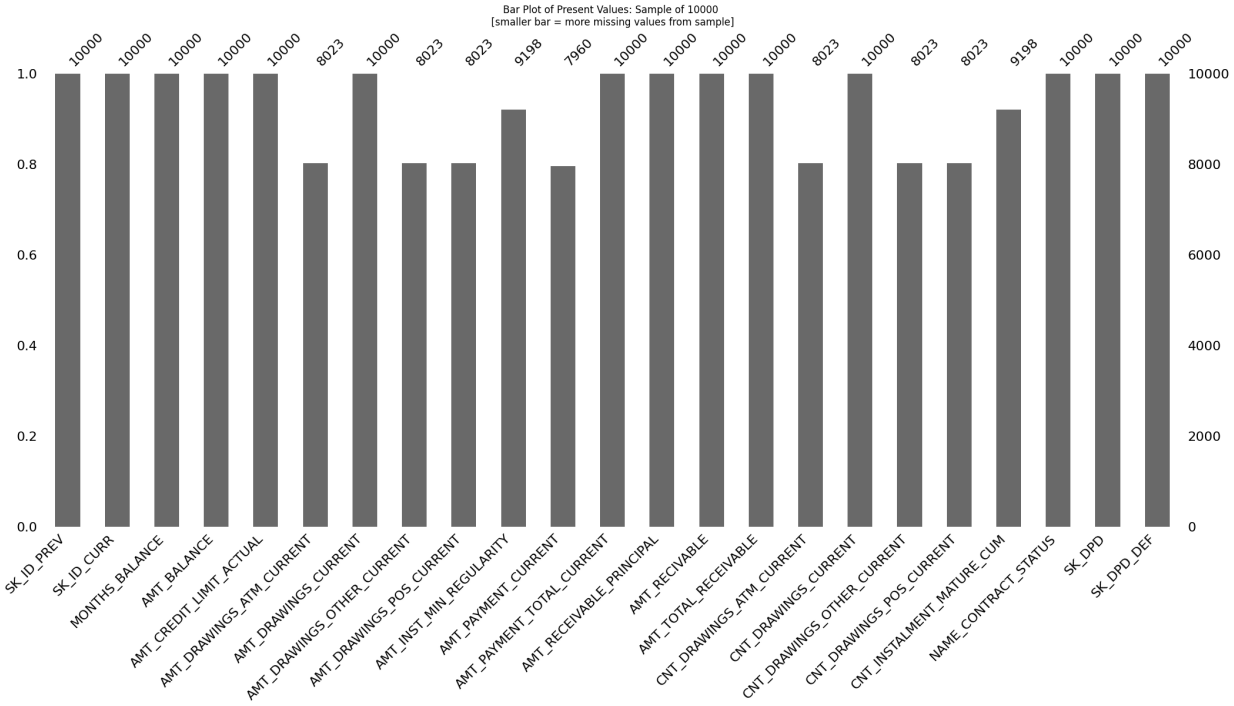
	Missing (%)	Missing (Count)
AMT_PAYMENT_CURRENT	20.00	767988
AMT_DRAWINGS_ATM_CURRENT	19.52	749816
CNT_DRAWINGS_POS_CURRENT	19.52	749816
AMT_DRAWINGS_OTHER_CURRENT	19.52	749816
AMT_DRAWINGS_POS_CURRENT	19.52	749816
CNT_DRAWINGS_OTHER_CURRENT	19.52	749816
CNT_DRAWINGS_ATM_CURRENT	19.52	749816
CNT_INSTALMENT_MATURE_CUM	7.95	305236
AMT_INST_MIN_REGULARITY	7.95	305236
SK_ID_PREV	0.00	0
AMT_TOTAL_RECEIVABLE	0.00	0
SK_DPD	0.00	0
NAME_CONTRACT_STATUS	0.00	0
CNT_DRAWINGS_CURRENT	0.00	0
AMT_PAYMENT_TOTAL_CURRENT	0.00	0
AMT_RECIVABLE	0.00	0
AMT_RECEIVABLE_PRINCIPAL	0.00	0
SK_ID_CURR	0.00	0
AMT_DRAWINGS_CURRENT	0.00	0
AMT_CREDIT_LIMIT_ACTUAL	0.00	0

In [ ]:

```
# Visualization of these Misssing Values
fig, ax = plt.subplots(1,1, sharex=False, figsize=(20,10))
ax = msno.bar(df_credit_card_bal.sample(10000))
ax.set_title("Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more missi
```

Out[ ]:

```
Text(0.5, 1.0, 'Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more mis
sing values from sample]')
```



DISCUSSION </br> From these visualizations and numerical based analysis, we can see that there is a high concentration of missing values around the AMT_CURRENT Features. This is definitly and insight into the context of the data that may be helpful.

VEDA: Input Feature Visualiaztion (previous_application.csv)

VEDA: Correlation Analysis: previous_application.csv

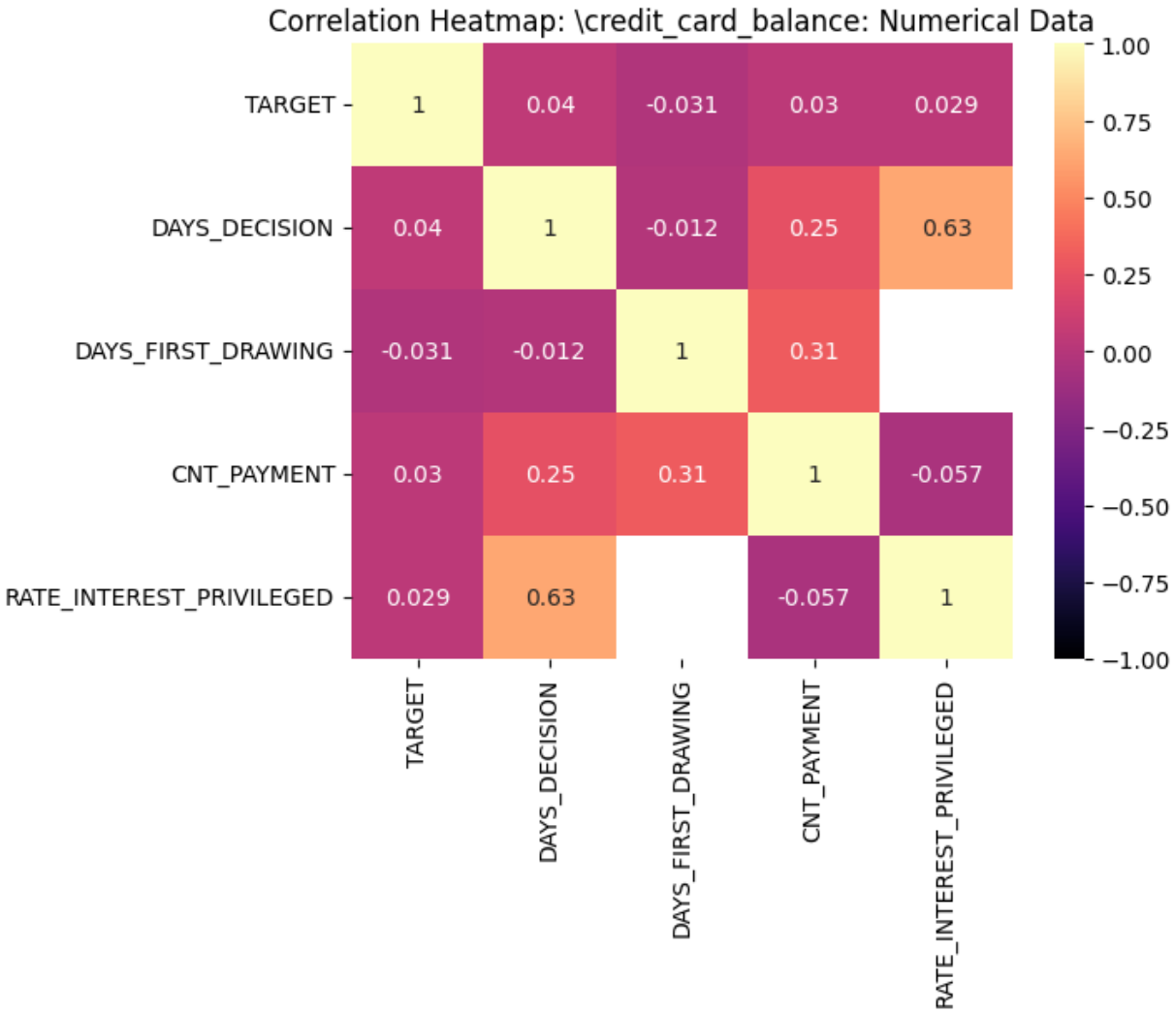
In [ ]:
```
import pandas as pd

pre_app_target_merge = pd.merge(df_pre_app, df_app_train[['SK_ID_CURR', 'TARGET']], o
pre_app_corr = pre_app_target_merge.corr()['TARGET']
pre_app_corr_sorted = pre_app_corr.abs().sort_values(ascending=False)

## Show the top correlated
pre_app_corr_sorted.head(10)

## select the top 5 correlated features including the target
n=4
pre_app_top_feat_list = pre_app_corr_sorted[0:n+1].index.tolist()

## Lets put these features into a dataframe with thier original values with the targe
pre_app_top_feat = pre_app_target_merge[pre_app_top_feat_list]
```

```
<ipython-input-317-beabb716ec20>:4: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  pre_app_corr = pre_app_target_merge.corr()['TARGET']
```

In [ ]:
```
corr_data = pre_app_top_feat
corr_data = corr_data.corr()
```

```
sns.heatmap(corr_data, cmap="magma", annot=True, vmin= -1.0, vmax=1.0)
plt.title("Correlation Heatmap: \credit_card_balance: Numerical Data")
```

