Step-1: Load the dataset and check the datatypes

```
import pandas as pd

df_project = pd.read_csv('charitydata.csv')
```

Step-2 : Check for data types and structure

```
print(df_project.info())
print(df_project.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6002 entries, 0 to 6001
Data columns (total 24 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   ID          6002 non-null   int64
 1   reg1        6002 non-null   int64
 2   reg2        6002 non-null   int64
 3   reg3        6002 non-null   int64
 4   reg4        6002 non-null   int64
 5   home        6002 non-null   int64
 6   kids        6002 non-null   int64
 7   hinc        6002 non-null   int64
 8   genf        6002 non-null   int64
 9   wrat        6002 non-null   int64
 10  avhv        5997 non-null   float64
 11  incm        5997 non-null   float64
 12  inca        6002 non-null   int64
 13  plow        6002 non-null   int64
 14  npro        6002 non-null   int64
 15  tgif        6002 non-null   int64
 16  lgif        6002 non-null   int64
 17  rgif        6002 non-null   int64
 18  tdon        6002 non-null   int64
 19  tlag        6002 non-null   int64
 20  agif        6002 non-null   float64
 21  donr        6002 non-null   int64
 22  damt        6002 non-null   int64
 23  Validation  6002 non-null   object
dtypes: float64(3), int64(20), object(1)
memory usage: 1.1+ MB
None
                ID         reg1         reg2         reg3         reg4  \
count  6002.000000  6002.000000  6002.000000  6002.000000  6002.000000
mean   3978.908197     0.201433     0.347051     0.121293     0.132456
std    2301.807612     0.401104     0.476072     0.326495     0.339014
min       1.000000     0.000000     0.000000     0.000000     0.000000
25%    1985.250000     0.000000     0.000000     0.000000     0.000000
50%    3945.500000     0.000000     0.000000     0.000000     0.000000
75%    5963.750000     0.000000     1.000000     0.000000     0.000000
```

```
max    8009.000000     1.000000     1.000000     1.000000     1.000000

              home         kids         hinc         genf         wrat  ...  \
count  6002.000000  6002.000000  6002.000000  6002.000000  6002.000000  ...
mean      0.884538     1.583972     3.939020     0.607797     7.023159  ...
std       0.319605     1.412455     1.401887     0.488282     2.330964  ...
min       0.000000     0.000000     1.000000     0.000000     0.000000  ...
25%       1.000000     0.000000     3.000000     0.000000     6.000000  ...
50%       1.000000     2.000000     4.000000     1.000000     8.000000  ...
75%       1.000000     3.000000     5.000000     1.000000     9.000000  ...
max       1.000000     5.000000     7.000000     1.000000     9.000000  ...

              plow         npro         tgif         lgif         rgif  \
count  6002.000000  6002.000000  6002.000000  6002.000000  6002.000000
mean     13.885038    61.354382   115.799567    22.981340    15.653949
std      13.104649    30.305150    86.537977    29.396428    12.424625
min       0.000000     2.000000    23.000000     3.000000     1.000000
25%       4.000000    37.000000    65.000000    10.000000     7.000000
```

Step 3 : Identify type of variables

```
numeric_vars = df_project.select_dtypes(include=['int64', 'float64']).columns.tolist()
categorical_vars = df_project.select_dtypes(include=['object']).columns.tolist()

print("Numeric Variables:", numeric_vars)
print("Categorical Variables:", categorical_vars)
```

```
Numeric Variables: ['ID', 'reg1', 'reg2', 'reg3', 'reg4', 'home', 'kids', 'hinc', 'genf', 'wrat', 'avhv', 'incm', 'inca', 'plow', 'npro', 'tgif', 'lgif', 'rgif',
Categorical Variables: ['Validation']
```

Step 4 : Check for the missing data

```
missing_data = df_project.isnull().sum()
print("Missing Values Summary:\n", missing_data)
```

```
Missing Values Summary:
 ID           0
reg1          0
reg2          0
reg3          0
reg4          0
home          0
kids          0
hinc          0
genf          0
wrat          0
avhv          5
incm          5
inca          0
plow          0
npro          0
```

```
tgif          0
lgif          0
rgif          0
tdon          0
tlag          0
agif          0
donr          0
damt          0
Validation    0
dtype: int64
```

