Data Preprocessing Assignment  
  
Made By : Arnav Bansal

SID : 23103046

Branch : CSE - G1

1) Exploring Data Types

a. Load the dataset into a Pandas DataFrame.

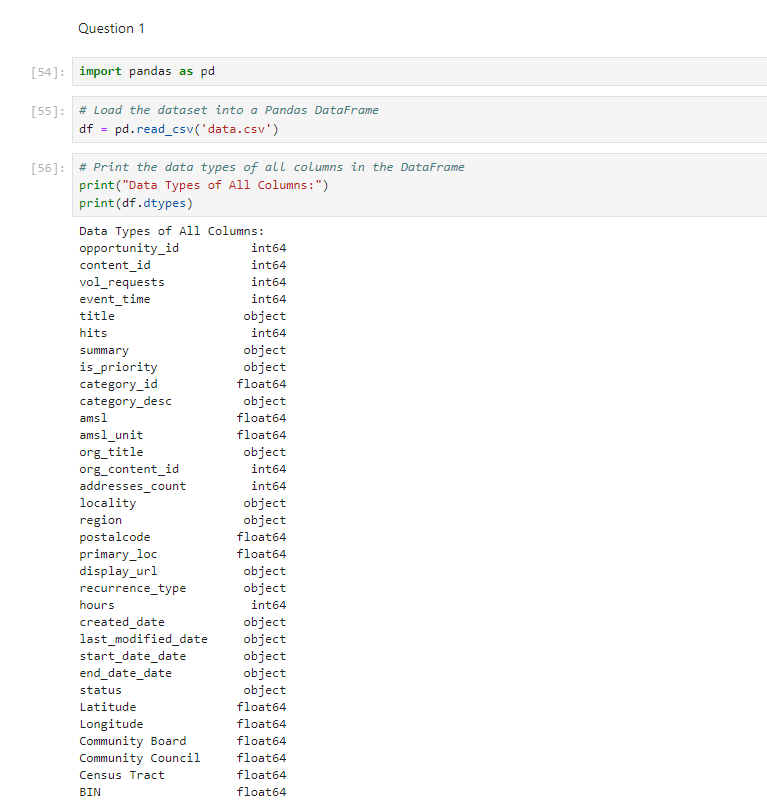
b. Print the data types of all columns in the DataFrame.

c. Identify any columns that are incorrectly typed (e.g., integers stored as objects). Provide

code to correct these data types.

Explanation:

1. Loading the Dataset: The data is loaded into a DataFrame using pd.read\_csv().
2. Printing Initial Data Types: The data types of all columns are printed.
3. Identifying and Converting Incorrectly Typed Columns:
   * Numeric columns that might be stored as object are converted to integers using pd.to\_numeric().
   * Date columns that might be stored as object are converted to datetime using pd.to\_datetime().
4. Printing Updated Data Types: After the conversions, the updated data types are printed.



2) Handling Missing Data

a. Identify and list all columns that have missing values.

b. For columns with at least 3 missing values, remove these columns from the DataFrame.