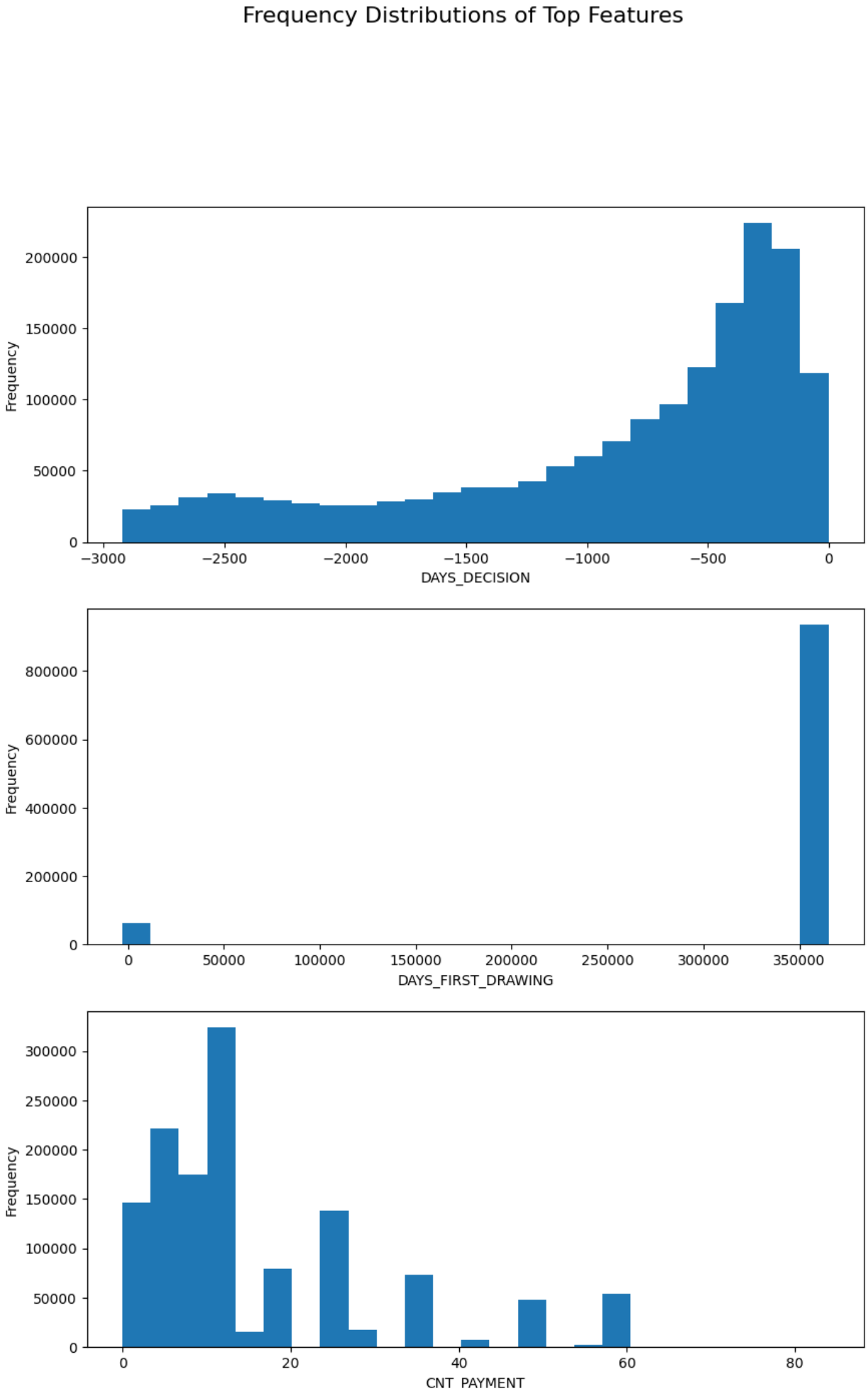
Out[ ]:
```
Text(0.5, 1.0, 'Correlation Heatmap: \\credit_card_balance: Numerical Data')
```



DISCUSSION </br> From this heat map of the correlations we can see that the highest correlation is between the DAYS_DECISION and DAYS_FIRST_DRAWING. Overall this data set seems to be more consistently positivly correlated with the target value being equal to 1 than most of the others.

VEDA: Distribution Analysis: previous_application.csv

In [ ]:
```
# Lets look at the distributions of these data
fig, axs = plt.subplots(nrows=n, figsize=(10,20))
fig.suptitle('Frequency Distributions of Top Features', fontsize=16)
for i, feature in enumerate(pre_app_top_feat_list):
  axs[i - 1].hist(pre_app_top_feat[feature], bins=25)
  axs[i - 1].set_xlabel(feature)
  axs[i - 1].set_ylabel("Frequency")

plt.show()
```


Frequency Distributions of Top Features
Frequency
DAYS_DECISION
DAYS_FIRST_DRAWING
CNT_PAYMENT

DISCUSSION </br> We can see from the distributions that there seems to be an interesting distribution with DAYS_DECISION and CNT_PAYMENT, however the other features seem to be categorical in nature.

VEDA: Missing Data Analysis: previous_application.csv

In [ ]:

```
# Numerical Analysis

# CITATION: Parts of code taken from HCDR_baseline_submission_phase2 starter code

missing_percentage = (df_pre_app.isnull().sum() / df_pre_app.isnull().count() * 100).
missing_count = df_pre_app.isna().sum().sort_values(ascending = False)
missing_app_table = pd.concat([missing_percentage, missing_count], axis=1, keys=["Mis
missing_app_table.head(20)
```

Out[ ]:

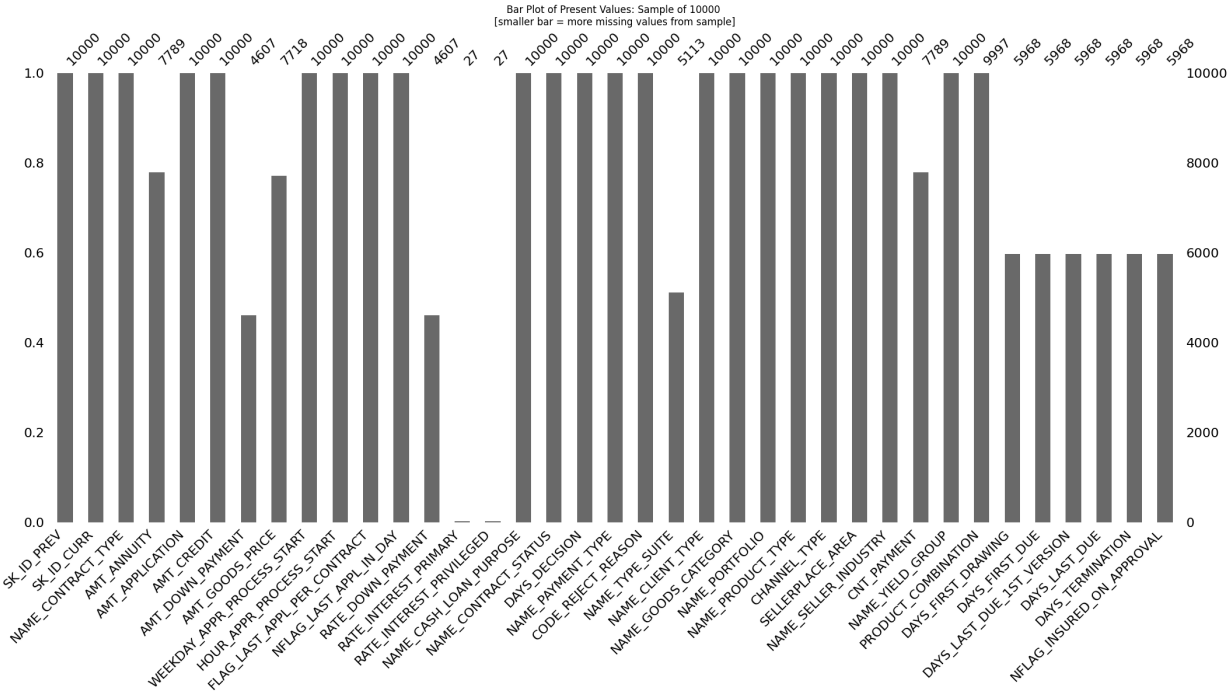
	Missing (%)	Missing (Count)
RATE_INTEREST_PRIVILEGED	99.64	1664263
RATE_INTEREST_PRIMARY	99.64	1664263
AMT_DOWN_PAYMENT	53.64	895844
RATE_DOWN_PAYMENT	53.64	895844
NAME_TYPE_SUITE	49.12	820405
NFLAG_INSURED_ON_APPROVAL	40.30	673065
DAYS_TERMINATION	40.30	673065
DAYS_LAST_DUE	40.30	673065
DAYS_LAST_DUE_1ST_VERSION	40.30	673065
DAYS_FIRST_DUE	40.30	673065
DAYS_FIRST_DRAWING	40.30	673065
AMT_GOODS_PRICE	23.08	385515
AMT_ANNUITY	22.29	372235
CNT_PAYMENT	22.29	372230
PRODUCT_COMBINATION	0.02	346
AMT_CREDIT	0.00	1
NAME_YIELD_GROUP	0.00	0
NAME_PORTFOLIO	0.00	0
NAME_SELLER_INDUSTRY	0.00	0
SELLERPLACE_AREA	0.00	0

In [ ]:

```
# Visualization of these Misssing Values
fig, ax = plt.subplots(1,1, sharex=False, figsize=(20,10))
ax = msno.bar(df_pre_app.sample(10000))
ax.set_title("Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more missi
```

Out[ ]:

```
Text(0.5, 1.0, 'Bar Plot of Present Values: Sample of 10000 \n[smaller bar = more mis
sing values from sample]')
```



DISCUSSION </br> From these figures it is quite obvious that both RATE_INTEREST_PRIMAR and RATE_INTEREST_PRIVLEGED are outliers in the amoun of data that is missing. This could aid in our feature selection later in the project.

