Matt Saperstein
DE300 Homework 1

1. Write a script that creates a database table schema for the input csv file and loads the file to a database of your choice (Amazon S3 is not a database). Some possible choices: mysql, postgres, duckdb, RedshiE. You will have to install the first three in a VM.
    a. I used postgres.
2. From this point on, all of your scripts and code must fetch the data from the underlying database.
    a. I created an initial table database with the original data from data.csv. I then manipulated the data according to all of the following steps to create a final table database.
3. Analyze missing values and come up with a strategy to impute them. You might need different imputation methods for different features. Justify your choices.
    a. Before doing this, I converted all day values from the DAYS_EMPLOYED and DAYS_BIRTH columns from negative to positive.
    b. There are 6 columns that have missing values: HOUSETYPE_MODE, EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3, TOTALAREA_MODE, and AMT_REQ_CREDIT_BUREAU_YEAR.
        i. HOUSETYPE_MODE is the only one categorical variable. To impute the missing values, I used the mode because categorical data like housing type often has a clear most common category, and using this value makes sense when the missing data is assumed to be randomly missing. This keeps the dataset consistent and prevents introducing arbitrary values. For example, if the most common HOUSETYPE_MODE is "Block of Flats", the missing HOUSETYPE_MODE values will be replaced with "Block of Flats".
        ii. EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3, and TOTALAREA_MODE are all normalized continuous variables. To impute the missing values, I used the median of each column's values. Since each column's values range between 0 and 1 and are normalized, the median is a safe choice. The median is less sensitive to outliers compared to the mean, especially in normalized data, and it retains the central tendency of the values. For example, if EXT_SOURCE_1 contains values like 0.2, 0.5, 0.7, and a few missing values, the median of the existing EXT_SOURCE_1 values would be used to fill the missing ones.
        iii. AMT_REQ_CREDIT_BUREAU_YEAR is an integer or count-based column. To impute the missing values, I used 0. AMT_REQ_CREDIT_BUREAU_YEAR represents the number of credit inquiries over a year so it is reasonable to assume that missing values mean no inquiries were made. This makes filling in missing values with 0 a logical choice. It avoids distorting the column with a value that could suggest an average (like a median of 1 or 2). For example, filling in

missing values for AMT_REQ_CREDIT_BUREAU_YEAR with 0 would indicate that we assume the client has had no inquiries in that time frame.

4. For the numerical features, identify outliers and appropriately deal with them. Justify your choices.
    a. Exclude TARGET from outlier handling because I verified in my code that the only possible TARGET values are 0 and 1. Therefore, there are no outliers.
    b. Numerical categories to consider for outliers: CNT_CHILDREN, CNT_FAM_MEMBERS, AMT_INCOME_TOTAL, AMT_CREDIT, TOTALAREA_MODE, EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3, DAYS_EMPLOYED, DAYS_BIRTH, and AMT_REQ_CREDIT_BUREAU_YEAR.
    c. To deal with outliers in the numerical features, I used the Interquartile Range (IQR) method to identify and handle them. I calculated the first (Q1) and third (Q3) quartiles, then computed the IQR as the difference between Q3 and Q1. Outliers were defined as values falling outside the range of 1.5 times the IQR below Q1 or above Q3. These outliers were then capped to the lower or upper bound of the range to ensure that they remained within a reasonable range, thereby minimizing their impact on subsequent analyses. Using this method was the simplest and most consistent way to identify outliers in all eligible numerical features.
5. Compute and analyze the various statistical measures.
    a. I separated the columns into two categories: numerical and categorical.
    b. Numerical: CNT_CHILDREN, CNT_FAM_MEMBERS, AMT_INCOME_TOTAL, AMT_CREDIT, TOTALAREA_MODE, EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3, DAYS_EMPLOYED, DAYS_BIRTH, and AMT_REQ_CREDIT_BUREAU_YEAR.
        i. For the numerical columns, I calculated the mean, standard deviation, minimum, 25% percentile, 50% percentile, 75% percentile, maximum, skewness, and kurtosis. The results are printed when the main.py script is run.
        ii. CNT_CHILDREN, and AMT_REQ_CREDIT_BUREAU_YEAR both have very high positive skewness to the right. CNT_FAM_MEMBERS, AMT_INCOME_TOTAL, and AMT_CREDIT all have moderate positive skewness to the right. EXIT_SOURCE_1 has no skewness and TOTALAREA_MODE and DAYS_BIRTH have slight amounts of negative skewness to the left. EXT_SOURCE_2 and EXT_SOURCE_3 have moderate negative skewness to the left. DAYS_EMPLOYED has very high negative skewness to the left. Lastly, all kurtosis values are less than 3, stating that the distributions for each column have lower peaks and thinner tails than a normal distribution.
    c. Categorical: TARGET, CODE_GENDER, FLAG_OWN_CAR, FLAG_OWN_REALTY, NAME_INCOME_TYPE, HOUSETYPE_MODE, NAME_EDUCATION_TYPE, NAME_HOUSING_TYPE, and NAME_FAMILY_STATUS.

        i. For the categorical columns, I calculated the mode and the counts of each value. The results are printed when the main.py script is run.
        ii. Some conclusions are that most clients have payment difficulties, are female, do not have their own car, have their own house or flat, are working, have a block of flats, had secondary or secondary special education as their highest level of education, have a house or apartment, and are married.