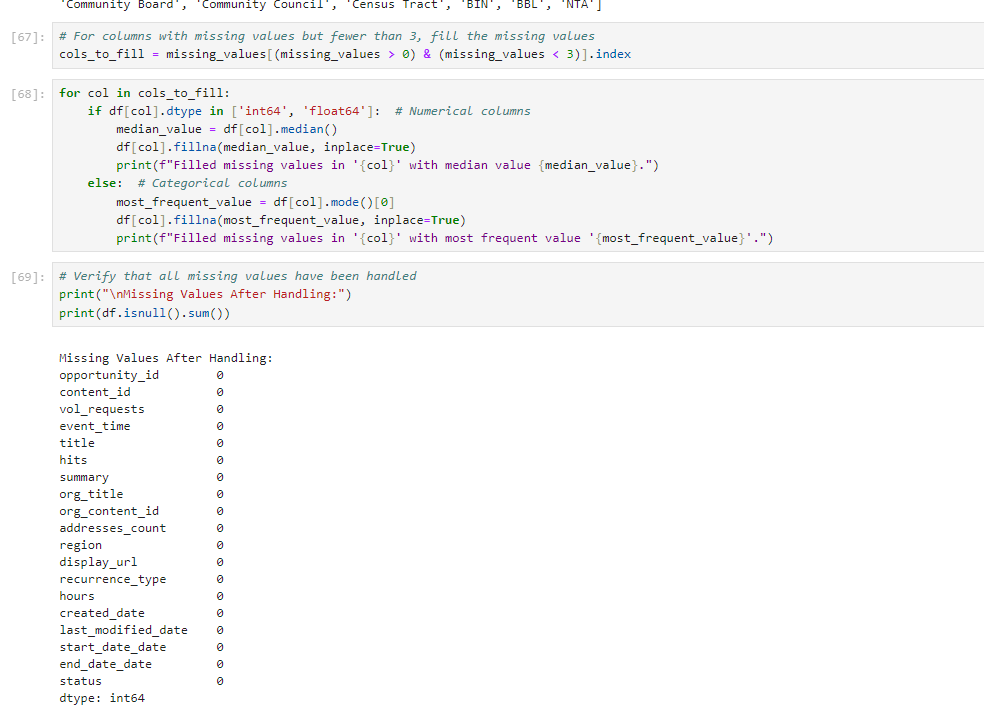
c. For columns with missing values but fewer than 3, fill the missing values using the median

(for numerical columns) or the most frequent value (for categorical columns). Provide the

code used to fill the missing values.

Explanation:

1. Identifying Columns with Missing Values:
   * The code first identifies columns with missing values using df.isnull().sum().
   * It prints out these columns and the number of missing values in each.
2. Removing Columns with at Least 3 Missing Values:
   * Columns with 3 or more missing values are identified and removed from the DataFrame using df.drop(columns=cols\_to\_remove).
3. Filling Missing Values in Remaining Columns:
   * For numerical columns, missing values are filled with the median of the column using df[col].median().
   * For categorical columns, missing values are filled with the most frequent value (mode) using df[col].mode()[0].
   * The code also prints out the values used to fill the missing data.
4. Verification:
   * Finally, the code prints out the DataFrame to confirm that all missing values have been handled.



3) Converting Column Types

a. Choose one column that should be converted to a different data type (e.g.., If you take a

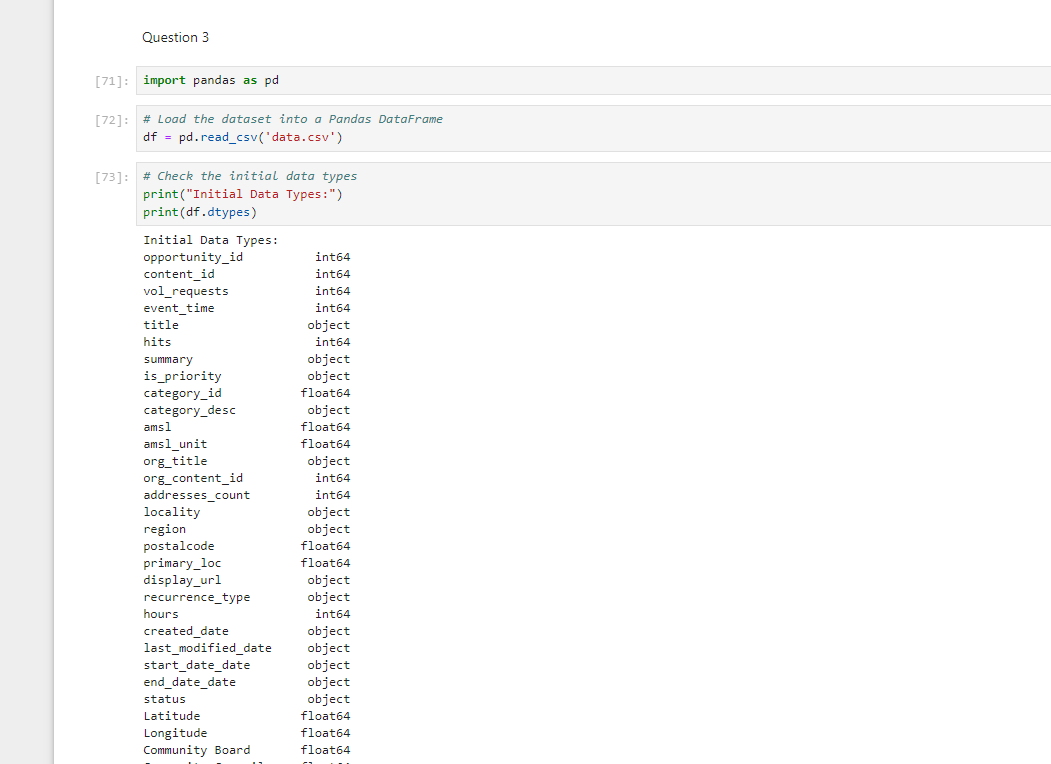
look at the volunteer dataset types, you&#39;ll see that the column hits I s type object. But, if