Preprocessing

Preprocessing: Feature Selection

Preprocessing: Feature Selection - application_train.csv

In [ ]:
```
import pandas as pd
```

In [ ]:
```
## Finding Feature names of categorical and numerical data

# string columns
obj_cols = df_app_train.select_dtypes(include='object').columns.tolist()

# numerical categorical columns
num_cat_cols = df_app_train.select_dtypes(include=['number']).loc[:, df_app_train.nun

# numerical columns
num_cols = df_app_train.select_dtypes(include=['number']).loc[:, df_app_train.nunique

## Add ID and target

if "TARGET" not in obj_cols:
  obj_cols.append("TARGET")
```

```
if "SK_ID_CURR" not in obj_cols:
  obj_cols.append("SK_ID_CURR")

if "TARGET" not in num_cat_cols:
  num_cat_cols.append("TARGET")

if "SK_ID_CURR" not in num_cat_cols:
  num_cat_cols.append("SK_ID_CURR")

if "TARGET" not in num_cols:
  num_cols.append("TARGET")

if "SK_ID_CURR" not in num_cols:
  num_cols.append("SK_ID_CURR")
```

In [ ]:

```
df_app_obj = df_app_train[obj_cols]
df_app_num_cat = df_app_train[num_cat_cols]
df_app_num = df_app_train[num_cols]
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# create a label encoder object
le = LabelEncoder()

exclude = ["TARGET", "SK_ID_CURR"]
# encode the values in column 'A'
df_app_obj = df_app_obj.apply(lambda x: x if x.name in exclude else le.fit_transform(
df_app_obj = df_app_obj.drop("TARGET", axis=1)
df_app_num_cat = df_app_num_cat.drop("TARGET", axis=1)
# pos_cash_target_merge = pd.merge(df_pos_cash_bal, df_app_train[['SK_ID_CURR', 'TARG
# join the df_app_obj with the numerical cat
df_num_cat = pd.merge(df_app_obj, df_app_num_cat, on="SK_ID_CURR")
df_num_cat = pd.merge(df_num_cat, df_app_train[["SK_ID_CURR","TARGET"]], on="SK_ID_CU
```

In [ ]:

```
# numerical feature selection

df_app_num["DAYS_BIRTH"] = abs(df_app_num["DAYS_BIRTH"])
app_num_corr = df_app_num.corr()['TARGET']
app_num_corr_sorted = app_num_corr.abs().sort_values(ascending=False)

## select the top 5 correlated features including the target
n=3
app_num_feat_list = app_num_corr_sorted[0:n+1].index.tolist()
app_num_feat_list = ["SK_ID_CURR"] + app_num_feat_list
print(app_num_feat_list)
## Lets put these features into a dataframe with thier original values with the targe
df_app_num_top_feat = df_app_train[app_num_feat_list]
```

```
<ipython-input-96-1be970523d2c>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df_app_num["DAYS_BIRTH"] = abs(df_app_num["DAYS_BIRTH"])
```

```
['SK_ID_CURR', 'TARGET', 'EXT_SOURCE_3', 'EXT_SOURCE_2', 'EXT_SOURCE_1']
```

In [ ]:

```
# categorical feature selection
app_cat_corr = df_num_cat.corr()['TARGET']
app_num_cat_corr_sorted = app_cat_corr.abs().sort_values(ascending=False)

## select the top 5 correlated features including the target
n=1
app_num_cat_feat_list = app_num_cat_corr_sorted[1:n+1].index.tolist()
app_num_cat_feat_list = ["SK_ID_CURR"] + app_num_cat_feat_list
## Lets put these features into a dataframe with thier original values with the targe
df_app_cat_top_feat = df_num_cat[app_num_cat_feat_list]
```

In [ ]:

```
# final features
## List of features
#print(app_num_feat_list)
#print(app_num_cat_feat_list)


if "TARGET" in df_app_cat_top_feat:
  df_app_cat_top_feat.drop("TARGET", axis=1)

if "TARGET" in df_app_num_top_feat:
  df_app_num_top_feat.drop("TARGET", axis=1)

#print(app_train_best_feat_list)
```

In [ ]:

```
# Final Table (Merge)
df_final_app_train = pd.merge(df_app_cat_top_feat, df_app_num_top_feat, on="SK_ID_CUR
df_final_app_train.head(10)
```

Out[ ]:

	SK_ID_CURR	REGION_RATING_CLIENT_W_CITY	TARGET	EXT_SOURCE_3	EXT_SOURCE_2	EXT_SOURC
0	100002	2	1	0.139376	0.262949	0.083
1	100003	1	0	NaN	0.622246	0.311
2	100004	2	0	0.729567	0.555912	I
3	100006	2	0	NaN	0.650442	I
4	100007	2	0	NaN	0.322738	I
5	100008	2	0	0.621226	0.354225	I
6	100009	2	0	0.492060	0.724000	0.774
7	100010	3	0	0.540654	0.714279	I
8	100011	2	0	0.751724	0.205747	0.587
9	100012	2	0	NaN	0.746644	I

Preprocessing: Feature Selection - bureau.csv

In [ ]:
```
## Finding Feature names of categorical and numerical data

# string columns
obj_cols = df_bureau.select_dtypes(include='object').columns.tolist()

# numerical categorical columns
num_cat_cols = df_bureau.select_dtypes(include=['number']).loc[:, df_bureau.nunique()

# numerical columns
num_cols = df_bureau.select_dtypes(include=['number']).loc[:, df_bureau.nunique() >=

## Add ID and target

if "SK_ID_CURR" not in obj_cols:
  obj_cols.append("SK_ID_CURR")

if "SK_ID_CURR" not in num_cat_cols:
  num_cat_cols.append("SK_ID_CURR")

if "SK_ID_CURR" not in num_cols:
  num_cols.append("SK_ID_CURR")
```

In [ ]:
```
df_bur_obj = df_bureau[obj_cols]
df_bur_num_cat = df_bureau[num_cat_cols]
df_bur_num = df_bureau[num_cols]
```

In [ ]:
```
# add the targets
# pos_cash_target_merge = pd.merge(df_pos_cash_bal, df_app_train[['SK_ID_CURR', 'TARG
df_bur_obj = pd.merge(df_bur_obj, df_app_train[['SK_ID_CURR', 'TARGET']], on='SK_ID_C
df_bur_num_cat = pd.merge(df_bur_num_cat, df_app_train[['SK_ID_CURR', 'TARGET']], on=
df_bur_num = pd.merge(df_bur_num, df_app_train[['SK_ID_CURR', 'TARGET']], on='SK_ID_C
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# create a label encoder object
le = LabelEncoder()

exclude = ["TARGET", "SK_ID_CURR"]
# encode the values in column 'A'
df_bur_obj = df_bur_obj.apply(lambda x: x if x.name in exclude else le.fit_transform(
df_bur_obj = df_bur_obj.drop("TARGET", axis=1)
df_bur_num_cat = df_bur_num_cat.drop("TARGET", axis=1)
# pos_cash_target_merge = pd.merge(df_pos_cash_bal, df_app_train[['SK_ID_CURR', 'TARG
# join the df_app_obj with the numerical cat
df_num_cat = pd.merge(df_bur_obj, df_bur_num_cat, on="SK_ID_CURR")
df_num_cat = pd.merge(df_num_cat, df_app_train[["SK_ID_CURR","TARGET"]], on="SK_ID_CU
```

In [ ]:
```
# numerical feature selection

bur_num_corr = df_bur_num.corr()['TARGET']
bur_num_corr_sorted = bur_num_corr.abs().sort_values(ascending=False)
```

```
## select the top 5 correlated features including the target

n=1
bur_num_feat_list = bur_num_corr_sorted[0:n+1].index.tolist()
if "SK_ID_CURR" in bur_num_feat_list:
  bur_num_feat_list.remove("SK_ID_CURR")
bur_num_feat_list = ["SK_ID_CURR"] + bur_num_feat_list
print(bur_num_feat_list)
if "TARGET" in bur_num_feat_list:
  bur_num_feat_list.remove("TARGET")
## Lets put these features into a dataframe with thier original values with the targe
df_bur_num = df_bureau[bur_num_feat_list]
```

```
['SK_ID_CURR', 'TARGET', 'DAYS_CREDIT']
```

In [ ]:

```
# categorical feature selection
bur_cat_corr = df_num_cat.corr()['TARGET']
bur_num_cat_corr_sorted = bur_cat_corr.abs().sort_values(ascending=False)

## select the top 5 correlated features including the target
n=1
bur_num_cat_feat_list = bur_num_cat_corr_sorted[1:n+1].index.tolist()
bur_num_cat_feat_list = ["SK_ID_CURR"] + bur_num_cat_feat_list
## Lets put these features into a dataframe with thier original values with the targe
df_bur_cat_top_feat = df_num_cat[bur_num_cat_feat_list]
```

In [ ]:

```
df_bur_cat_top_feat.head(10)
```

Out[ ]:

	SK_ID_CURR	CREDIT_ACTIVE
0	215354	2
1	215354	2
2	215354	2
3	215354	2
4	215354	2
5	215354	2
6	215354	2
7	215354	2
8	215354	2
9	215354	2

In [ ]:

```
# final features
## List of features
#print(bur_num_feat_list)
#print(bur_num_cat_feat_list)


if "TARGET" in df_bur_cat_top_feat:
  df_bur_cat_top_feat.drop("TARGET", axis=1)
```

```
if "TARGET" in df_bur_num:
  df_bur_num.drop("TARGET", axis=1)
```