6. Perform the necessary feature transformations. (In all of the steps distinguish between numerical features and categorical features.)
    a. First, I used a log transformation for the CNT_CHILDREN and AMT_REQ_CREDIT_BUREAU_YEAR categories because they both have a very high positive skew. Doing this compresses large values and makes the distribution more normal.
    b. Next, I used advanced target encoding and one-hot encoding to preprocess categorical variables.
        i. I used advanced target encoding for the following categories because they are all binary or ordinal categorical data. This data type involves only two possible categories, which makes it ideal for target encoding because you can easily map each category to the proportion of positive and negative instances of the target class.
            1. Categories: TARGET (Binary: 0 or 1), CODE_GENDER (Binary: M or F), FLAG_OWN_CAR (Binary: Y or N), and FLAG_OWN_REALTY (Binary: Y or N).
        ii. I used one-hot encoding for the following nominal categories because nominal data simply categorizes the data into distinct, unrelated classes. For example, "Pensioner" and "Commercial associate" do not have any inherent ranking—it's just different types of income, making them nominal. Additionally, each of these categories represents a distinct class, and one-hot encoding ensures that each category is treated as a separate feature without implying any sort of ranking or relationship between them.
            1. Categories: NAME_INCOME_TYPE (e.g., Commercial associate, Pensioner), HOUSETYPE_MODE (e.g., block of flats, villa), NAME_EDUCATION_TYPE (e.g., Higher education, Secondary education), NAME_HOUSING_TYPE (e.g., House / apartment, Rented apartment), and NAME_FAMILY_STATUS (e.g., Civil marriage, Single).
7. Generate box and scatter plots. Reason about them.
    a. I generated 8 box plots that download to the folder where you run the code. From these box plots, I came to various conclusions. The credit amount of each person’s loan varies greatly, in a wide range of 100,000 to 1,600,000, with the majority being around 200,000 to 800,000. The total income of each person also varies from about 30,000 to 350,000, with the majority being around 110,000 to 200,000. The number of times each person requests an enquiry to the Credit Bureau is between 0 and 3 times, which is pretty expected knowing that most

people do not apply to open up more than 3 credit cards per year. When comparing credit and income values, it makes sense that the amount of credit is often larger than a person's income, with the majority of credit being from 300,000 to 800,000 and the income being from 100,000 to 200,000. Comparing days employed and days since birth makes sense because people work between 500 and 3500 days while they have usually been alive for 12,000 to 20,000 days. It is interesting comparing the normalized scores from external data sources because EXT_SOURCE_2 and EXT_SOURCE_3 are very similar with a mean around 0.5 and an IQR of 0.4 to 0.7. However, EXT_SOURCE_1 is missing the most values out of every category. As a result, I imputed the missing values with the mean so the majority of EXT_SOURCE_1 values are 0.5. Most people have no children or 1 child while most people have 2-3 family members total. This makes sense because most people are married based on the statistical measures for the NAME_FAMILY_STATUS column. The TOTALAREA_MODE values for most people are around 0.067 to 0.071. I am not sure what to make of this range since this is normalized information about where the client lives.

   b. I generated 4 scatter plots that download to the folder where you run the code. The Age vs Credit Amount plot shows that people at all ages have loans of all amounts of credit. The Employment Duration vs Age plot shows that as people work for longer, they get older (which makes sense logically speaking). There is a whole assortment of values of people based on the number of days they have worked and how long they have been alive. The Income vs Credit Amount plot shows that at certain income values, people take all different values of credit amounts for loans. The last scatter plot shows EXT_SOURCE_2 vs EXT_SOURCE_3. We do not have much information about these normalized scores from external data sources. However, this plot shows a wide range of values for the two scores. Additionally, it is interesting that there is a horizontal line in the plot around scores from source 3 that are 0.55 and around scores from source 3 that are 0.1. This makes me wonder what these scores mean about the people who were given them.

8. The cleaned, transformed, and imputed data should at the end be stored in a new database table.
   a. Created an initial table database and a final table database.

Sources: Used ChatGPT and Claude for help using Docker, AWS, feature transformations, plotting, and writing pseudocode.