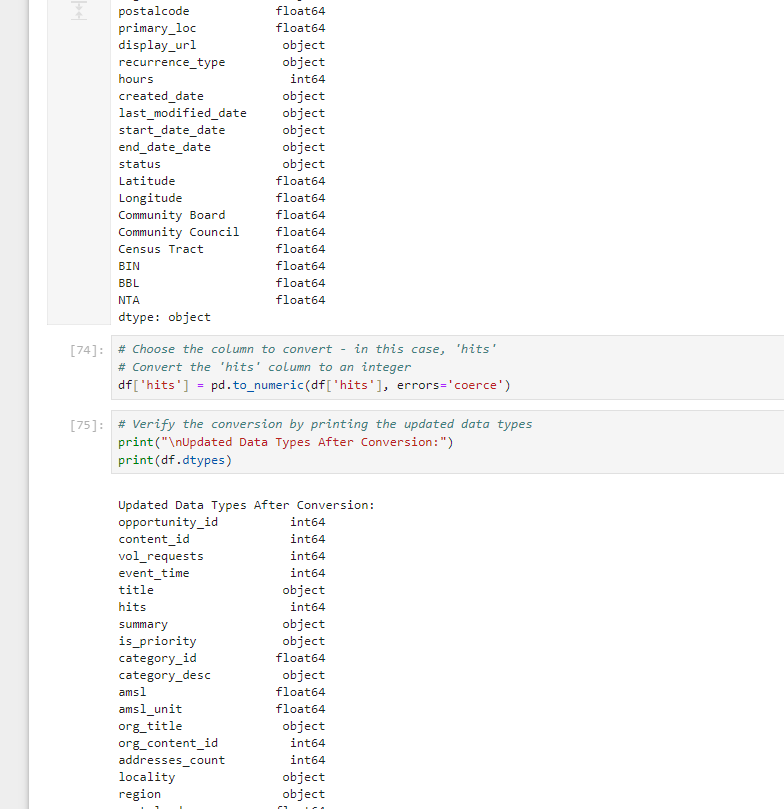
you actually look at the column, you&#39;ll see that it consists of integers)

b. Convert this column to the correct data type and verify the conversion by printing the

updated data types.

Explanation:

1. Loading the Data: The dataset is loaded into a DataFrame using pd.read\_csv().
2. Initial Data Types: The initial data types of all columns are printed using df.dtypes.
3. Convert the hits Column:
   * The pd.to\_numeric() function is used to convert the hits column to an integer type. The errors='coerce' argument ensures that any non-numeric values are converted to NaN (Not a Number), preventing errors during the conversion.
4. Verify the Conversion: The updated data types are printed to confirm that the hits column has been successfully converted to an integer type.