In [ ]:
```
df_bur_cat_top_feat_un = df_bureau[bur_num_cat_feat_list]
df_final_bur_unen = pd.merge(df_bur_cat_top_feat_un, df_bur_num, on="SK_ID_CURR")
```

In [ ]:
```
df_final_bur_unen.head(10)
```

Out[ ]:

	SK_ID_CURR	CREDIT_ACTIVE	DAYS_CREDIT
0	215354	Closed	-497
1	215354	Closed	-208
2	215354	Closed	-203
3	215354	Closed	-203
4	215354	Closed	-629
5	215354	Closed	-273
6	215354	Closed	-43
7	215354	Closed	-1872
8	215354	Closed	-1734
9	215354	Closed	-1333

In [ ]:
```
df_final_bureau = pd.merge(df_bur_cat_top_feat, df_bur_num, on="SK_ID_CURR")
df_final_bureau.head(10)
```

Out[ ]:

	SK_ID_CURR	CREDIT_ACTIVE	DAYS_CREDIT
0	215354	2	-497
1	215354	2	-208
2	215354	2	-203
3	215354	2	-203
4	215354	2	-629
5	215354	2	-273
6	215354	2	-43
7	215354	2	-1872
8	215354	2	-1734
9	215354	2	-1333

Preprocessing: Feature Selection - bureau_balance.csv

From the previous round of EDA we know that the only feature that is worth keeping is the MONTHS_BALANCE

In [ ]:
```
df_final_bureau_balance = df_bureau_bal[["SK_ID_BUREAU", "MONTHS_BALANCE"]]
```

In [ ]:
```
df_final_bureau_balance.head(10)
```

Out[ ]:

	SK_ID_BUREAU	MONTHS_BALANCE
0	5715448	0
1	5715448	-1
2	5715448	-2
3	5715448	-3
4	5715448	-4
5	5715448	-5
6	5715448	-6
7	5715448	-7
8	5715448	-8
9	5715448	-9

In [ ]:
```
# Lets add the SK_ID_CURR to this table for the final merge

df_final_bureau_balance = pd.merge(df_final_bureau_balance, df_bureau[['SK_ID_BUREAU'
```

In [ ]:
```
df_final_bureau_balance.head(10)
```

Out[ ]:

	SK_ID_BUREAU	MONTHS_BALANCE	SK_ID_CURR
0	5715448	0	380361.0
1	5715448	-1	380361.0
2	5715448	-2	380361.0
3	5715448	-3	380361.0
4	5715448	-4	380361.0
5	5715448	-5	380361.0
6	5715448	-6	380361.0
7	5715448	-7	380361.0
8	5715448	-8	380361.0
9	5715448	-9	380361.0

In [ ]:
```
df_final_bureau_balance = df_final_bureau_balance.drop("SK_ID_BUREAU", axis=1)
```

In [ ]:
```
df_final_bureau_balance.head()
```

Out[ ]:

	MONTHS_BALANCE	SK_ID_CURR
0	0	380361.0
1	-1	380361.0
2	-2	380361.0
3	-3	380361.0
4	-4	380361.0

Preprocessing: Feature Selection - POS_CASH_balance.csv

In [ ]:
```
## Finding Feature names of categorical and numerical data

# numerical columns
num_cols = df_pos_cash_bal.select_dtypes(include=['number']).loc[:, df_pos_cash_bal.n

## Add ID and target

if "SK_ID_CURR" not in num_cols:
  num_cols.append("SK_ID_CURR")
```

In [ ]:
```
df_pos_num = df_pos_cash_bal[num_cols]
```

In [ ]:
```
df_pos_num.head(10)
```

Out[ ]:

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT_FUTURE	SK_D
0	1803195	182943	-31	48.0	45.0	
1	1715348	367990	-33	36.0	35.0	
2	1784872	397406	-32	12.0	9.0	
3	1903291	269225	-35	48.0	42.0	
4	2341044	334279	-35	36.0	35.0	
5	2207092	342166	-32	12.0	12.0	
6	1110516	204376	-38	48.0	43.0	
7	1387235	153211	-35	36.0	36.0	
8	1220500	112740	-31	12.0	12.0	
9	2371489	274851	-32	24.0	16.0	

NOTE </br> Since there are no numerical categorical features in this data set and only one categorical feature we will ignore the categorical feature. This is because in EDA it did not result in any high correlation, so we will only run the correlation analysis on the numerical features for the best result.

In [ ]:
```
df_pos_num = pd.merge(df_pos_num, df_app_train[['SK_ID_CURR', 'TARGET']], on='SK_ID_C
```

In [ ]:
```
# numerical feature selection

pos_num_corr = df_pos_num.corr()['TARGET']
pos_num_corr_sorted = pos_num_corr.abs().sort_values(ascending=False)

## select the top 5 correlated features including the target
n=1
pos_num_feat_list = pos_num_corr_sorted[0:n+1].index.tolist()
pos_num_feat_list = ["SK_ID_CURR"] + pos_num_feat_list
if "TARGET" in pos_num_feat_list:
  pos_num_feat_list.remove("TARGET")
print(pos_num_feat_list)
## Lets put these features into a dataframe with thier original values with the targe
df_pos_num_top_feat = df_pos_cash_bal[pos_num_feat_list]
```

```
['SK_ID_CURR', 'CNT_INSTALMENT_FUTURE']
```

In [ ]:
```
df_final_pos_cash_bal = df_pos_num_top_feat
```

In [ ]:
```
df_final_pos_cash_bal.head(10)
```

Out[ ]:

	SK_ID_CURR	CNT_INSTALMENT_FUTURE
0	182943	45.0
1	367990	35.0
2	397406	9.0
3	269225	42.0
4	334279	35.0
5	342166	12.0
6	204376	43.0
7	153211	36.0
8	112740	12.0
9	274851	16.0

Preprocessing: Feature Selection - credit_card_balance.csv

In [ ]:

```
## Finding Feature names of categorical and numerical data

# string columns
obj_cols = df_credit_card_bal.select_dtypes(include='object').columns.tolist()

# numerical categorical columns
num_cat_cols = df_credit_card_bal.select_dtypes(include=['number']).loc[:, df_credit_

# numerical columns
num_cols = df_credit_card_bal.select_dtypes(include=['number']).loc[:, df_credit_card

## Add ID and target


if "SK_ID_CURR" not in obj_cols:
  obj_cols.append("SK_ID_CURR")


if "SK_ID_CURR" not in num_cat_cols:
  num_cat_cols.append("SK_ID_CURR")

if "SK_ID_CURR" not in num_cols:
  num_cols.append("SK_ID_CURR")
```

In [ ]:

```
print(obj_cols)
print(num_cat_cols)
print(num_cols)
```

```
['NAME_CONTRACT_STATUS', 'SK_ID_CURR']
['SK_ID_CURR']
['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'AMT_BALANCE', 'AMT_CREDIT_LIMIT_ACTUA
L', 'AMT_DRAWINGS_ATM_CURRENT', 'AMT_DRAWINGS_CURRENT', 'AMT_DRAWINGS_OTHER_CURRENT',
'AMT_DRAWINGS_POS_CURRENT', 'AMT_INST_MIN_REGULARITY', 'AMT_PAYMENT_CURRENT', 'AMT_PA
YMENT_TOTAL_CURRENT', 'AMT_RECEIVABLE_PRINCIPAL', 'AMT_RECIVABLE', 'AMT_TOTAL_RECEIVA
BLE', 'CNT_DRAWINGS_ATM_CURRENT', 'CNT_DRAWINGS_CURRENT', 'CNT_DRAWINGS_OTHER_CURREN
T', 'CNT_DRAWINGS_POS_CURRENT', 'CNT_INSTALMENT_MATURE_CUM', 'SK_DPD', 'SK_DPD_DEF']
```

NOTE We can see from this information that there are no numerical catigorical or catigorical data features of significiance. The only categorical column being NAME_CONTRACT_STATUS, which we know from our EDA sept to have 0.002 correlation with the target value, will be omitted

In [ ]:
```
df_credit_num = df_credit_card_bal[num_cols]
```

In [ ]:
```
df_credit_num = pd.merge(df_credit_num, df_app_train[['SK_ID_CURR', 'TARGET']], on='S
```

In [ ]:
```
# numerical feature selection

credit_num_corr = df_credit_num.corr()['TARGET']
credit_num_corr_sorted = credit_num_corr.abs().sort_values(ascending=False)

## select the top 3 correlated features including the target
n=1
credit_num_feat_list = credit_num_corr_sorted[0:n+1].index.tolist()
credit_num_feat_list = ["SK_ID_CURR"] + credit_num_feat_list
if "TARGET" in credit_num_feat_list:
  credit_num_feat_list.remove("TARGET")
print(credit_num_feat_list)
## Lets put these features into a dataframe with thier original values with the targe
df_credit_num_top_feat = df_credit_card_bal[credit_num_feat_list]
```

```
['SK_ID_CURR', 'AMT_BALANCE']
```

In [ ]:
```
df_credit_num_top_feat.head(10)
```

Out[ ]:

	SK_ID_CURR	AMT_BALANCE
0	378907	56.970
1	363914	63975.555
2	371185	31815.225
3	337855	236572.110
4	126868	453919.455
5	380010	82903.815
6	171320	353451.645
7	118650	47962.125
8	367360	291543.075
9	203885	201261.195

In [ ]:
```
df_final_credit = df_credit_num_top_feat
```

Preprocessing: Feature Selection - installments_payments.csv

In [ ]:
```
## Finding Feature names of categorical and numerical data

# string columns
obj_cols = df_installments_payments.select_dtypes(include='object').columns.tolist()

# numerical categorical columns
num_cat_cols = df_installments_payments.select_dtypes(include=['number']).loc[:, df_i

# numerical columns
num_cols = df_installments_payments.select_dtypes(include=['number']).loc[:, df_insta

## Add ID

if "SK_ID_CURR" not in obj_cols:
  obj_cols.append("SK_ID_CURR")

if "SK_ID_CURR" not in num_cat_cols:
  num_cat_cols.append("SK_ID_CURR")

if "SK_ID_CURR" not in num_cols:
  num_cols.append("SK_ID_CURR")
```

In [ ]:
```
print(obj_cols)
print(num_cat_cols)
print(num_cols)
```

```
['SK_ID_CURR']
['SK_ID_CURR']
['SK_ID_PREV', 'SK_ID_CURR', 'NUM_INSTALMENT_VERSION', 'NUM_INSTALMENT_NUMBER', 'DAYS
_INSTALMENT', 'DAYS_ENTRY_PAYMENT', 'AMT_INSTALMENT', 'AMT_PAYMENT']
```

NOTE </br> Given That there are no categorical or numerically categorical features we will ignore this part of the process.