Step 5 : Handling Missing data

- By imputation through mean for all the numerical variables
- By Filling with mode for all the categorical variables

Ensure whether the variables are in numerical format and fill the values again

```
for col in numeric_vars:
    if df_project[col].isnull().sum() > 0:
        print(f"Handling missing data in numeric column: {col}")
        print(f"Number of missing values before: {df_project[col].isnull().sum()}")
        df[col] = df_project[col].fillna(df_project[col].mean())
        print(f"Number of missing values after: {df_project[col].isnull().sum()}\n")

for col in categorical_vars:
    if df_project[col].isnull().sum() > 0:
        print(f"Handling missing data in categorical column: {col}")
        print(f"Number of missing values before: {df_project[col].isnull().sum()}")
        df_project[col] = df_project[col].fillna(df_project[col].mode()[0])
        print(f"Number of missing values after: {df_project[col].isnull().sum()}\n")
```

```
Handling missing data in numeric column: avhv
Number of missing values before: 5
Number of missing values after: 5

Handling missing data in numeric column: incm
Number of missing values before: 5
Number of missing values after: 5
```

```
df_project['avhv'] = pd.to_numeric(df_project['avhv'], errors='coerce')
df_project['incm'] = pd.to_numeric(df_project['incm'], errors='coerce')

df_project['avhv'] = df_project['avhv'].fillna(df_project['avhv'].mean())
df_project['incm'] = df_project['incm'].fillna(df_project['incm'].mean())

print("Number of missing values in 'avhv' after conversion and filling:", df_project['avhv'].isnull().sum())
print("Number of missing values in 'incm' after conversion and filling:", df_project['incm'].isnull().sum())
```

```
Number of missing values in 'avhv' after conversion and filling: 0
Number of missing values in 'incm' after conversion and filling: 0
```

Step-6 : Outlier Detection by using IQR(Interquantile Range) and can be handled by capping method as shown below

```
import numpy as np

for col in numeric_vars:
    Q1 = df_project[col].quantile(0.25)
    Q3 = df_project[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df_project[(df_project[col] < lower_bound) | (df_project[col] > upper_bound)]
    print(f"Outliers detected in {col}: {len(outliers)}")

    df_project[col] = np.where(df_project[col] < lower_bound, lower_bound, df_project[col])
    df_project[col] = np.where(df_project[col] > upper_bound, upper_bound, df_project[col])

    print(f"Outliers in {col} capped. Number of outliers after handling: {((df_project[col] < lower_bound) | (df_project[col] > upper_bound)).sum()}\n")
```

```
Outliers detected in ID: 0
Outliers in ID capped. Number of outliers after handling: 0

Outliers detected in reg1: 1209
Outliers in reg1 capped. Number of outliers after handling: 0

Outliers detected in reg2: 0
Outliers in reg2 capped. Number of outliers after handling: 0

Outliers detected in reg3: 728
Outliers in reg3 capped. Number of outliers after handling: 0

Outliers detected in reg4: 795
Outliers in reg4 capped. Number of outliers after handling: 0

Outliers detected in home: 693
Outliers in home capped. Number of outliers after handling: 0

Outliers detected in kids: 0
```

```
Outliers in kids capped. Number of outliers after handling: 0

Outliers detected in hinc: 0
Outliers in hinc capped. Number of outliers after handling: 0

Outliers detected in genf: 0
Outliers in genf capped. Number of outliers after handling: 0

Outliers detected in wrat: 282
Outliers in wrat capped. Number of outliers after handling: 0

Outliers detected in avhv: 202
Outliers in avhv capped. Number of outliers after handling: 0

Outliers detected in incm: 211
Outliers in incm capped. Number of outliers after handling: 0

Outliers detected in inca: 208
Outliers in inca capped. Number of outliers after handling: 0

Outliers detected in plow: 218
Outliers in plow capped. Number of outliers after handling: 0

Outliers detected in npro: 7
Outliers in npro capped. Number of outliers after handling: 0

Outliers detected in tgif: 340
Outliers in tgif capped. Number of outliers after handling: 0

Outliers detected in lgif: 511
Outliers in lgif capped. Number of outliers after handling: 0

Outliers detected in rgif: 276
Outliers in rgif capped. Number of outliers after handling: 0

Outliers detected in tdon: 227
Outliers in tdon capped. Number of outliers after handling: 0

Outliers detected in tlag: 496
```