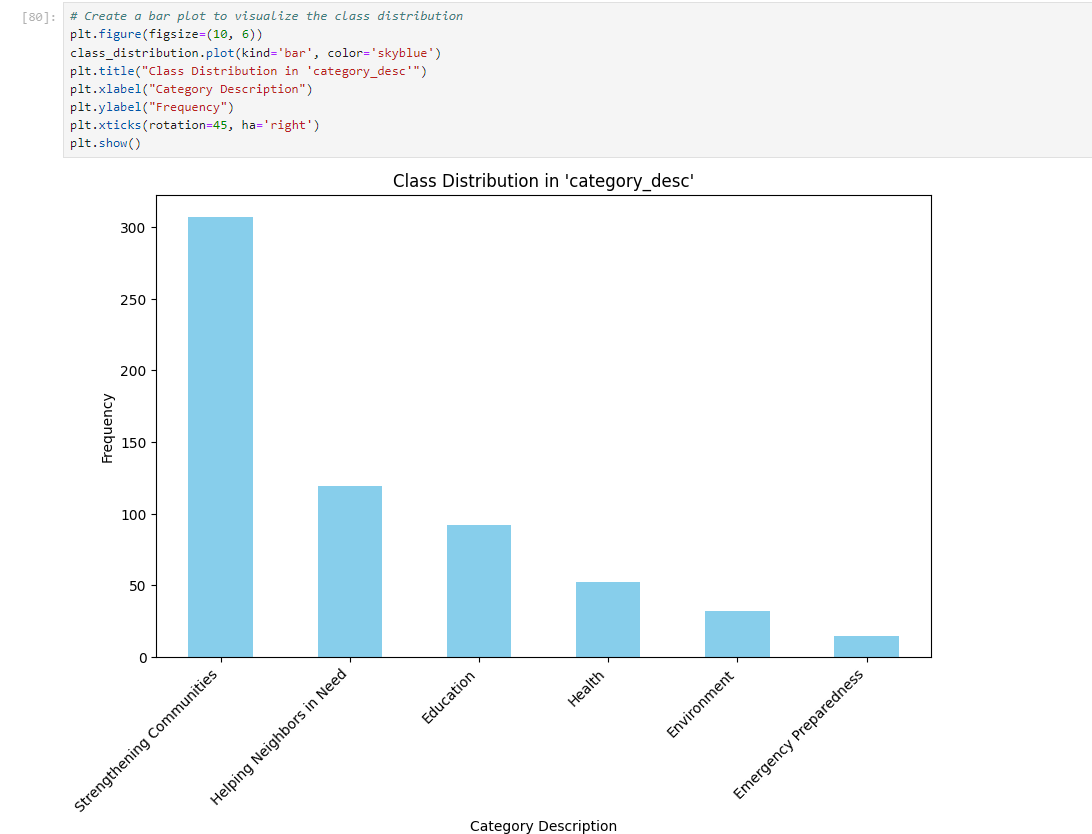
4) Class Distribution Analysis

a. If the dataset contains a categorical column for classification (e.g., category\_desc),

calculate the distribution of classes in this column.

b. Create a bar plot to visualize the class distribution.

Explanation:

1. Loading the Data: The dataset is loaded into a DataFrame using pd.read\_csv().
2. Calculate Class Distribution:
   * The value\_counts() function is used to calculate the frequency of each class in the category\_desc column.
3. Visualize the Class Distribution:
   * A bar plot is created using matplotlib to visualize the distribution of classes.
   * The plot is customized with titles and labels, and the x-axis labels are rotated for better readability.

5) Stratified Sampling

a. Using the class distribution calculated in the previous step, perform stratified sampling to

create a new DataFrame that is a representative sample of the original data. Assume you

want to create a sample that is 20% of the original dataset.

b. Show the class distribution in the sampled DataFrame and compare it with the original

distribution to ensure it is representative.

Explanation :   
Import Libraries: Load pandas for data manipulation and StratifiedShuffleSplit for sampling.

1. Load Dataset: Read the CSV file into a DataFrame.
2. Define Class Column: Specify the column used for stratified sampling.
3. Handle Missing Values: Remove rows with missing values in the class column.
4. Calculate Original Distribution: Determine class proportions in the original dataset.
5. Perform Stratified Sampling: Split the dataset while maintaining class proportions using StratifiedShuffleSplit.
6. Compare Distributions: Check class proportions in the sampled data against the original to ensure representativeness.