In [ ]:
```
df_install_num = df_installments_payments[num_cols]
```

In [ ]:
```
df_install_num = pd.merge(df_install_num, df_app_train[['SK_ID_CURR', 'TARGET']], on=
```

In [ ]:
```
# numerical feature selection

install_num_corr = df_install_num.corr()['TARGET']
install_num_corr_sorted = install_num_corr.abs().sort_values(ascending=False)

## select the top 2 correlated features including the target
n=1
install_num_feat_list = install_num_corr_sorted[0:n+1].index.tolist()
install_num_feat_list = ["SK_ID_CURR"] + install_num_feat_list
if "TARGET" in install_num_feat_list:
  install_num_feat_list.remove("TARGET")
print(install_num_feat_list)

## Lets put these features into a dataframe with thier original values with the targe
df_install_num_top_feat = df_installments_payments[install_num_feat_list]
```

```
['SK_ID_CURR', 'DAYS_ENTRY_PAYMENT']
```

In [ ]:
```
df_final_installments_payments = df_install_num_top_feat
```

In [ ]:
```
df_final_installments_payments.head(10)
```

Out[ ]:

	SK_ID_CURR	DAYS_ENTRY_PAYMENT
0	161674	-1187.0
1	151639	-2156.0
2	193053	-63.0
3	199697	-2426.0
4	167756	-1366.0
5	164489	-1417.0
6	184693	-352.0
7	111420	-994.0
8	112102	-197.0
9	109741	-609.0

Preprocessing: Feature Selection - previous_application.csv

In [ ]:
```
## Finding Feature names of categorical and numerical data

# string columns
obj_cols = df_pre_app.select_dtypes(include='object').columns.tolist()

# numerical categorical columns
num_cat_cols = df_pre_app.select_dtypes(include=['number']).loc[:, df_pre_app.nunique

# numerical columns
num_cols = df_pre_app.select_dtypes(include=['number']).loc[:, df_pre_app.nunique() >

## Add ID

if "SK_ID_CURR" not in obj_cols:
  obj_cols.append("SK_ID_CURR")

if "SK_ID_CURR" not in num_cat_cols:
  num_cat_cols.append("SK_ID_CURR")

if "SK_ID_CURR" not in num_cols:
  num_cols.append("SK_ID_CURR")
```

In [ ]:
```
print(obj_cols)
print(num_cat_cols)
print(num_cols)
```

```
['NAME_CONTRACT_TYPE', 'WEEKDAY_APPR_PROCESS_START', 'FLAG_LAST_APPL_PER_CONTRACT',
'NAME_CASH_LOAN_PURPOSE', 'NAME_CONTRACT_STATUS', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_R
EASON', 'NAME_TYPE_SUITE', 'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY', 'NAME_PORTFOLI
O', 'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE', 'NAME_SELLER_INDUSTRY', 'NAME_YIELD_GROUP',
'PRODUCT_COMBINATION', 'SK_ID_CURR']
['NFLAG_LAST_APPL_IN_DAY', 'NFLAG_INSURED_ON_APPROVAL', 'SK_ID_CURR']
['SK_ID_PREV', 'SK_ID_CURR', 'AMT_ANNUITY', 'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_DOW
N_PAYMENT', 'AMT_GOODS_PRICE', 'HOUR_APPR_PROCESS_START', 'RATE_DOWN_PAYMENT', 'RATE_
INTEREST_PRIMARY', 'RATE_INTEREST_PRIVILEGED', 'DAYS_DECISION', 'SELLERPLACE_AREA',
'CNT_PAYMENT', 'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION',
'DAYS_LAST_DUE', 'DAYS_TERMINATION']
```

In [ ]:
```
df_pre_obj = df_pre_app[obj_cols]
df_pre_num_cat = df_pre_app[num_cat_cols]
df_pre_num = df_pre_app[num_cols]
```

In [ ]:
```
# add targets
df_pre_obj = pd.merge(df_pre_obj, df_app_train[['SK_ID_CURR', 'TARGET']], on='SK_ID_C
df_pre_num_cat = pd.merge(df_pre_num_cat, df_app_train[['SK_ID_CURR', 'TARGET']], on=
df_pre_num = pd.merge(df_pre_num, df_app_train[['SK_ID_CURR', 'TARGET']], on='SK_ID_C
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
import pandas as pd
```

```
# create a label encoder object
le = LabelEncoder()

exclude = ["TARGET", "SK_ID_CURR"]
# encode the values in column 'A'
df_pre_obj = df_pre_obj.apply(lambda x: x if x.name in exclude else le.fit_transform(
df_pre_obj = df_pre_obj.drop("TARGET", axis=1)
df_pre_num_cat = df_pre_num_cat.drop("TARGET", axis=1)

# join the df_pre_obj with the numerical cat
df_num_cat = pd.merge(df_pre_obj, df_pre_num_cat, on="SK_ID_CURR")
df_num_cat = pd.merge(df_num_cat, df_app_train[["SK_ID_CURR","TARGET"]], on="SK_ID_CU
```

In [ ]:

```
# numerical feature selection

pre_num_corr = df_pre_num.corr()['TARGET']
pre_num_corr_sorted = pre_num_corr.abs().sort_values(ascending=False)

## select the top 10 correlated features including the target

n=1
pre_num_feat_list = pre_num_corr_sorted[0:n+1].index.tolist()
if "SK_ID_CURR" in pre_num_feat_list:
  pre_num_feat_list.remove("SK_ID_CURR")
pre_num_feat_list = ["SK_ID_CURR"] + pre_num_feat_list
if "TARGET" in pre_num_feat_list:
  pre_num_feat_list.remove("TARGET")
## Lets put these features into a dataframe with thier original values with the targe
df_pre_num = df_pre_app[pre_num_feat_list]
print(pre_num_feat_list)
```

```
['SK_ID_CURR', 'DAYS_DECISION']
```

In [ ]:

```
df_pre_num.head(10)
```

Out[ ]:

	SK_ID_CURR	DAYS_DECISION
0	271877	-73
1	108129	-164
2	122040	-301
3	176158	-512
4	202054	-781
5	199383	-684
6	175704	-14
7	296299	-21
8	342292	-386
9	334349	-57

In [ ]:

```
df_pre_num
```

Out[ ]:

	SK_ID_CURR	DAYS_DECISION
0	271877	-73
1	108129	-164
2	122040	-301
3	176158	-512
4	202054	-781
...	...	...
1670209	352015	-544
1670210	334635	-1694
1670211	249544	-1488
1670212	400317	-1185
1670213	261212	-1193

1670214 rows × 2 columns

In [ ]:
```
# categorical feature selection
pre_cat_corr = df_num_cat.corr()['TARGET']
pre_num_cat_corr_sorted = pre_cat_corr.abs().sort_values(ascending=False)

## select the top 5 correlated features including the target
n=1
pre_num_cat_feat_list = pre_num_cat_corr_sorted[1:n+1].index.tolist()
pre_num_cat_feat_list = pre_num_cat_feat_list
## Lets put these features into a dataframe with thier original values with the targe
df_pre_cat_top_feat = df_num_cat[pre_num_cat_feat_list]
print(pre_num_cat_feat_list)
```

```
['NAME_CONTRACT_STATUS']
```

In [ ]:
```
df_pre_cat = df_pre_app[pre_num_cat_feat_list]
```

In [ ]:
```
df_pre_cat.head(10)
```

Out[ ]:

	NAME_CONTRACT_STATUS
0	Approved
1	Approved
2	Approved
3	Approved
4	Refused
5	Approved
6	Canceled
7	Canceled
8	Canceled
9	Canceled

In [ ]:

```
# final features
## List of features
#print(pre_num_feat_list)
#print(pre_num_cat_feat_list)


if "TARGET" in df_pre_cat_top_feat:
  df_pre_cat_top_feat.drop("TARGET", axis=1)

if "TARGET" in df_pre_num:
  df_pre_num.drop("TARGET", axis=1)
```

In [ ]:

```
if "SK_ID_CURR" in pre_num_feat_list:
  pre_num_feat_list.remove("SK_ID_CURR")

pre_feature_list = ["SK_ID_CURR"] + pre_num_cat_feat_list + pre_num_feat_list
print(pre_feature_list)
```

```
['SK_ID_CURR', 'NAME_CONTRACT_STATUS', 'DAYS_DECISION']
```

In [ ]:

```
# can not get final data frame with the current method due to ram issues, this will n
df_final_pre = df_pre_app.loc[:, pre_feature_list]
#df_final_pre = pd.merge(df_pre_cat_top_feat, df_pre_num, on="SK_ID_CURR")
df_final_pre.head(10)
```

Out[ ]:

	SK_ID_CURR	NAME_CONTRACT_STATUS	DAYS_DECISION
0	271877	Approved	-73
1	108129	Approved	-164
2	122040	Approved	-301
3	176158	Approved	-512
4	202054	Refused	-781
5	199383	Approved	-684
6	175704	Canceled	-14
7	296299	Canceled	-21
8	342292	Canceled	-386
9	334349	Canceled	-57

Preprocessing: Feature Selection - Merge

In [ ]:
```
# List of Final DFs
# app
df__1 = df_final_app_train

# bur
df__2 = df_final_bur_unen

# bur_bal
df__3 = df_final_bureau_balance

# pos cash
df__4 = df_final_pos_cash_bal

# credit
df__5 = df_final_credit

# install
df__6 = df_final_installments_payments

# pre
df__7 = df_final_pre
```

DISCLAIMER </br> Do to resource of computation we will only be taking the most important features, compared to our previous strategy of all that had a relivtly strong correlation.

In [ ]:
```
df__1 = df__1.drop("REGION_RATING_CLIENT_W_CITY", axis=1)
```

In [ ]:
```
df_concat = pd.merge(df__1, df__2, on=["SK_ID_CURR"], how='right')
```

In [ ]:
```
df_concat.shape
```

Out[ ]:
```
(15636260, 7)
```

In [ ]:
```
df_hcdr = df_concat
```

In [ ]:
```
df_hcdr.head(10)
```

Out[ ]:

	SK_ID_CURR	TARGET	EXT_SOURCE_3	EXT_SOURCE_2	EXT_SOURCE_1	CREDIT_ACTIVE	DAYS_CRED
0	215354	0.0	0.231439	0.522745	0.873736	Closed	-4
1	215354	0.0	0.231439	0.522745	0.873736	Closed	-2
2	215354	0.0	0.231439	0.522745	0.873736	Closed	-2
3	215354	0.0	0.231439	0.522745	0.873736	Closed	-2
4	215354	0.0	0.231439	0.522745	0.873736	Closed	-6
5	215354	0.0	0.231439	0.522745	0.873736	Closed	-2
6	215354	0.0	0.231439	0.522745	0.873736	Closed	-
7	215354	0.0	0.231439	0.522745	0.873736	Closed	-18
8	215354	0.0	0.231439	0.522745	0.873736	Closed	-17
9	215354	0.0	0.231439	0.522745	0.873736	Closed	-13

In [ ]:
```
# Numerical
hcdr_num = ["EXT_SOURCE_3", "EXT_SOURCE_2", "EXT_SOURCE_1", "DAYS_CREDT"]
# Categorical
hcdr_cat = ["CREDIT_ACTIVE"]
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline, FeatureUnion
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
```

In [ ]:
```

```

HCDR Preprocessing

In [ ]:
```
from sklearn.base import BaseEstimator, TransformerMixin
# Create a class to select numerical or categorical columns
# since Scikit-Learn doesn't handle DataFrames yet
class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values
```

In [ ]:
```
df_hcdr.head(10)
```

Out[ ]:

	SK_ID_CURR	TARGET	EXT_SOURCE_3	EXT_SOURCE_2	EXT_SOURCE_1	CREDIT_ACTIVE	DAYS_CRED
0	215354	0.0	0.231439	0.522745	0.873736	Closed	-49
1	215354	0.0	0.231439	0.522745	0.873736	Closed	-2(
2	215354	0.0	0.231439	0.522745	0.873736	Closed	-2(
3	215354	0.0	0.231439	0.522745	0.873736	Closed	-2(
4	215354	0.0	0.231439	0.522745	0.873736	Closed	-62
5	215354	0.0	0.231439	0.522745	0.873736	Closed	-27
6	215354	0.0	0.231439	0.522745	0.873736	Closed	-4
7	215354	0.0	0.231439	0.522745	0.873736	Closed	-187
8	215354	0.0	0.231439	0.522745	0.873736	Closed	-173
9	215354	0.0	0.231439	0.522745	0.873736	Closed	-133

In [ ]:
```
# Establish X and y
y = df_app_train['TARGET'].copy()
X = df_app_train.copy().drop(["TARGET"],axis=1)

# Split X & y into train & test sets
# Subsequently split train into train & validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_stat
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.2
X_kaggle_test = df_app_test

# Identify the numeric features we wish to consider.
num_attribs = X.select_dtypes(include = ['int64','float64']).columns

num_pipeline = Pipeline([
        ('selector', DataFrameSelector(num_attribs)),
        ('imputer', SimpleImputer(strategy='mean')),
        ('std_scaler', StandardScaler()),
    ])
# Identify the categorical features we wish to consider.
cat_attribs =  X.select_dtypes(include = ['object']).columns
```

```
# Notice handle_unknown="ignore" in OHE which ignore values from the validation/test
# do NOT occur in the training set
cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        #('imputer', SimpleImputer(strategy='most_frequent')),
        ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
        ('ohe', OneHotEncoder(sparse=False, handle_unknown="ignore"))
    ])

data_prep_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", num_pipeline),
        ("cat_pipeline", cat_pipeline),
    ])
```

Modeling Pipelines

Modeling Pipelines : Loss Functions

- L1 Loss (Mean Absolute Error):

$$L_1(x, y) = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i|$$

where: \ x and y are the predicted and actual values, respectively n is the number of samples in the dataset i is the index of each sample in the dataset </br> $|\cdot|$ denotes the absolute value \ The L1 loss function measures the absolute difference between the predicted values and actual values, and then takes the mean of those differences. It is less sensitive to outliers than the L2 loss function.

- L2 Loss (Mean Squared Error):

$$L_2(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2$$

where: \ x and y are the predicted and actual values, respectively n is the number of samples in the dataset i is the index of each sample in the dataset </br> This loss function is commonly used in regression problems, where the goal is to predict continuous values. It measures the average of the squared differences between the predicted and actual values. The L2 loss function measures the squared difference between the predicted values and actual values, and

> > > then takes the mean of those differences. It is more sensitive to outliers than the L1 loss function.

Modeling Pipelines : Metrics

- F1 Score:

> The F1 score is a metric that combines precision and recall. It is useful in situations where both precision and recall are important, such as in binary classification problems where the classes are imbalanced. The F1 score ranges from 0 to 1, where 1 represents perfect precision and recall. It is calculated as:

> > > $$F1 = 2\frac{precision*recall}{precision+recall}$$

- Accuracy Score: The accuracy score is a metric that measures the proportion of correctly classified samples out of all samples. It is useful when the classes in a dataset are balanced. However, it can be misleading in situations where the classes are imbalanced. The accuracy score ranges from 0 to 1, where 1 represents perfect classification. It is calculated as:

> > > $$accuracy = \frac{number\ of\ correctly\ classified\ samples}{total\ number\ of\ samples}$$

- AUC (Area Under the ROC Curve): The AUC is a metric that measures the performance of a binary classification model by calculating the area under the receiver operating characteristic (ROC) curve. The ROC curve is a graph that shows the true positive rate (sensitivity) against the false positive rate (1 - specificity) at different classification thresholds. The AUC ranges from 0 to 1, where 1 represents perfect classification. It is useful in situations where the classes are imbalanced and where the model's output is a probability. The AUC can be calculated using the trapezoidal rule or other numerical integration methods.

In summary, F1 score is useful in situations where both precision and recall are important, accuracy score is useful when the classes in a dataset are balanced, and AUC is useful in situations where the classes are imbalanced and where the model's output is a probability.

In [ ]:
```
try:
    expLog
except NameError:
    expLog = pd.DataFrame(columns=["exp_name",
                                   "Train Acc",
                                   "Valid Acc",
                                   "Test  Acc",
                                   "Train AUC",
                                   "Valid AUC",
                                   "Test  AUC"
                                  ])
```

Logistic Regression

In [ ]:
```
X_train.head(10)
```

Out[ ]:

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
35339	140933	Cash loans	F	Y	Y
82049	195150	Cash loans	F	Y	N
226288	362102	Cash loans	F	Y	Y
265467	407465	Cash loans	M	N	Y
175195	303015	Cash loans	F	Y	Y
92993	207984	Cash loans	M	N	Y
7206	108388	Cash loans	F	N	Y
164322	290486	Cash loans	F	N	Y
305651	454126	Cash loans	F	N	Y
137245	259172	Cash loans	F	N	N

10 rows × 121 columns

In [ ]:
```
y_train.head(10)
```

Out[ ]:
```
35339     0
82049     0
226288    0
265467    0
175195    0
92993     1
7206      0
164322    1
305651    0
137245    0
Name: TARGET, dtype: int64
```

In [ ]:
```
from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn import metrics

#Create the Logistic Regression Pipeline
lr_pipeline = Pipeline([
        ("preparation", data_prep_pipeline),
        ("linear", LogisticRegression())
    ])

#Fit the data to the pipeline
model = lr_pipeline.fit(X_train, y_train)

#Log the results of Accuracy and AUC for Train,Valid and Test datasets
exp_name = f"Baseline_Logistic_Regression"
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
               [accuracy_score(y_train, model.predict(X_train)),
                accuracy_score(y_valid, model.predict(X_valid)),
```

```
                accuracy_score(y_test, model.predict(X_test)),
                roc_auc_score(y_train, model.predict_proba(X_train)[:, 1]),
                roc_auc_score(y_valid, model.predict_proba(X_valid)[:, 1]),
                roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])],
    4))
expLog
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC
0	Baseline_Logistic_Regression	0.92	0.9162	0.9194	0.7485	0.7475	0.7438
1	Baseline_Logistic_Regression	0.92	0.9162	0.9194	0.7485	0.7475	0.7438

In [ ]:

```
#Create the AUC graph
#metrics.plot_roc_curve(lr_pipeline, X_valid, y_valid)
```

Random Forest

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier

#Create the Random Forest Pipeline
rf_pipeline = Pipeline([
        ("preparation", data_prep_pipeline),
        ("RandomForest",RandomForestClassifier(random_state=42))
        ])

#Fit the data to the pipeline
model = rf_pipeline.fit(X_train, y_train)

#Log the results of Accuracy and AUC for Train,Valid and Test datasets
exp_name = f"Baseline_Random_Forest"
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
               [accuracy_score(y_train, model.predict(X_train)),
                accuracy_score(y_valid, model.predict(X_valid)),
                accuracy_score(y_test, model.predict(X_test)),
                roc_auc_score(y_train, model.predict_proba(X_train)[:, 1]),
                roc_auc_score(y_valid, model.predict_proba(X_valid)[:, 1]),
                roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])],
    4))
expLog
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_encoders.py:868: Future
Warning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed i
n 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC
0	Baseline_Logistic_Regression	0.9200	0.9162	0.9194	0.7485	0.7475	0.7438
1	Baseline_Logistic_Regression	0.9200	0.9162	0.9194	0.7485	0.7475	0.7438
2	Baseline_Random_Forest	0.9999	0.9165	0.9194	1.0000	0.7102	0.7109

In [ ]:

```
#Create the AUC graph
#metrics.plot_roc_curve(rf_pipeline, X_valid, y_valid)
```

Decision Tree

In [ ]:

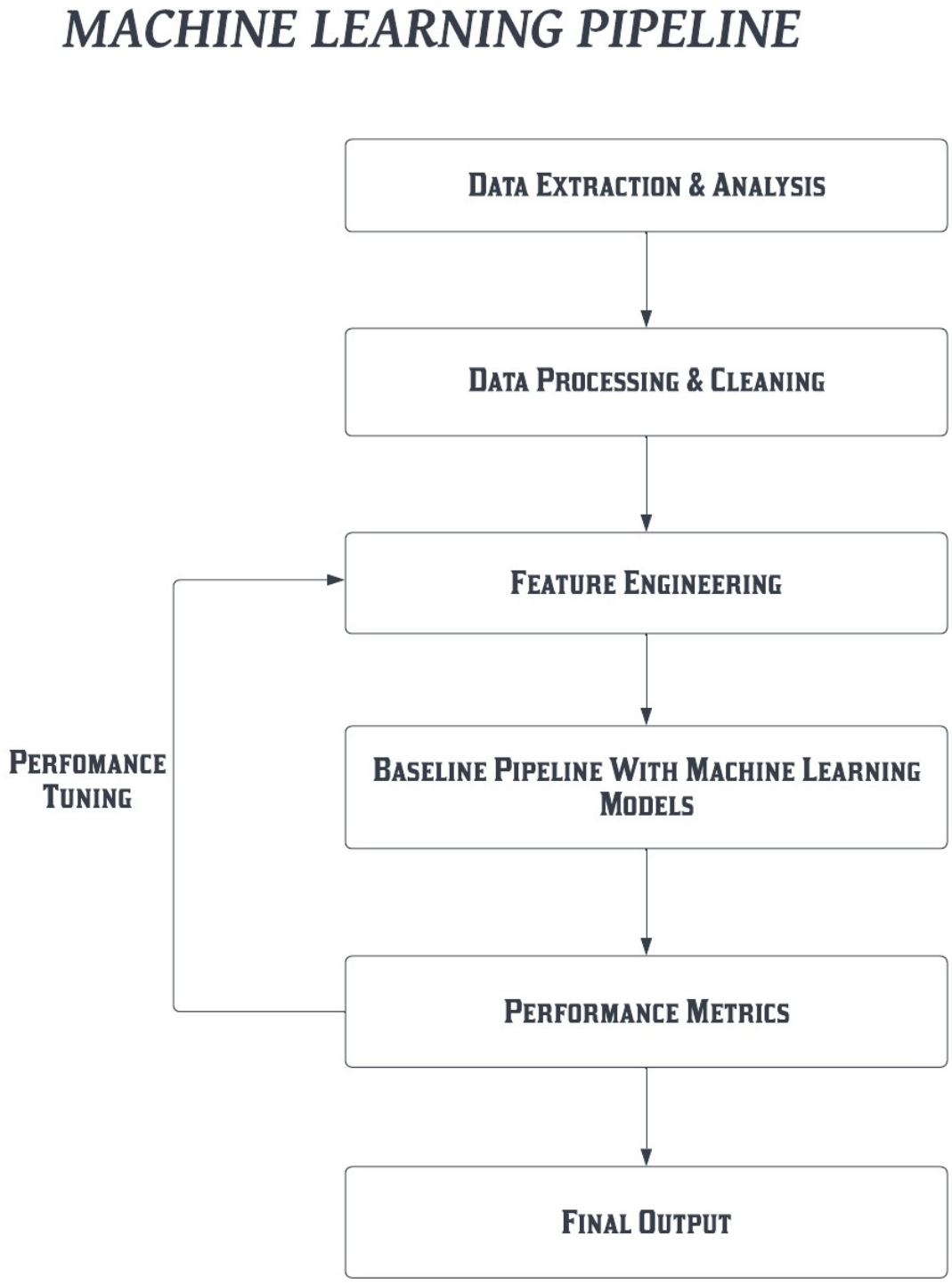
```
from sklearn.tree import DecisionTreeClassifier

#Create the Decision Tree Pipeline
dt_pipeline = Pipeline([
        ("preparation", data_prep_pipeline),
        ("RandomForest",DecisionTreeClassifier(random_state=42))
        ])

#Fit the data to the pipeline
model = dt_pipeline.fit(X_train, y_train)

#Log the results of Accuracy and AUC for Train,Valid and Test datasets
exp_name = f"Baseline_Decision_Tree"
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
               [accuracy_score(y_train, model.predict(X_train)),
                accuracy_score(y_valid, model.predict(X_valid)),
                accuracy_score(y_test, model.predict(X_test)),
                roc_auc_score(y_train, model.predict_proba(X_train)[:, 1]),
                roc_auc_score(y_valid, model.predict_proba(X_valid)[:, 1]),
                roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])],
    4))
expLog
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_encoders.py:868: Future
Warning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed i
n 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

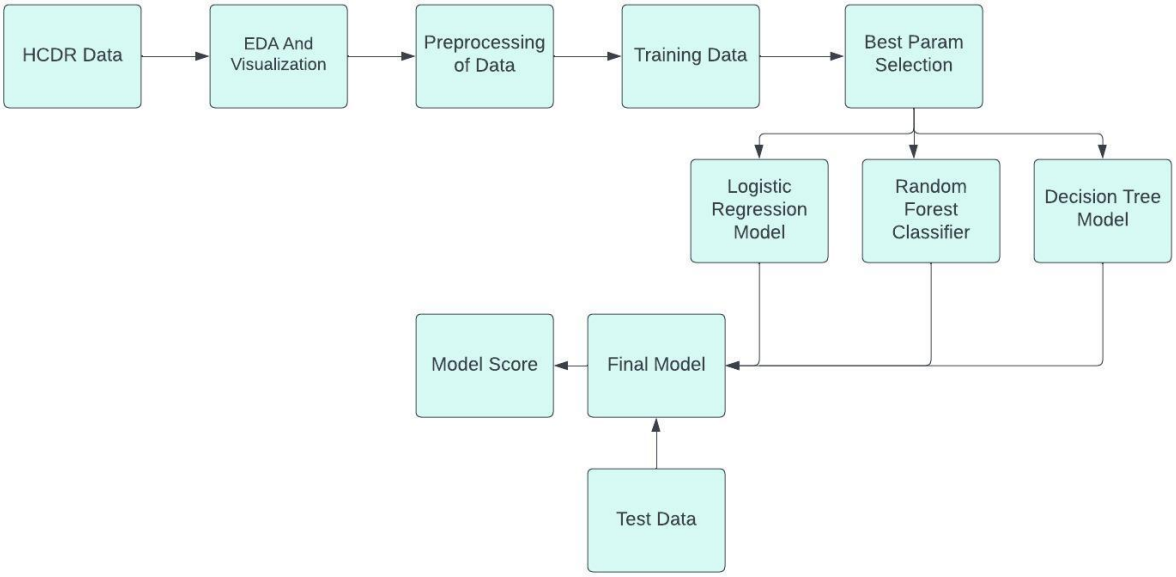
Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC
0	Baseline_Logistic_Regression	0.9200	0.9162	0.9194	0.7485	0.7475	0.7438
1	Baseline_Logistic_Regression	0.9200	0.9162	0.9194	0.7485	0.7475	0.7438
2	Baseline_Random_Forest	0.9999	0.9165	0.9194	1.0000	0.7102	0.7109
3	Baseline_Decision_Tree	1.0000	0.8528	0.8529	1.0000	0.5427	0.5367

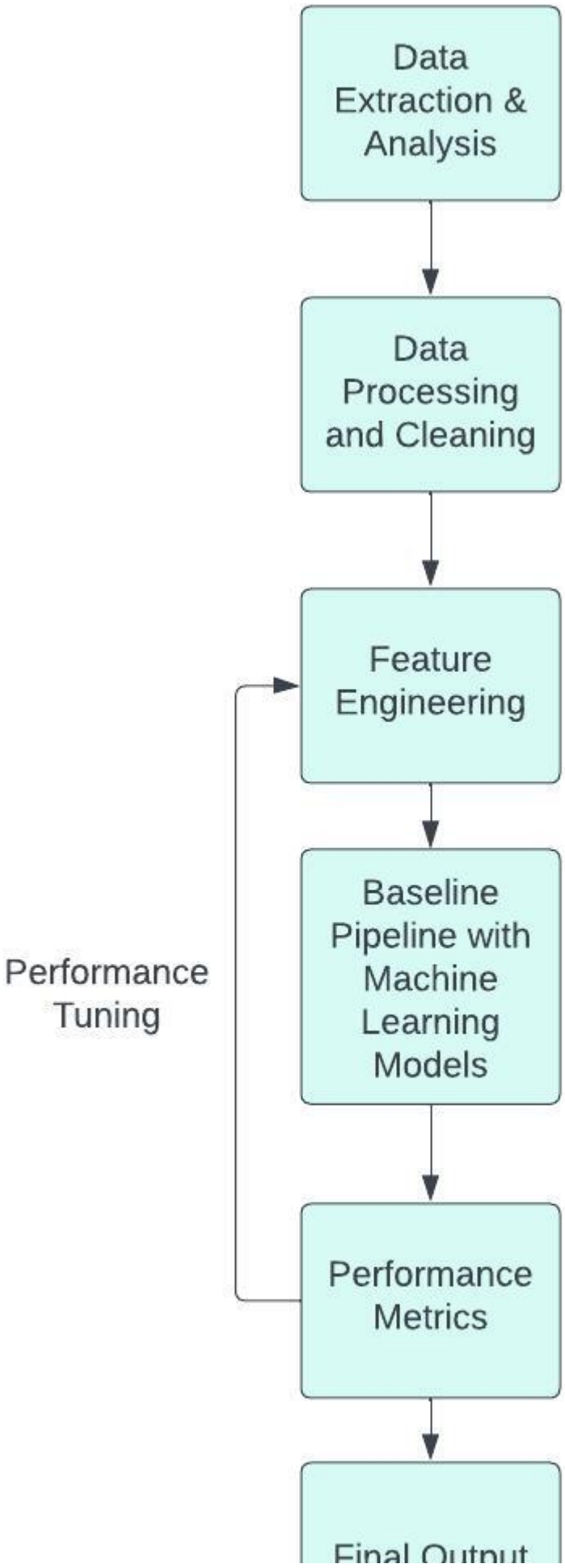
Basic ML Pipline Outline Diagram



Phase 2 Process Diagram



Phase 2 Process Diagram + Tuning Step


Data Extraction & Analysis
Data Processing and Cleaning
Feature Engineering
Baseline Pipeline with Machine Learning Models
Performance Tuning
Performance Metrics
Final Output

Kaggle Submission File Prep

For each SK_ID_CURR in the test set, you must predict a probability for the TARGET variable. The file should contain a header and have the following format:

```
SK_ID_CURR,TARGET
100001,0.1
100005,0.9
100013,0.2
etc.
```

In [ ]:
```
model = lr_pipeline.fit(X_train, y_train)
X_kaggle_test = df_app_test.copy()
test_class_scores = model.predict_proba(X_kaggle_test)[:, 1]
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [ ]:
```
test_class_scores[0:10]
```

Out[ ]:
```
array([0.06095341, 0.23342843, 0.055663  , 0.02879285, 0.12086613,
       0.03513475, 0.02080647, 0.09970679, 0.01536396, 0.11598527])
```

In [ ]:
```
# Submission dataframe
submit_df = df_app_test[['SK_ID_CURR']]
submit_df['TARGET'] = test_class_scores

submit_df.head()
```

```
<ipython-input-175-e10dd3e85e77>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  submit_df['TARGET'] = test_class_scores
```

Out[ ]:

	SK_ID_CURR	TARGET
0	100001	0.060953
1	100005	0.233428
2	100013	0.055663
3	100028	0.028793
4	100038	0.120866

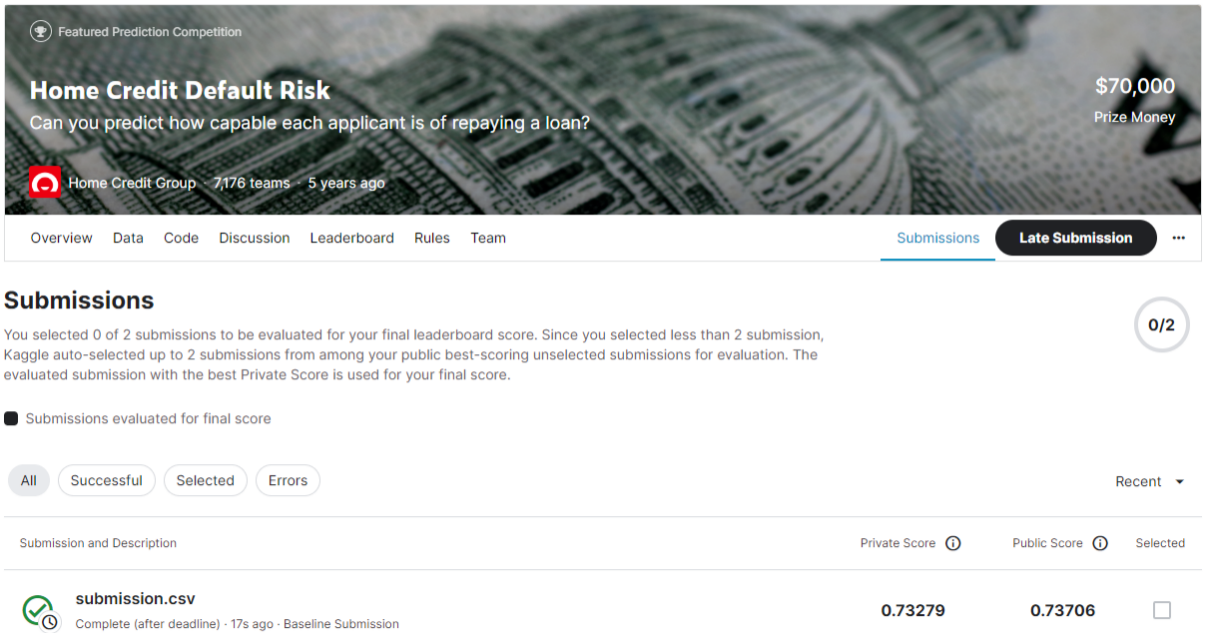
In [ ]:
```
submit_df.to_csv("submission.csv",index=False)
```

Kaggle submission via the command line API

In [ ]:
```
! kaggle competitions submit -c home-credit-default-risk -f submission.csv -m "baseli
```

report submission

Click on this link



Hyper Parameter Tuning

We have selected the following Hyperparamters with respect to the different Machine Learning Algorithms we will try:

1. Logisitc Regresssion: For our LR model the parameters chosen for hyperparameter tuning are:

A. C parameter which controls the penalty strength.
B. Penalty parameter which imposes a penalty to the logistic model for having too many variables. We will try with ['none', l1','l2']
C. Solver which allows us to see useful difference in performance or convergence with different solvers

2. Random Forest

For our RF model we have chosen the following hyperparamters:

A. bootstrap - this parameter means that each tree in the random forest runs on a subset of the observations.

B. max_depth - maximum number of levels allowed in each decision tree

C. forest__max_features - number of features in consideration at every split

D. forest__n_estimators - number of trees in the random forest

1. Decision Tree

For our DT model we have chosen the following parameters:

A. criterion - The function to measure the quality of a split.
B. max_depth - The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
C. min_samples_leaf - The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

Results & Discussion

As per the Project pipeline we first downloaded the data from Kaggle. Then performed EDA on the data to get a better understanding of what all features are present and how they are correlated to the 'Target' variable present in the application_trai datatset. In all we have 9 tables which we had to merge together to get our training and test datasets. We chose 3 Machine Learing models to be run on our given dataset of HCDR. The models are as follows:

1. Logistic Regression
2. Random Forest
3. Decision Tree

We divided the application_train data into 3 subsets of train,valid and test with a random seed of 42 and test size = 0.15. We used 2 metrics : Accuracy and Area Under Curve

Upon running these models using the above mentioned metrics we found out the results of each of the metric for every train,valid and test datasets. We found out that without any Hyperparameter tuning, as a Baseline model, Logistic Regression performed the best with each of the metric. The experiment log table is :

In [ ]:
```
expLog
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC
0	Baseline_Logistic_Regression	0.9200	0.9162	0.9194	0.7485	0.7475	0.7438
1	Baseline_Logistic_Regression	0.9200	0.9162	0.9194	0.7485	0.7475	0.7438
2	Baseline_Random_Forest	0.9999	0.9165	0.9194	1.0000	0.7102	0.7109
3	Baseline_Decision_Tree	1.0000	0.8528	0.8529	1.0000	0.5427	0.5367

In [ ]:
```
```

Conclusion

</br> The aim of the HCDR study is to predict the population's capacity for repayment among those who are economically neglected. This project is important because both the lender and the borrower want accurate estimates. The ML pipelines used by Real-time Home Credit allow them to present their customers with loan offers that have the highest amount and APR because they use EDA to fit the data to the model and generate scores. A user's average, minimum, and maximum balances as well as reported Bureau scores, salary, and other factors are used to generate a credit history, which serves as a gauge of their reliability. The user's timely defaults and repayments can be used to assess repayment habits. Alternative data also includes criteria like location data, social media data, calling/SMS data, etc. In order to complete this project, we would build machine learning pipelines, do exploratory data analysis using the Kaggle datasets, and test many models before deploying one. The estimation of many models was a part of phase 2. When we dug into the data we were able to create a pipline that acuratly predicts the target with 74% AUC. This is significant becuase, both feature selection and data imputation were performed. We chose characteristics and imputed values in the beginning. The values of a few lacking features were filled in. Finally, based on our prior knowledge, we decided which features to incorporate. To find the most effective model, we trained and evaluated a number of them, including Random Forest, Decision tree Model, and Logistic Regression . Out of all the models, the logistic regression model performs the best. We intend to put all models into practice in phase 3 by fine-tuning their individual parameters. In the future we would like to preform hyperparmeter tuning with more compute power, allowing us to accurately merge and estimate the target class with data that we have deemed significant.

In [ ]: