Identify_Customer_Segments

September 9, 2023

1 Project: Identify Customer Segments

In this project, you will apply unsupervised learning techniques to identify segments of the population that form the core customer base for a mail-order sales company in Germany. These segments can then be used to direct marketing campaigns towards audiences that will have the highest expected rate of returns. The data that you will use has been provided by our partners at Bertelsmann Arvato Analytics, and represents a real-life data science task.

This notebook will help you complete this task by providing a framework within which you will perform your analysis steps. In each step of the project, you will see some text describing the subtask that you will perform, followed by one or more code cells for you to complete your work. **Feel free to add additional code and markdown cells as you go along so that you can explore everything in precise chunks.** The code cells provided in the base template will outline only the major tasks, and will usually not be enough to cover all of the minor tasks that comprise it.

It should be noted that while there will be precise guidelines on how you should handle certain tasks in the project, there will also be places where an exact specification is not provided. **There will be times in the project where you will need to make and justify your own decisions on how to treat the data.** These are places where there may not be only one way to handle the data. In real-life tasks, there may be many valid ways to approach an analysis task. One of the most important things you can do is clearly document your approach so that other scientists can understand the decisions you've made.

At the end of most sections, there will be a Markdown cell labeled **Discussion**. In these cells, you will report your findings for the completed section, as well as document the decisions that you made in your approach to each subtask. **Your project will be evaluated not just on the code used to complete the tasks outlined, but also your communication about your observations and conclusions at each stage.**

In [1]:
```
# import libraries here; add more as necessary
import numpy as np
import pandas as pd
import re
import warnings
import matplotlib.pyplot as plt
from sklearn.preprocessing import Imputer
import re
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import seaborn as sns

# magic word for producing visualizations in notebook
%matplotlib inline

'''
Import note: The classroom currently uses sklearn version 0.19.
If you need to use an imputer, it is available in sklearn.preprocessing.Imputer,
instead of sklearn.impute as in newer versions of sklearn.
'''
```

```
Out[1]: '\nImport note: The classroom currently uses sklearn version 0.19.\nIf you need to use a
```

1.0.1 Step 0: Load the Data

There are four files associated with this project (not including this one):

- `Udacity_AZDIAS_Subset.csv`: Demographics data for the general population of Germany; 891211 persons (rows) x 85 features (columns).
- `Udacity_CUSTOMERS_Subset.csv`: Demographics data for customers of a mail-order company; 191652 persons (rows) x 85 features (columns).
- `Data_Dictionary.md`: Detailed information file about the features in the provided datasets.
- `AZDIAS_Feature_Summary.csv`: Summary of feature attributes for demographics data; 85 features (rows) x 4 columns

Each row of the demographics files represents a single person, but also includes information outside of individuals, including information about their household, building, and neighborhood. You will use this information to cluster the general population into groups with similar demographic properties. Then, you will see how the people in the customers dataset fit into those created clusters. The hope here is that certain clusters are over-represented in the customers data, as compared to the general population; those over-represented clusters will be assumed to be part of the core userbase. This information can then be used for further applications, such as targeting for a marketing campaign.

To start off with, load in the demographics data for the general population into a pandas DataFrame, and do the same for the feature attributes summary. Note for all of the `.csv` data files in this project: they're semicolon (`;`) delimited, so you'll need an additional argument in your `read_csv()` call to read in the data properly. Also, considering the size of the main dataset, it may take some time for it to load completely.

Once the dataset is loaded, it's recommended that you take a little bit of time just browsing the general structure of the dataset and feature summary file. You'll be getting deep into the innards of the cleaning in the first major step of the project, so gaining some general familiarity can help you get your bearings.

```
In [2]: # Load in the general demographics data.
        azdias = pd.read_csv('Udacity_AZDIAS_Subset.csv',sep=';')

        # Load in the feature summary file.
        feat_info =  pd.read_csv('AZDIAS_Feature_Summary.csv',sep=';')
```

```
In [3]: # Check the structure of the data after it's loaded (e.g. print the number of
        # rows and columns, print the first few rows).

        print("General demographics data shape:", azdias.shape)

        # Print the first few rows of the demographics data
        azdias.shape
        display (azdias.describe ())

        azdias.head()
```

```
General demographics data shape: (891221, 85)
```

```
            AGER_TYP  ALTERSKATEGORIE_GROB      ANREDE_KZ  CJT_GESAMTTYP  \
count  891221.000000         891221.000000  891221.000000  886367.000000
mean       -0.358435              2.777398       1.522098       3.632838
std         1.198724              1.068775       0.499512       1.595021
min        -1.000000              1.000000       1.000000       1.000000
25%        -1.000000              2.000000       1.000000       2.000000
50%        -1.000000              3.000000       2.000000       4.000000
75%        -1.000000              4.000000       2.000000       5.000000
max         3.000000              9.000000       2.000000       6.000000

       FINANZ_MINIMALIST  FINANZ_SPARER  FINANZ_VORSORGER  FINANZ_ANLEGER  \
count      891221.000000  891221.000000     891221.000000   891221.000000
mean            3.074528       2.821039          3.401106        3.033328
std             1.321055       1.464749          1.322134        1.529603
min             1.000000       1.000000          1.000000        1.000000
25%             2.000000       1.000000          3.000000        2.000000
50%             3.000000       3.000000          3.000000        3.000000
75%             4.000000       4.000000          5.000000        5.000000
max             5.000000       5.000000          5.000000        5.000000

       FINANZ_UNAUFFAELLIGER  FINANZ_HAUSBAUER      ...           PLZ8_ANTG1  \
count          891221.000000     891221.000000      ...        774706.000000
mean                2.874167          3.075121      ...             2.253330
std                 1.486731          1.353248      ...             0.972008
min                 1.000000          1.000000      ...             0.000000
25%                 2.000000          2.000000      ...             1.000000
50%                 3.000000          3.000000      ...             2.000000
75%                 4.000000          4.000000      ...             3.000000
max                 5.000000          5.000000      ...             4.000000

          PLZ8_ANTG2     PLZ8_ANTG3     PLZ8_ANTG4    PLZ8_BAUMAX  \
count  774706.000000  774706.000000  774706.000000  774706.000000
mean        2.801858       1.595426       0.699166       1.943913
std         0.920309       0.986736       0.727137       1.459654
```

```
min          0.000000      0.000000      0.000000      1.000000
25%          2.000000      1.000000      0.000000      1.000000
50%          3.000000      2.000000      1.000000      1.000000
75%          3.000000      2.000000      1.000000      3.000000
max          4.000000      3.000000      2.000000      5.000000

            PLZ8_HHZ      PLZ8_GBZ        ARBEIT   ORTSGR_KLS9  \
count  774706.000000 774706.000000 794005.000000 794005.000000
mean        3.612821      3.381087      3.167854      5.293002
std         0.973967      1.111598      1.002376      2.303739
min         1.000000      1.000000      1.000000      0.000000
25%         3.000000      3.000000      3.000000      4.000000
50%         4.000000      3.000000      3.000000      5.000000
75%         4.000000      4.000000      4.000000      7.000000
max         5.000000      5.000000      9.000000      9.000000

           RELAT_AB
count  794005.00000
mean        3.07222
std         1.36298
min         1.00000
25%         2.00000
50%         3.00000
75%         4.00000
max         9.00000

[8 rows x 81 columns]
```

```
Out[3]:    AGER_TYP  ALTERSKATEGORIE_GROB  ANREDE_KZ  CJT_GESAMTTYP  \
        0        -1                     2          1            2.0
        1        -1                     1          2            5.0
        2        -1                     3          2            3.0
        3         2                     4          2            2.0
        4        -1                     3          1            5.0

           FINANZ_MINIMALIST  FINANZ_SPARER  FINANZ_VORSORGER  FINANZ_ANLEGER  \
        0                  3              4                 3               5
        1                  1              5                 2               5
        2                  1              4                 1               2
        3                  4              2                 5               2
        4                  4              3                 4               1

           FINANZ_UNAUFFAELLIGER  FINANZ_HAUSBAUER    ...     PLZ8_ANTG1  PLZ8_ANTG2  \
        0                      5                 3    ...            NaN         NaN
        1                      4                 5    ...            2.0         3.0
        2                      3                 5    ...            3.0         3.0
        3                      1                 2    ...            2.0         2.0
```

```
4                       3                  2    ...            2.0          4.0

   PLZ8_ANTG3  PLZ8_ANTG4  PLZ8_BAUMAX  PLZ8_HHZ  PLZ8_GBZ  ARBEIT  \
0         NaN         NaN          NaN       NaN       NaN     NaN
1         2.0         1.0          1.0       5.0       4.0     3.0
2         1.0         0.0          1.0       4.0       4.0     3.0
3         2.0         0.0          1.0       3.0       4.0     2.0
4         2.0         1.0          2.0       3.0       3.0     4.0

   ORTSGR_KLS9  RELAT_AB
0          NaN       NaN
1          5.0       4.0
2          5.0       2.0
3          3.0       3.0
4          6.0       5.0

[5 rows x 85 columns]
```

In [4]:
```
feat_info.shape
feat_info.head()
```

Out[4]:

	attribute	information_level	type	missing_or_unknown
0	AGER_TYP	person	categorical	[-1,0]
1	ALTERSKATEGORIE_GROB	person	ordinal	[-1,0,9]
2	ANREDE_KZ	person	categorical	[-1,0]
3	CJT_GESAMTTYP	person	categorical	[0]
4	FINANZ_MINIMALIST	person	ordinal	[-1]

Tip: Add additional cells to keep everything in reasonably-sized chunks! Keyboard shortcut `esc --> a` (press escape to enter command mode, then press the 'A' key) adds a new cell before the active cell, and `esc --> b` adds a new cell after the active cell. If you need to convert an active cell to a markdown cell, use `esc --> m` and to convert to a code cell, use `esc --> y`.

1.1 Step 1: Preprocessing

1.1.1 Step 1.1: Assess Missing Data

The feature summary file contains a summary of properties for each demographics data column. You will use this file to help you make cleaning decisions during this stage of the project. First of all, you should assess the demographics data in terms of missing data. Pay attention to the following points as you perform your analysis, and take notes on what you observe. Make sure that you fill in the **Discussion** cell with your findings and decisions at the end of each step that has one!

Step 1.1.1: Convert Missing Value Codes to NaNs The fourth column of the feature attributes summary (loaded in above as `feat_info`) documents the codes from the data dictionary that indicate missing or unknown data. While the file encodes this as a list (e.g. `[-1,0]`), this will get read in as a string object. You'll need to do a little bit of parsing to make use of it to identify and clean the data. Convert data that matches a 'missing' or 'unknown' value code into a numpy NaN value. You might want to see how much data takes on a 'missing' or 'unknown' code, and how much data is naturally missing, as a point of interest.

As one more reminder, you are encouraged to add additional cells to break up your analysis into manageable chunks.

In [5]:
```
azdias_cleaned = azdias.copy() #for the manipulation of the data
```

In [6]:
```
# Identify NaNs in dataframe before processing

missing_data_counts = azdias_cleaned.isnull().sum()
print("Missing Values Count in Each Column:")
print(missing_data_counts)
```

```
Missing Values Count in Each Column:
AGER_TYP                     0
ALTERSKATEGORIE_GROB         0
ANREDE_KZ                    0
CJT_GESAMTTYP             4854
FINANZ_MINIMALIST            0
FINANZ_SPARER                0
FINANZ_VORSORGER             0
FINANZ_ANLEGER               0
FINANZ_UNAUFFAELLIGER        0
FINANZ_HAUSBAUER             0
FINANZTYP                    0
GEBURTSJAHR                  0
GFK_URLAUBERTYP           4854
GREEN_AVANTGARDE             0
HEALTH_TYP                   0
LP_LEBENSPHASE_FEIN       4854
LP_LEBENSPHASE_GROB       4854
LP_FAMILIE_FEIN           4854
LP_FAMILIE_GROB           4854
LP_STATUS_FEIN            4854
LP_STATUS_GROB            4854
NATIONALITAET_KZ             0
PRAEGENDE_JUGENDJAHRE        0
RETOURTYP_BK_S            4854
SEMIO_SOZ                    0
SEMIO_FAM                    0
SEMIO_REL                    0
SEMIO_MAT                    0
SEMIO_VERT                   0
SEMIO_LUST                   0
                         ...
OST_WEST_KZ              93148
WOHNLAGE                 93148
```

```
CAMEO_DEUG_2015          98979
CAMEO_DEU_2015           98979
CAMEO_INTL_2015          98979
KBA05_ANTG1             133324
KBA05_ANTG2             133324
KBA05_ANTG3             133324
KBA05_ANTG4             133324
KBA05_BAUMAX            133324
KBA05_GBZ               133324
BALLRAUM                 93740
EWDICHTE                 93740
INNENSTADT               93740
GEBAEUDETYP_RASTER       93155
KKK                     121196
MOBI_REGIO              133324
ONLINE_AFFINITAET         4854
REGIOTYP                121196
KBA13_ANZAHL_PKW        105800
PLZ8_ANTG1              116515
PLZ8_ANTG2              116515
PLZ8_ANTG3              116515
PLZ8_ANTG4              116515
PLZ8_BAUMAX             116515
PLZ8_HHZ                116515
PLZ8_GBZ                116515
ARBEIT                   97216
ORTSGR_KLS9              97216
RELAT_AB                 97216
Length: 85, dtype: int64
```

In [7]:
```
# Identify missing or unknown data values and convert them to NaNs.
# Referenced: https://stackoverflow.com/questions/73114793/how-can-i-iterate-through-eac

# Iterate over each row of feat_info and process missing_or_unknown values
for index, row in feat_info.iterrows():
    missing_or_unknown_str = row['missing_or_unknown']

    # Remove unnecessary characters and split the string into a list of values
    missing_or_unknown_list = missing_or_unknown_str.strip('[]').split(',')

    # Convert non-'X' and non-'XX' values to integers, and keep 'X' and 'XX' as they are
    missing_or_unknown_list = [int(value) if value not in ['X', 'XX', ''] else value for

    # Check if the resulting list contains missing or unknown values
    if missing_or_unknown_list != ['']:
        # Replace missing_or_unknown values with NaNs in the azdias DataFrame
        azdias_cleaned = azdias_cleaned.replace({row['attribute']: missing_or_unknown_li
```

```
In [8]: missing_data_counts = azdias_cleaned.isnull().sum()
        print("Missing Values Count in Each Column:")
        print(missing_data_counts)
```

```
Missing Values Count in Each Column:
AGER_TYP                 685843
ALTERSKATEGORIE_GROB       2881
ANREDE_KZ                     0
CJT_GESAMTTYP              4854
FINANZ_MINIMALIST             0
FINANZ_SPARER                 0
FINANZ_VORSORGER              0
FINANZ_ANLEGER                0
FINANZ_UNAUFFAELLIGER         0
FINANZ_HAUSBAUER              0
FINANZTYP                     0
GEBURTSJAHR              392318
GFK_URLAUBERTYP            4854
GREEN_AVANTGARDE              0
HEALTH_TYP               111196
LP_LEBENSPHASE_FEIN       97632
LP_LEBENSPHASE_GROB       94572
LP_FAMILIE_FEIN           77792
LP_FAMILIE_GROB           77792
LP_STATUS_FEIN             4854
LP_STATUS_GROB             4854
NATIONALITAET_KZ         108315
PRAEGENDE_JUGENDJAHRE    108164
RETOURTYP_BK_S             4854
SEMIO_SOZ                     0
SEMIO_FAM                     0
SEMIO_REL                     0
SEMIO_MAT                     0
SEMIO_VERT                    0
SEMIO_LUST                    0
                          ...
OST_WEST_KZ               93148
WOHNLAGE                  93148
CAMEO_DEUG_2015           99352
CAMEO_DEU_2015            99352
CAMEO_INTL_2015           99352
KBA05_ANTG1              133324
KBA05_ANTG2              133324
KBA05_ANTG3              133324
KBA05_ANTG4              133324
KBA05_BAUMAX             476524
KBA05_GBZ                133324
BALLRAUM                  93740
```

```
EWDICHTE                   93740
INNENSTADT                 93740
GEBAEUDETYP_RASTER         93155
KKK                       158064
MOBI_REGIO                133324
ONLINE_AFFINITAET           4854
REGIOTYP                  158064
KBA13_ANZAHL_PKW          105800
PLZ8_ANTG1                116515
PLZ8_ANTG2                116515
PLZ8_ANTG3                116515
PLZ8_ANTG4                116515
PLZ8_BAUMAX               116515
PLZ8_HHZ                  116515
PLZ8_GBZ                  116515
ARBEIT                     97375
ORTSGR_KLS9                97274
RELAT_AB                   97375
Length: 85, dtype: int64
```

Step 1.1.2: Assess Missing Data in Each Column How much missing data is present in each column? There are a few columns that are outliers in terms of the proportion of values that are missing. You will want to use matplotlib's `hist()` function to visualize the distribution of missing value counts to find these columns. Identify and document these columns. While some of these columns might have justifications for keeping or re-encoding the data, for this project you should just remove them from the dataframe. (Feel free to make remarks about these outlier columns in the discussion, however!)

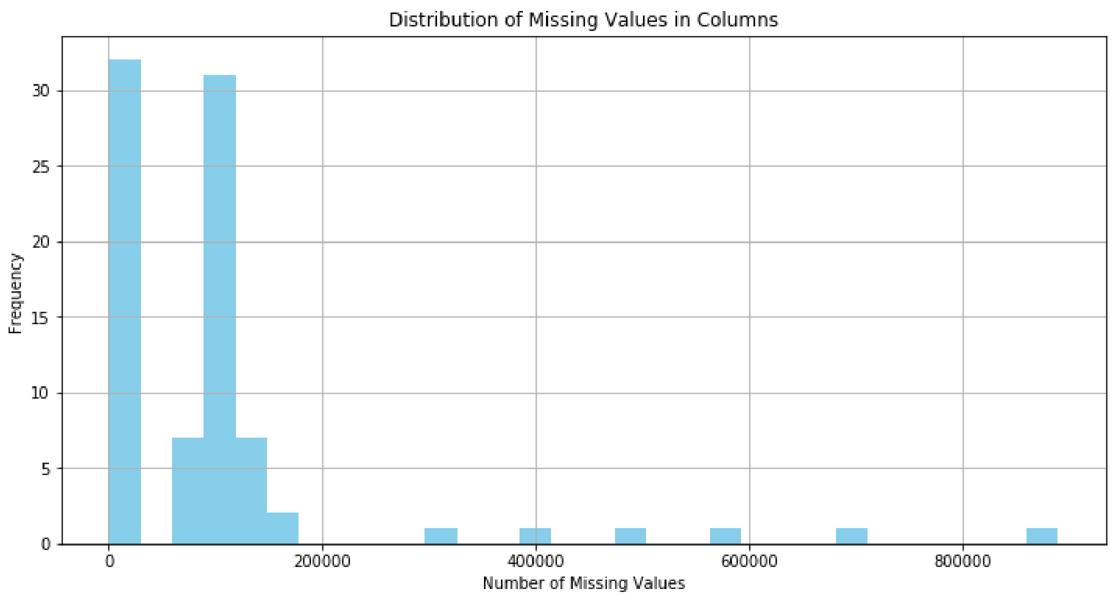
For the remaining features, are there any patterns in which columns have, or share, missing data?

In [9]:
```python
# Perform an assessment of how much missing data there is in each column of the
# dataset.

missing_percentage = (azdias_cleaned.isnull().sum() / len(azdias_cleaned)) * 100

threshold = 25
outlier_columns = missing_percentage[missing_percentage > threshold].index
print("Columns with Missing Value Percentages > 25%:")
print(outlier_columns)

# Looks like there is a '\n' in the list for some reason
outlier_columns = [item.strip() for item in outlier_columns]
print(outlier_columns)
```

```
Columns with Missing Value Percentages > 25%:
Index(['AGER_TYP', 'GEBURTSJAHR', 'TITEL_KZ', 'ALTER_HH', 'KK_KUNDENTYP',
       'KBA05_BAUMAX'],
```

```
      dtype='object')
['AGER_TYP', 'GEBURTSJAHR', 'TITEL_KZ', 'ALTER_HH', 'KK_KUNDENTYP', 'KBA05_BAUMAX']
```

In [10]:
```
# Investigate patterns in the amount of missing data in each column.
# Calculate the number of missing values in each column

missing_values_count = azdias_cleaned.isnull().sum()

# Plot the distribution of missing value counts
plt.figure(figsize=(12, 6))
plt.hist(missing_values_count, bins=30, color='skyblue')
plt.xlabel('Number of Missing Values')
plt.ylabel('Frequency')
plt.title('Distribution of Missing Values in Columns')
plt.grid(True)
plt.show()
```



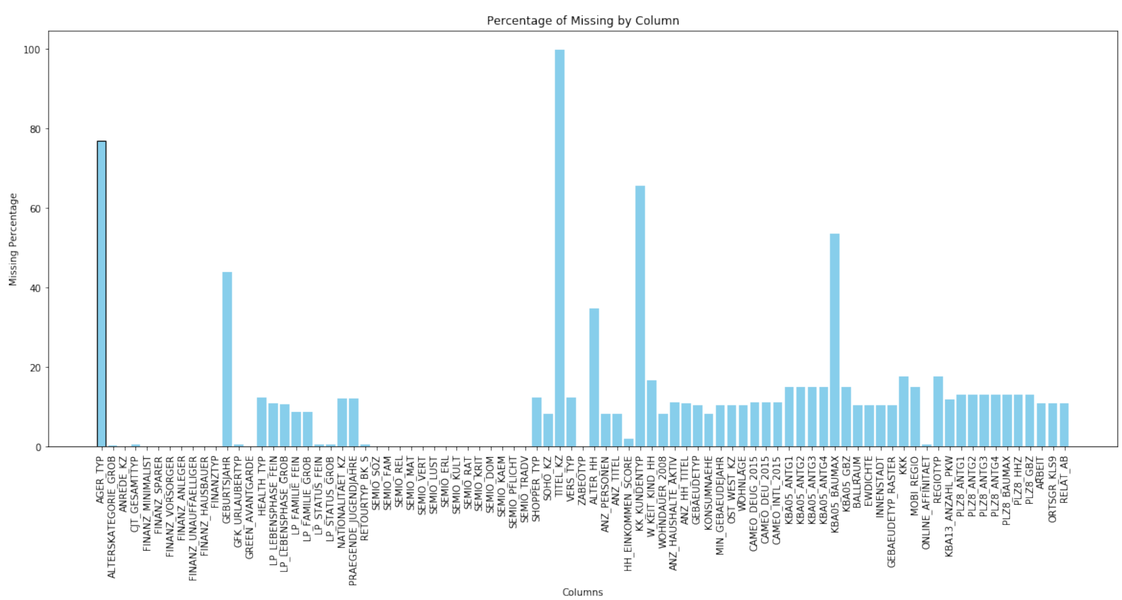
In [11]:
```
# Create DataFrame to store the missing values information
missing_values_df = pd.DataFrame({'Column': azdias_cleaned.columns, 'Missing Percentage

# Sort DataFrame by missing percentage in descending order
missing_values_df = missing_values_df.sort_values(by='Missing Percentage', ascending=Fa

# Display the missing values information
print(missing_values_df)
```

```
                                        Column  Missing Percentage
TITEL_KZ                              TITEL_KZ           99.757636
```

AGER_TYP	AGER_TYP	76.955435
KK_KUNDENTYP	KK_KUNDENTYP	65.596749
KBA05_BAUMAX	KBA05_BAUMAX	53.468668
GEBURTSJAHR	GEBURTSJAHR	44.020282
ALTER_HH	ALTER_HH	34.813699
KKK	KKK	17.735668
REGIOTYP	REGIOTYP	17.735668
W_KEIT_KIND_HH	W_KEIT_KIND_HH	16.605084
KBA05_ANTG1	KBA05_ANTG1	14.959701
KBA05_ANTG2	KBA05_ANTG2	14.959701
KBA05_ANTG3	KBA05_ANTG3	14.959701
KBA05_ANTG4	KBA05_ANTG4	14.959701
KBA05_GBZ	KBA05_GBZ	14.959701
MOBI_REGIO	MOBI_REGIO	14.959701
PLZ8_ANTG3	PLZ8_ANTG3	13.073637
PLZ8_ANTG2	PLZ8_ANTG2	13.073637
PLZ8_GBZ	PLZ8_GBZ	13.073637
PLZ8_HHZ	PLZ8_HHZ	13.073637
PLZ8_ANTG1	PLZ8_ANTG1	13.073637
PLZ8_BAUMAX	PLZ8_BAUMAX	13.073637
PLZ8_ANTG4	PLZ8_ANTG4	13.073637
VERS_TYP	VERS_TYP	12.476816
HEALTH_TYP	HEALTH_TYP	12.476816
SHOPPER_TYP	SHOPPER_TYP	12.476816
NATIONALITAET_KZ	NATIONALITAET_KZ	12.153551
PRAEGENDE_JUGENDJAHRE	PRAEGENDE_JUGENDJAHRE	12.136608
KBA13_ANZAHL_PKW	KBA13_ANZAHL_PKW	11.871354
ANZ_HAUSHALTE_AKTIV	ANZ_HAUSHALTE_AKTIV	11.176913
CAMEO_INTL_2015	CAMEO_INTL_2015	11.147852
...	...	...
CJT_GESAMTTYP	CJT_GESAMTTYP	0.544646
LP_STATUS_FEIN	LP_STATUS_FEIN	0.544646
LP_STATUS_GROB	LP_STATUS_GROB	0.544646
RETOURTYP_BK_S	RETOURTYP_BK_S	0.544646
ONLINE_AFFINITAET	ONLINE_AFFINITAET	0.544646
ALTERSKATEGORIE_GROB	ALTERSKATEGORIE_GROB	0.323264
FINANZ_UNAUFFAELLIGER	FINANZ_UNAUFFAELLIGER	0.000000
FINANZTYP	FINANZTYP	0.000000
FINANZ_HAUSBAUER	FINANZ_HAUSBAUER	0.000000
GREEN_AVANTGARDE	GREEN_AVANTGARDE	0.000000
FINANZ_SPARER	FINANZ_SPARER	0.000000
FINANZ_MINIMALIST	FINANZ_MINIMALIST	0.000000
FINANZ_VORSORGER	FINANZ_VORSORGER	0.000000
FINANZ_ANLEGER	FINANZ_ANLEGER	0.000000
ANREDE_KZ	ANREDE_KZ	0.000000
SEMIO_KAEM	SEMIO_KAEM	0.000000
SEMIO_SOZ	SEMIO_SOZ	0.000000
SEMIO_PFLICHT	SEMIO_PFLICHT	0.000000

```
SEMIO_FAM                              SEMIO_FAM            0.000000
SEMIO_REL                              SEMIO_REL            0.000000
SEMIO_MAT                              SEMIO_MAT            0.000000
SEMIO_VERT                            SEMIO_VERT            0.000000
SEMIO_LUST                            SEMIO_LUST            0.000000
SEMIO_ERL                              SEMIO_ERL            0.000000
SEMIO_KULT                            SEMIO_KULT            0.000000
SEMIO_RAT                              SEMIO_RAT            0.000000
SEMIO_KRIT                            SEMIO_KRIT            0.000000
SEMIO_DOM                              SEMIO_DOM            0.000000
SEMIO_TRADV                          SEMIO_TRADV            0.000000
ZABEOTYP                                ZABEOTYP            0.000000

[85 rows x 2 columns]
```

In [12]:
```python
# Plot the percentage of missing values by number of columns
plt.figure(figsize=(20, 8))
plt.bar(range(len(missing_percentage)), missing_percentage, tick_label=missing_percenta
plt.xlabel('Columns')
plt.ylabel('Missing Percentage')
plt.title('Percentage of Missing by Column')
plt.xticks(rotation=90)
plt.show()
```



In [13]:
```python
# Remove the outlier columns from the dataset. (You'll perform other data
# engineering tasks such as re-encoding and imputation later.)
```

```python
# Remove the outlier columns from the dataset
azdias_cleaned.drop(columns=outlier_columns, inplace=True)

# Confirm removal
print("Previous DataFrame shape before removing outlier columns:", azdias.shape)
print("Updated DataFrame shape after removing outlier columns:", azdias_cleaned.shape)
```

```
Previous DataFrame shape before removing outlier columns: (891221, 85)
Updated DataFrame shape after removing outlier columns: (891221, 79)
```

Discussion 1.1.2: Assess Missing Data in Each Column The distribution of missing values in the dataset shows that most columns have a low percentage of missing values, with only a few columns having a relatively higher proportion of missing values. There are a few outlier columns with a significantly higher percentage of missing values, making them stand out from the rest of the columns.

Based on a threshold of 25% missing values, the following 6 columns were removed from the dataset:

1. 'AGER_TYP': Indicates the type of the person (categorical)
2. 'GEBURTSJAHR': Year of birth of the person (interval)
3. 'TITEL_KZ': Academic title flag (categorical)
4. 'ALTER_HH': Main age within the household (categorical)
5. 'KK_KUNDENTYP': Customer type of the building society (categorical)
6. 'KBA05_BAUMAX': Most common building type within the microcell (categorical)

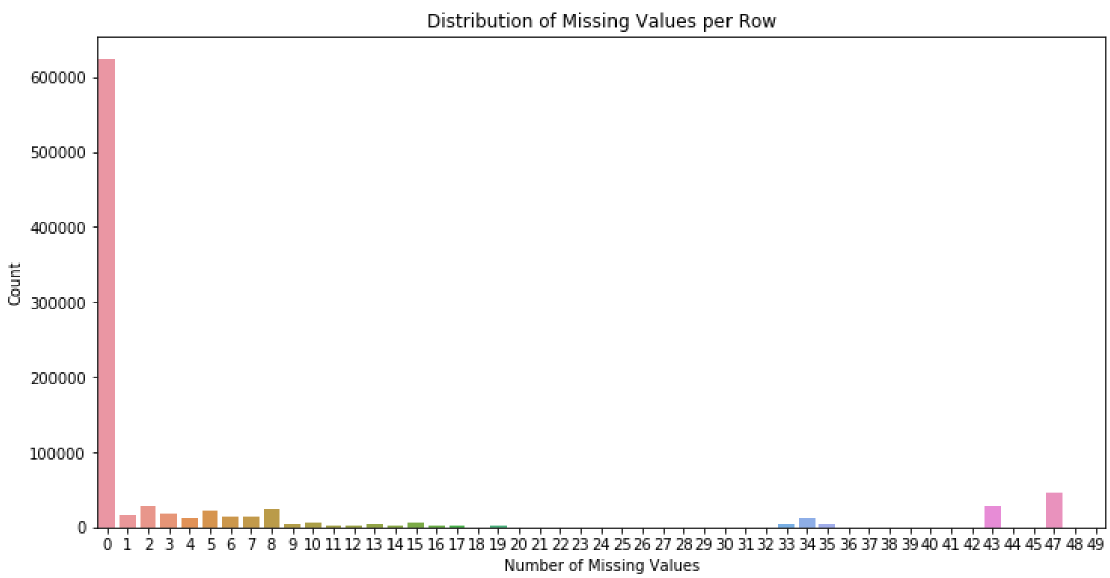
These columns were removed to ensure that the dataset is more consistent and to avoid introducing biases in further analysis. Since these columns contained a significant amount of missing data, they were considered outliers and excluded from the analysis.

Step 1.1.3: Assess Missing Data in Each Row Now, you'll perform a similar assessment for the rows of the dataset. How much data is missing in each row? As with the columns, you should see some groups of points that have a very different numbers of missing values. Divide the data into two subsets: one for data points that are above some threshold for missing values, and a second subset for points below that threshold.

In order to know what to do with the outlier rows, we should see if the distribution of data values on columns that are not missing data (or are missing very little data) are similar or different between the two groups. Select at least five of these columns and compare the distribution of values. - You can use seaborn's `countplot()` function to create a bar chart of code frequencies and matplotlib's `subplot()` function to put bar charts for the two subplots side by side. - To reduce repeated code, you might want to write a function that can perform this comparison, taking as one of its arguments a column to be compared.

Depending on what you observe in your comparison, this will have implications on how you approach your conclusions later in the analysis. If the distributions of non-missing features look similar between the data with many missing values and the data with few or no missing values, then we could argue that simply dropping those points from the analysis won't present a major issue. On the other hand, if the data with many missing values looks very different from the data with few or no missing values, then we should make a note on those data as special. We'll revisit these data later on. **Either way, you should continue your analysis for now using just the subset of the data with few or no missing values.**

In [14]:
```
# How much data is missing in each row of the dataset?

# Calculate the number of missing values in each row
missing_values_row = azdias_cleaned.isnull().sum(axis=1)

# Plot the distribution of missing value counts for rows for visual inspection
plt.figure(figsize=(12, 6))
sns.countplot(missing_values_row)
plt.xlabel('Number of Missing Values')
plt.ylabel('Count')
plt.title('Distribution of Missing Values per Row')
plt.show()
```

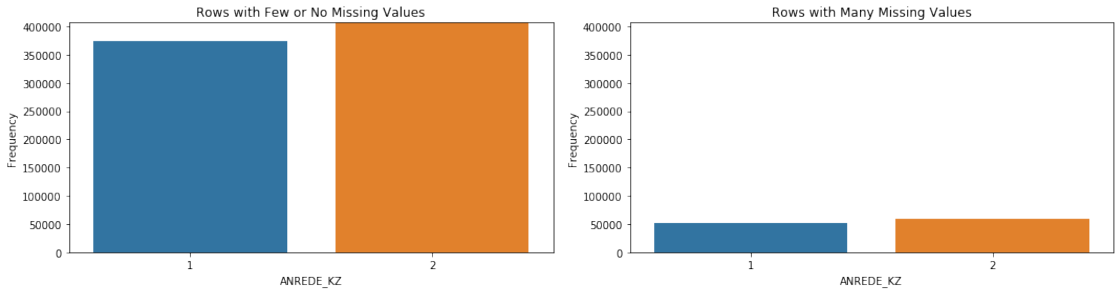


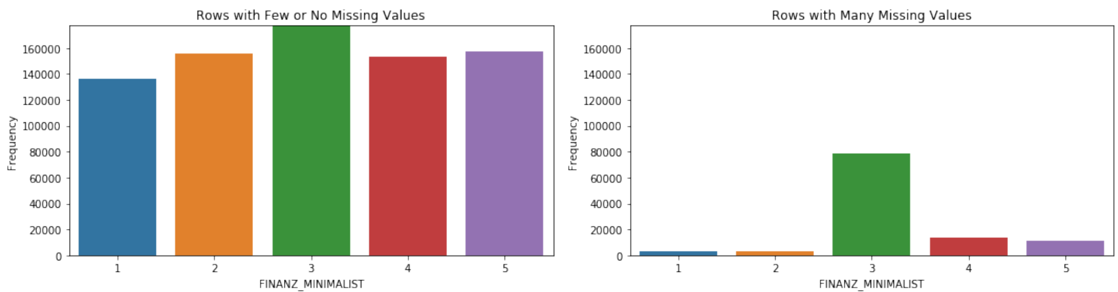
In [15]:
```
# Write code to divide the data into two subsets based on the number of missing
# values in each row.
# Define the threshold to divide the data into subsets
threshold = 10

# Create two subsets based on the threshold
subset_few_missing_values = azdias_cleaned[missing_values_row <= threshold]
subset_many_missing_values = azdias_cleaned[missing_values_row > threshold]

# Confirm the shape of each subset
print("Full set, before the subsets were created shape:", azdias_cleaned.shape)
print("Subset with few or no missing values shape:", subset_few_missing_values.shape)
print("Subset with many missing values shape:", subset_many_missing_values.shape)
```

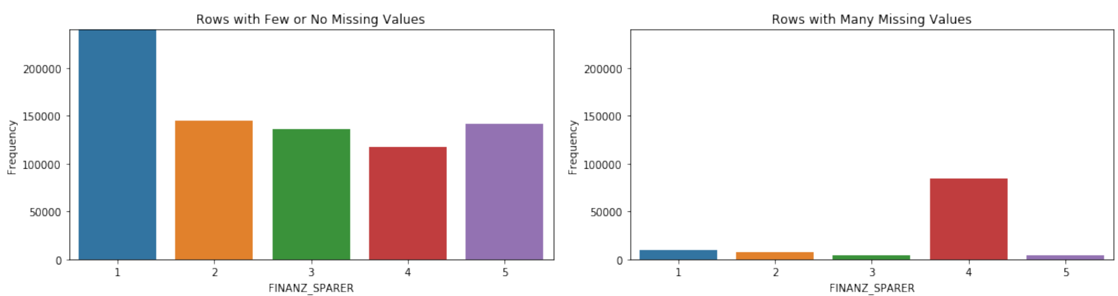
```
Full set, before the subsets were created shape: (891221, 79)
Subset with few or no missing values shape: (780153, 79)
Subset with many missing values shape: (111068, 79)
```

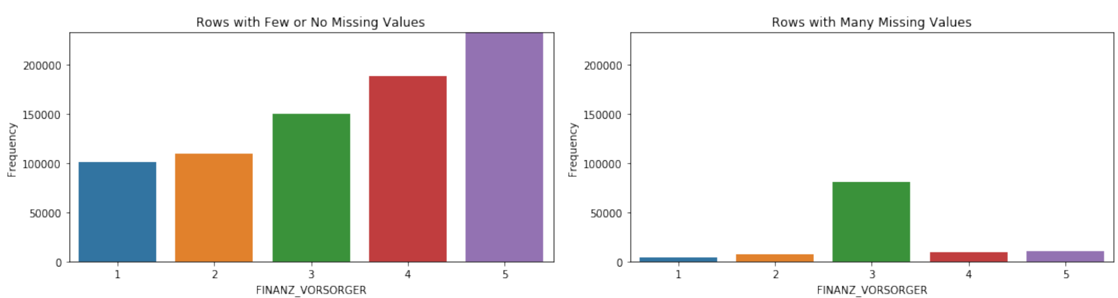
In [16]:
```python
# Compare the distribution of values for at least five columns where there are
# no or few missing values, between the two subsets.
# Referenced: https://stackoverflow.com/questions/44890713/selection-with-loc-in-python
# Referenced:
# Define the threshold for few or no missing values
threshold = 10

# Calculate the number of missing values in each row
missing_values_row = azdias_cleaned.isnull().sum(axis=1)

# Create a new column 'Subset' to indicate whether a row has few or many missing values
azdias_cleaned['Subset'] = 'Few or No Missing'
azdias_cleaned.loc[missing_values_row > threshold, 'Subset'] = 'Many Missing'

# Function to compare the distribution of a column between the two subsets
def compare_column_distribution(rows_below, rows_above, column_name):
    max_y = max(rows_below.value_counts().max(), rows_above.value_counts().max())

    plt.figure(figsize=(15, 4))
    plt.subplot(1, 2, 1)
    plt.title('Rows with Few or No Missing Values')
    sns.countplot(rows_below)
    plt.xlabel(column_name)
    plt.ylabel('Frequency')
    plt.ylim(0, max_y)  # Set the same y-axis limit for both subplots for a better side

    plt.subplot(1, 2, 2)
    plt.title('Rows with Many Missing Values')
    sns.countplot(rows_above)
    plt.xlabel(column_name)
    plt.ylabel('Frequency')
    plt.ylim(0, max_y)  # Set the same y-axis limit for both subplots for a better side

    plt.tight_layout()
    plt.show()

# Select five columns with few or no missing values based on the threshold
few_missing_cols = missing_values_count[missing_values_count <= threshold].index[:5]

# Loop through each column and create a countplot for each subset
for column in few_missing_cols:
    rows_below = azdias_cleaned.loc[azdias_cleaned['Subset'] == 'Few or No Missing', co
    rows_above = azdias_cleaned.loc[azdias_cleaned['Subset'] == 'Many Missing', column]
```

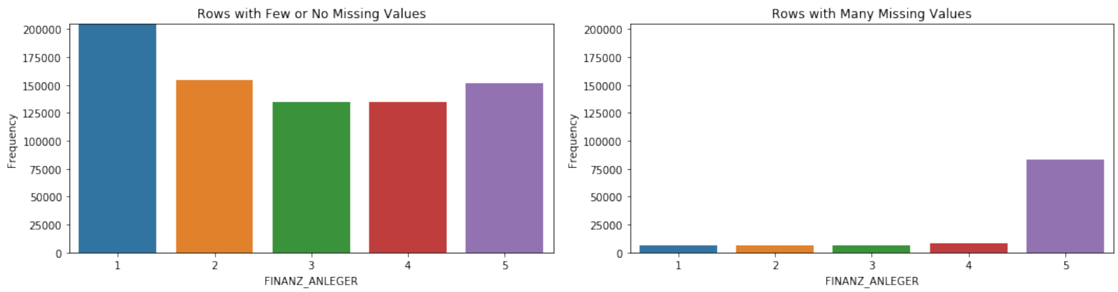
```
compare_column_distribution(rows_below, rows_above, column)
```





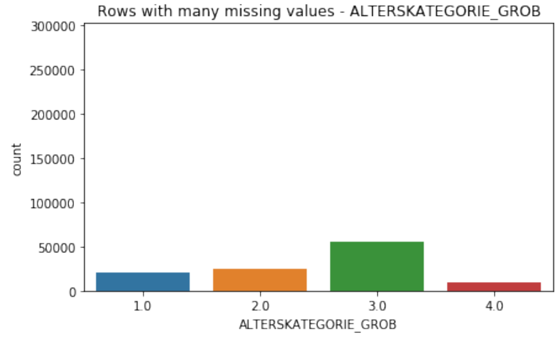


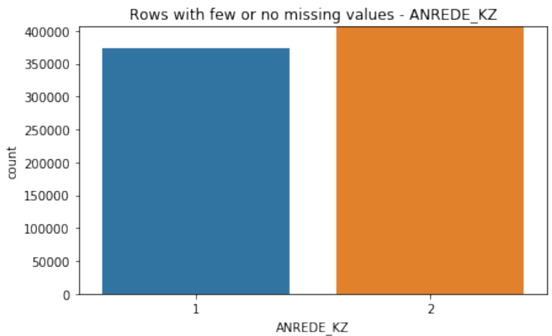


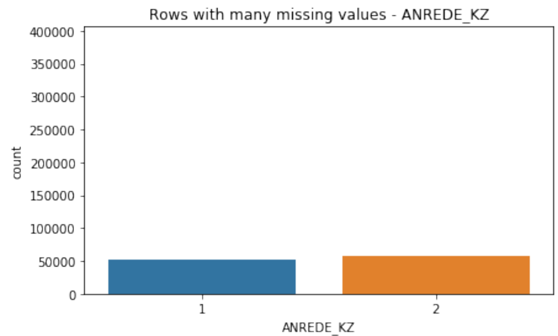


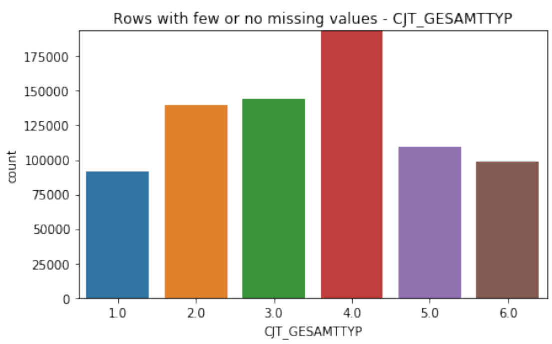
In [17]:
```
# Set a threshold for missing values per row
threshold = 10

# Calculate the number of missing values in each row
missing_values_per_row2 = azdias_cleaned.isnull().sum(axis=1)

# Getting a new percentages variable without AGER_TYP
missing_data_percentages2 = (azdias_cleaned.isnull().sum() / len(azdias_cleaned)) * 100

# Create subsets for data points above and below the threshold
subset_below_threshold2 = azdias_cleaned[missing_values_per_row2 <= threshold]
subset_above_threshold2 = azdias_cleaned[missing_values_per_row2 > threshold]

# Function to compare column distribution between subsets
def compare_column_distribution(rows_below, rows_above, column_name):
    plt.figure(figsize=(15, 4))
    plt.subplot(1, 2, 1)
    plt.title(f'Rows with few or no missing values - {column_name}')
    sns.countplot(rows_below)
    plt.ylim(0, max(rows_below.value_counts().max(), rows_above.value_counts().max()))
    plt.subplot(1, 2, 2)
    plt.title(f'Rows with many missing values - {column_name}')
    sns.countplot(rows_above)
    plt.ylim(0, max(rows_below.value_counts().max(), rows_above.value_counts().max()))
    plt.show()


# Get columns with low missing data percentages
few_missing_vals2 = missing_data_percentages2[missing_data_percentages2 <= 1].index.tol

# Compare the distribution of selected columns between the two subsets
for feature in few_missing_vals2:
    compare_column_distribution(subset_below_threshold2[feature], subset_above_threshol
```

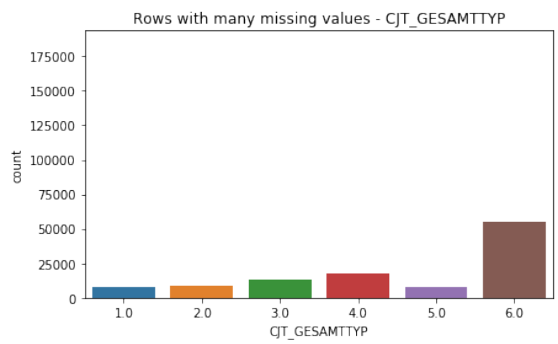

Rows with few or no missing values - ALTERSKATEGORIE_GROB
count
300000
250000
200000
150000
100000
50000
0
1.0
2.0
3.0
4.0
ALTERSKATEGORIE_GROB

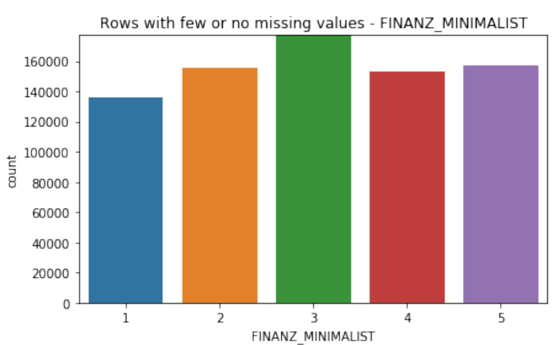

Rows with many missing values - ALTERSKATEGORIE_GROB
count
300000
250000
200000
150000
100000
50000
0
1.0
2.0
3.0
4.0
ALTERSKATEGORIE_GROB

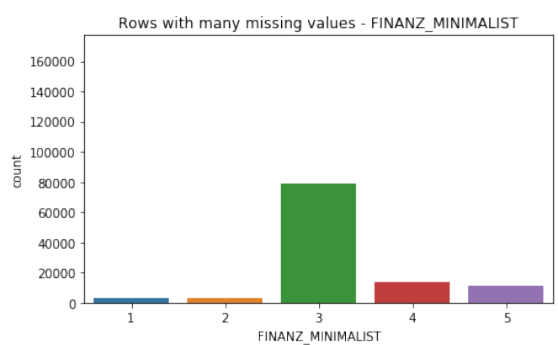

Rows with few or no missing values - ANREDE_KZ
count
400000
350000
300000
250000
200000
150000
100000
50000
0
1
2
ANREDE_KZ

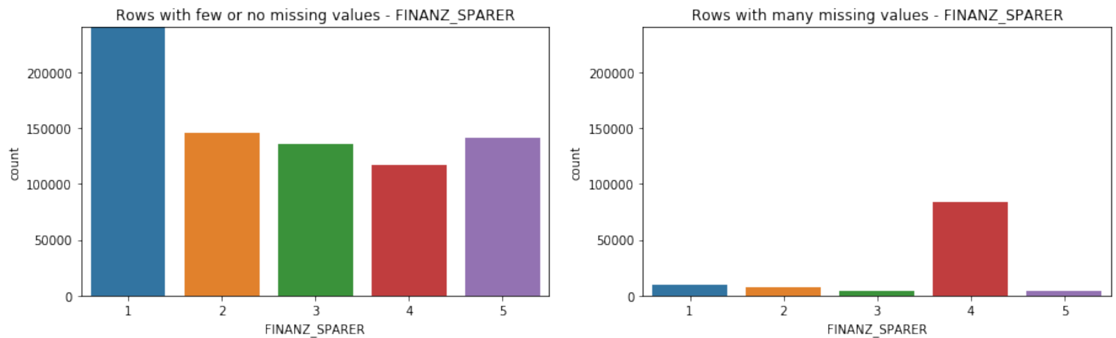

Rows with many missing values - ANREDE_KZ
count
400000
350000
300000
250000
200000
150000
100000
50000
0
1
2
ANREDE_KZ


Rows with few or no missing values - CJT_GESAMTTYP
count
175000
150000
125000
100000
75000
50000
25000
0
1.0
2.0
3.0
4.0
5.0
6.0
CJT_GESAMTTYP


Rows with many missing values - CJT_GESAMTTYP
count
175000
150000
125000
100000
75000
50000
25000
0
1.0
2.0
3.0
4.0
5.0
6.0
CJT_GESAMTTYP


Rows with few or no missing values - FINANZ_MINIMALIST
count
160000
140000
120000
100000
80000
60000
40000
20000
0
1
2
3
4
5
FINANZ_MINIMALIST


Rows with many missing values - FINANZ_MINIMALIST
count
160000
140000
120000
100000
80000
60000
40000
20000
0
1
2
3
4
5
FINANZ_MINIMALIST




```
In [18]: # Referenced: https://medium.com/analytics-vidhya/data-cleaning-dealing-with-missing-va

         azdias_cleaned.drop('Subset', axis=1, inplace=True)
```

Discussion 1.1.3: Assess Missing Data in Each Row Overall, the data with lots of missing values are qualitatively different from the data with few or no missing values, indicating that missingness in the dataset could be important.

This has implications for further analysis, so it will depend upon what types of analysis are being performed as to whether the columns should be dropped. I will maintain a copy of the data without the columns dropped in case I need to use these columns in the future.

1.1.2 Step 1.2: Select and Re-Encode Features

Checking for missing data isn't the only way in which you can prepare a dataset for analysis. Since the unsupervised learning techniques to be used will only work on data that is encoded numerically, you need to make a few encoding changes or additional assumptions to be able to make progress. In addition, while almost all of the values in the dataset are encoded using numbers, not all of them represent numeric values. Check the third column of the feature summary (`feat_info`) for a summary of types of measurement. - For numeric and interval data, these features can be kept without changes. - Most of the variables in the dataset are ordinal in nature. While ordinal values may technically be non-linear in spacing, make the simplifying assumption that the ordinal variables can be treated as being interval in nature (that is, kept without any changes). - Special handling may be necessary for the remaining two variable types: categorical, and 'mixed'.

In the first two parts of this sub-step, you will perform an investigation of the categorical and mixed-type features and make a decision on each of them, whether you will keep, drop, or re-encode each. Then, in the last part, you will create a new data frame with only the selected and engineered columns.

Data wrangling is often the trickiest part of the data analysis process, and there's a lot of it to be done here. But stick with it: once you're done with this step, you'll be ready to get to the machine learning parts of the project!

```
In [19]: # How many features are there of each data type?
         # Count the occurrences of each data type in the feature summary
         data_type_counts = feat_info['type'].value_counts()

         # Display the result
         print(data_type_counts)
```

```
ordinal        49
categorical    21
mixed           7
numeric         7
interval        1
Name: type, dtype: int64
```

Step 1.2.1: Re-Encode Categorical Features For categorical data, you would ordinarily need to encode the levels as dummy variables. Depending on the number of categories, perform one of the following: - For binary (two-level) categoricals that take numeric values, you can keep them without needing to do anything. - There is one binary variable that takes on non-numeric values. For this one, you need to re-encode the values as numbers or create a dummy variable. - For multi-level categoricals (three or more values), you can choose to encode the values using multiple dummy variables (e.g. via OneHotEncoder), or (to keep things straightforward) just drop them from the analysis. As always, document your choices in the Discussion section.

```
In [20]: # Get the list of columns present in azdias_cleaned
         azdias_cleaned_columns = azdias_cleaned.columns.tolist()

         # Filter feat_info to include only the columns that exist in azdias4
         feat_info_filtered = feat_info[feat_info['attribute'].isin(azdias_cleaned_columns)]

         # Display the data types of the filtered columns in feat_info_filtered
         print(feat_info_filtered[['attribute', 'type']])
```

```
                attribute         type
1    ALTERSKATEGORIE_GROB      ordinal
2               ANREDE_KZ  categorical
3           CJT_GESAMTTYP  categorical
4       FINANZ_MINIMALIST      ordinal
5           FINANZ_SPARER      ordinal
6        FINANZ_VORSORGER      ordinal
7          FINANZ_ANLEGER      ordinal
8   FINANZ_UNAUFFAELLIGER      ordinal
9        FINANZ_HAUSBAUER      ordinal
10              FINANZTYP  categorical
12         GFK_URLAUBERTYP  categorical
13        GREEN_AVANTGARDE  categorical
14             HEALTH_TYP      ordinal
15     LP_LEBENSPHASE_FEIN        mixed
```

16	LP_LEBENSPHASE_GROB	mixed
17	LP_FAMILIE_FEIN	categorical
18	LP_FAMILIE_GROB	categorical
19	LP_STATUS_FEIN	categorical
20	LP_STATUS_GROB	categorical
21	NATIONALITAET_KZ	categorical
22	PRAEGENDE_JUGENDJAHRE	mixed
23	RETOURTYP_BK_S	ordinal
24	SEMIO_SOZ	ordinal
25	SEMIO_FAM	ordinal
26	SEMIO_REL	ordinal
27	SEMIO_MAT	ordinal
28	SEMIO_VERT	ordinal
29	SEMIO_LUST	ordinal
30	SEMIO_ERL	ordinal
31	SEMIO_KULT	ordinal
..	...	...
54	MIN_GEBAEUDEJAHR	numeric
55	OST_WEST_KZ	categorical
56	WOHNLAGE	mixed
57	CAMEO_DEUG_2015	categorical
58	CAMEO_DEU_2015	categorical
59	CAMEO_INTL_2015	mixed
60	KBA05_ANTG1	ordinal
61	KBA05_ANTG2	ordinal
62	KBA05_ANTG3	ordinal
63	KBA05_ANTG4	ordinal
65	KBA05_GBZ	ordinal
66	BALLRAUM	ordinal
67	EWDICHTE	ordinal
68	INNENSTADT	ordinal
69	GEBAEUDETYP_RASTER	ordinal
70	KKK	ordinal
71	MOBI_REGIO	ordinal
72	ONLINE_AFFINITAET	ordinal
73	REGIOTYP	ordinal
74	KBA13_ANZAHL_PKW	numeric
75	PLZ8_ANTG1	ordinal
76	PLZ8_ANTG2	ordinal
77	PLZ8_ANTG3	ordinal
78	PLZ8_ANTG4	ordinal
79	PLZ8_BAUMAX	mixed
80	PLZ8_HHZ	ordinal
81	PLZ8_GBZ	ordinal
82	ARBEIT	ordinal
83	ORTSGR_KLS9	ordinal
84	RELAT_AB	ordinal

```
[79 rows x 2 columns]
```

```
In [21]: # Assess categorical variables: which are binary, which are multi-level, and which one

         # Create a list to store the names of the categorical features that need re-encoding
         categorical_features = []

         categorical_features = feat_info_filtered[feat_info_filtered['type'] == 'categorical']
         categorical_data = subset_below_threshold2[categorical_features]
         print(categorical_features)
```

```
2            ANREDE_KZ
3         CJT_GESAMTTYP
10           FINANZTYP
12      GFK_URLAUBERTYP
13     GREEN_AVANTGARDE
17      LP_FAMILIE_FEIN
18      LP_FAMILIE_GROB
19       LP_STATUS_FEIN
20       LP_STATUS_GROB
21     NATIONALITAET_KZ
38          SHOPPER_TYP
39              SOHO_KZ
41             VERS_TYP
42             ZABEOTYP
52          GEBAEUDETYP
55          OST_WEST_KZ
57      CAMEO_DEUG_2015
58       CAMEO_DEU_2015
Name: attribute, dtype: object
```

```
In [22]: # Print unique values in categorical_features
         for item in categorical_features:
             print('Unique values for "{}" are {}'.format(item, azdias_cleaned[item].unique()))
```

```
Unique values for "ANREDE_KZ" are [1 2]
Unique values for "CJT_GESAMTTYP" are [  2.   5.   3.   4.   1.   6.  nan]
Unique values for "FINANZTYP" are [4 1 6 5 2 3]
Unique values for "GFK_URLAUBERTYP" are [ 10.   1.   5.  12.   9.   3.   8.  11.   4.   2.   7.
Unique values for "GREEN_AVANTGARDE" are [0 1]
Unique values for "LP_FAMILIE_FEIN" are [  2.   5.   1.  nan  10.   7.  11.   3.   8.   4.   6.
Unique values for "LP_FAMILIE_GROB" are [  2.   3.   1.  nan   5.   4.]
Unique values for "LP_STATUS_FEIN" are [  1.   2.   3.   9.   4.  10.   5.   8.   6.   7.  nan]
Unique values for "LP_STATUS_GROB" are [  1.   2.   4.   5.   3.  nan]
Unique values for "NATIONALITAET_KZ" are [ nan   1.   3.   2.]
Unique values for "SHOPPER_TYP" are [ nan   3.   2.   1.   0.]
Unique values for "SOHO_KZ" are [ nan   1.   0.]
```

```
Unique values for "VERS_TYP" are [ nan   2.   1.]
Unique values for "ZABEOTYP" are [3 5 4 1 6 2]
Unique values for "GEBAEUDETYP" are [ nan   8.   1.   3.   2.   6.   4.   5.]
Unique values for "OST_WEST_KZ" are [nan 'W' 'O']
Unique values for "CAMEO_DEUG_2015" are [nan '8' '4' '2' '6' '1' '9' '5' '7' '3']
Unique values for "CAMEO_DEU_2015" are [nan '8A' '4C' '2A' '6B' '8C' '4A' '2D' '1A' '1E' '9D' '5
 '9E' '9B' '1B' '3D' '4E' '4B' '3C' '5A' '7B' '9A' '6D' '6E' '2C' '7C' '9C'
 '7D' '5E' '1D' '8D' '6C' '6A' '5B' '4D' '3A' '2B' '7E' '3B' '6F' '5F' '1C']
```

In [23]:
```python
from pandas.api.types import is_string_dtype
# Referenced: https://pandas.pydata.org/docs/reference/api/pandas.api.types.is_string_d

# Filter categorical features that are binary and non-numeric
binary_categorical_features = categorical_data.select_dtypes(include=['object']).column
    categorical_data.select_dtypes(include=['object']).nunique() < 2]

# Filter categorical features that are multi-level or binary and have less than 5 categ
multi_level_categorical_features = categorical_data.select_dtypes(include=['object']).c
    (categorical_data.select_dtypes(include=['object']).nunique() >= 2) &
    (categorical_data.select_dtypes(include=['object']).nunique() < 5)]

# Initialize the list of categorical features to encode
features_to_encode = []

# Print the results
for feature in binary_categorical_features:
    print(f'++ Keeping {feature} has: 2 unique values, type: {categorical_data[feature]

for feature in multi_level_categorical_features:
    print(f'++ Keeping {feature} has: {categorical_data[feature].nunique()} unique valu
    # Check for non-numeric values
    if is_string_dtype(categorical_data[feature]):
        features_to_encode.append(feature)

# Filter categorical features to drop
categorical_features_to_drop = categorical_data.select_dtypes(include=['object']).colum
    ~categorical_data.select_dtypes(include=['object']).columns.isin(binary_categorical
    ~categorical_data.select_dtypes(include=['object']).columns.isin(multi_level_catego

# Print the results for features to drop
for feature in categorical_features_to_drop:
    print(f'-- Dropping {feature} has: {categorical_data[feature].nunique()} unique val
```

```
++ Keeping OST_WEST_KZ has: 2 unique values, type: object with Re-encoding
-- Dropping CAMEO_DEUG_2015 has: 9 unique values, type: object
-- Dropping CAMEO_DEU_2015 has: 44 unique values, type: object
```

```
In [24]: #OST_WEST_KZ needs to be re-encoded. Checking the values

         unique_values = categorical_data['OST_WEST_KZ'].unique()
         print(unique_values)
```

```
['W' 'O']
```

```
In [25]: # Re-encode categorical variable(s) to be kept in the analysis.

         # replace 'O' with 0 and 'W' with 1
         categorical_data['OST_WEST_KZ'] = categorical_data['OST_WEST_KZ'].replace({'O': 0, 'W':

         # Convert to numeric type
         categorical_data['OST_WEST_KZ'] = pd.to_numeric(categorical_data['OST_WEST_KZ'])


         azdias_cleaned['OST_WEST_KZ'] = categorical_data['OST_WEST_KZ']
```

```
In [26]: # Drop the identified categorical features from the azdias_cleaned DataFrame
         azdias_cleaned.drop(['CAMEO_DEUG_2015', 'CAMEO_DEU_2015'], axis=1, inplace=True)

         # Drop the identified categorical features from the feat_info DataFrame
         feat_info.drop(feat_info[feat_info['attribute'].isin(['CAMEO_DEUG_2015', 'CAMEO_DEU_201
```

Discussion 1.2.1: Re-Encode Categorical Features For the categorical features, we kept most of them as they are or treated them as ordinal variables. Only 'OST_WEST_KZ' needed re-encoding because it has two unique values represented as strings 'O' and 'W', which should be transformed to numerical values 0 and 1, respectively. The CAMEO_DEUG_2015 and CAMEO_DEU_2015 columns were removed based upon the specifications.

Step 1.2.2: Engineer Mixed-Type Features There are a handful of features that are marked as "mixed" in the feature summary that require special treatment in order to be included in the analysis. There are two in particular that deserve attention; the handling of the rest are up to your own choices: - "PRAEGENDE_JUGENDJAHRE" combines information on three dimensions: generation by decade, movement (mainstream vs. avantgarde), and nation (east vs. west). While there aren't enough levels to disentangle east from west, you should create two new variables to capture the other two dimensions: an interval-type variable for decade, and a binary variable for movement. - "CAMEO_INTL_2015" combines information on two axes: wealth and life stage. Break up the two-digit codes by their 'tens'-place and 'ones'-place digits into two new ordinal variables (which, for the purposes of this project, is equivalent to just treating them as their raw numeric values). - If you decide to keep or engineer new features around the other mixed-type features, make sure you note your steps in the Discussion section.

Be sure to check `Data_Dictionary.md` for the details needed to finish these tasks.

```
In [27]: # Investigate "PRAEGENDE_JUGENDJAHRE" and engineer two new variables.
         # Read the content of Data_Dictionary.md file
         with open('Data_Dictionary.md', 'r') as file:
```

```python
        data_dictionary_content = file.read()

    # Find the section that contains information about "PRAEGENDE_JUGENDJAHRE"
    start_index = data_dictionary_content.find("### 1.18. PRAEGENDE_JUGENDJAHRE")
    end_index = data_dictionary_content.find("###", start_index + 1)

    # Extract the relevant information
    praegende_jugendjahre_info = data_dictionary_content[start_index:end_index].strip()

    # Print the information
    print(praegende_jugendjahre_info)
```

```
### 1.18. PRAEGENDE_JUGENDJAHRE
Dominating movement of person's youth (avantgarde vs. mainstream; east vs. west)
- -1: unknown
-  0: unknown
-  1: 40s - war years (Mainstream, E+W)
-  2: 40s - reconstruction years (Avantgarde, E+W)
-  3: 50s - economic miracle (Mainstream, E+W)
-  4: 50s - milk bar / Individualisation (Avantgarde, E+W)
-  5: 60s - economic miracle (Mainstream, E+W)
-  6: 60s - generation 68 / student protestors (Avantgarde, W)
-  7: 60s - opponents to the building of the Wall (Avantgarde, E)
-  8: 70s - family orientation (Mainstream, E+W)
-  9: 70s - peace movement (Avantgarde, E+W)
- 10: 80s - Generation Golf (Mainstream, W)
- 11: 80s - ecological awareness (Avantgarde, W)
- 12: 80s - FDJ / communist party youth organisation (Mainstream, E)
- 13: 80s - Swords into ploughshares (Avantgarde, E)
- 14: 90s - digital media kids (Mainstream, E+W)
- 15: 90s - ecological awareness (Avantgarde, E+W)
```

In [28]:
```python
# Investigate "PRAEGENDE_JUGENDJAHRE" and engineer two new variables.
# Read the content of Data_Dictionary.md file
with open('Data_Dictionary.md', 'r') as file:
    data_dictionary_content = file.read()

# Find the section that contains information about "PRAEGENDE_JUGENDJAHRE"
start_index = data_dictionary_content.find("### 1.18. PRAEGENDE_JUGENDJAHRE")
end_index = data_dictionary_content.find("###", start_index + 1)

# Extract the relevant information
praegende_jugendjahre_info = data_dictionary_content[start_index:end_index].strip()

# Define the mapping dictionaries for movement and decade
decade_dict = {1: 0, 3: 0, 5: 0, 8: 0, 10: 0, 12: 0, 14: 0,
                    2: 1, 4: 1, 6: 1, 7: 1, 9: 1, 11: 1, 13: 1, 15: 1}
```

```
movement_dict = {1: 40, 2: 40, 3: 50, 4: 50, 5: 60, 6: 60, 7: 70, 8: 70, 9: 80,
                 10: 80, 11: 80, 12: 80, 13: 80, 14: 90, 15: 90}

# Create decade new column
azdias['PRAEGENDE_JUGENDJAHRE_Decade'] = azdias['PRAEGENDE_JUGENDJAHRE']
azdias['PRAEGENDE_JUGENDJAHRE_Decade'].replace(decade_dict, inplace=True)

# Create movement new column
azdias['PRAEGENDE_JUGENDJAHRE_Movement'] = azdias['PRAEGENDE_JUGENDJAHRE']
azdias['PRAEGENDE_JUGENDJAHRE_Movement'].replace(movement_dict, inplace=True)

# Display the updated DataFrame
azdias_cleaned.head()
```

Out[28]:

```
   ALTERSKATEGORIE_GROB  ANREDE_KZ  CJT_GESAMTTYP  FINANZ_MINIMALIST  \
0                   2.0          1            2.0                  3
1                   1.0          2            5.0                  1
2                   3.0          2            3.0                  1
3                   4.0          2            2.0                  4
4                   3.0          1            5.0                  4

   FINANZ_SPARER  FINANZ_VORSORGER  FINANZ_ANLEGER  FINANZ_UNAUFFAELLIGER  \
0              4                 3               5                      5
1              5                 2               5                      4
2              4                 1               2                      3
3              2                 5               2                      1
4              3                 4               1                      3

   FINANZ_HAUSBAUER  FINANZTYP    ...      PLZ8_ANTG1  PLZ8_ANTG2  PLZ8_ANTG3  \
0                 3          4    ...             NaN         NaN         NaN
1                 5          1    ...             2.0         3.0         2.0
2                 5          1    ...             3.0         3.0         1.0
3                 2          6    ...             2.0         2.0         2.0
4                 2          5    ...             2.0         4.0         2.0

   PLZ8_ANTG4  PLZ8_BAUMAX  PLZ8_HHZ  PLZ8_GBZ  ARBEIT  ORTSGR_KLS9  RELAT_AB
0         NaN          NaN       NaN       NaN     NaN          NaN       NaN
1         1.0          1.0       5.0       4.0     3.0          5.0       4.0
2         0.0          1.0       4.0       4.0     3.0          5.0       2.0
3         0.0          1.0       3.0       4.0     2.0          3.0       3.0
4         1.0          2.0       3.0       3.0     4.0          6.0       5.0

[5 rows x 77 columns]
```

In [29]:

```
azdias_cleaned.drop(columns=['PRAEGENDE_JUGENDJAHRE'], inplace=True)
```

In [30]:

```
# Investigate "CAMEO_INTL_2015" and engineer two new variables.
# Read the content of Data_Dictionary.md file
```

```
with open('Data_Dictionary.md', 'r') as file:
    data_dictionary_content = file.read()

# Find the section that contains information about "CAMEO_INTL_2015"
start_index = data_dictionary_content.find("### 4.3. CAMEO_INTL_2015")
end_index = data_dictionary_content.find("###", start_index + 1)

# Extract the relevant information
cameo_intl_2015_info = data_dictionary_content[start_index:end_index].strip()

# Print the information
print(cameo_intl_2015_info)
```

```
### 4.3. CAMEO_INTL_2015
German CAMEO: Wealth / Life Stage Typology, mapped to international code
- -1: unknown
- 11: Wealthy Households - Pre-Family Couples & Singles
- 12: Wealthy Households - Young Couples With Children
- 13: Wealthy Households - Families With School Age Children
- 14: Wealthy Households - Older Families &  Mature Couples
- 15: Wealthy Households - Elders In Retirement
- 21: Prosperous Households - Pre-Family Couples & Singles
- 22: Prosperous Households - Young Couples With Children
- 23: Prosperous Households - Families With School Age Children
- 24: Prosperous Households - Older Families & Mature Couples
- 25: Prosperous Households - Elders In Retirement
- 31: Comfortable Households - Pre-Family Couples & Singles
- 32: Comfortable Households - Young Couples With Children
- 33: Comfortable Households - Families With School Age Children
- 34: Comfortable Households - Older Families & Mature Couples
- 35: Comfortable Households - Elders In Retirement
- 41: Less Affluent Households - Pre-Family Couples & Singles
- 42: Less Affluent Households - Young Couples With Children
- 43: Less Affluent Households - Families With School Age Children
- 44: Less Affluent Households - Older Families & Mature Couples
- 45: Less Affluent Households - Elders In Retirement
- 51: Poorer Households - Pre-Family Couples & Singles
- 52: Poorer Households - Young Couples With Children
- 53: Poorer Households - Families With School Age Children
- 54: Poorer Households - Older Families & Mature Couples
- 55: Poorer Households - Elders In Retirement
- XX: unknown

-----

## 5. RR3 micro-cell features
```

In [31]:
```
# finally got this to work after referencing this article: https://www.analyticsvidhya.
```

```python
# Create new wealth dimension using the 10's digit from CAMEO_INTL_2015. Use lamba func
azdias_cleaned['CAMEO_INTL_2015_wealth'] = azdias_cleaned['CAMEO_INTL_2015'].apply(lamb

# Create new life_stage dimension using the 1's digit from CAMEO_INTL_2015. Use lamba f
azdias_cleaned['CAMEO_INTL_2015_life_stage'] = azdias_cleaned['CAMEO_INTL_2015'].apply(

# Drop CAMEO_INTL_2015 column
azdias_cleaned.drop('CAMEO_INTL_2015', axis=1, inplace=True)

azdias_cleaned.head()
```

Out[31]:

```
   ALTERSKATEGORIE_GROB  ANREDE_KZ  CJT_GESAMTTYP  FINANZ_MINIMALIST  \
0                   2.0          1            2.0                  3
1                   1.0          2            5.0                  1
2                   3.0          2            3.0                  1
3                   4.0          2            2.0                  4
4                   3.0          1            5.0                  4

   FINANZ_SPARER  FINANZ_VORSORGER  FINANZ_ANLEGER  FINANZ_UNAUFFAELLIGER  \
0              4                 3               5                      5
1              5                 2               5                      4
2              4                 1               2                      3
3              2                 5               2                      1
4              3                 4               1                      3

   FINANZ_HAUSBAUER  FINANZTYP             ...               PLZ8_ANTG3  \
0                 3          4             ...                      NaN
1                 5          1             ...                      2.0
2                 5          1             ...                      1.0
3                 2          6             ...                      2.0
4                 2          5             ...                      2.0

   PLZ8_ANTG4  PLZ8_BAUMAX  PLZ8_HHZ  PLZ8_GBZ  ARBEIT  ORTSGR_KLS9  RELAT_AB  \
0         NaN          NaN       NaN       NaN     NaN          NaN       NaN
1         1.0          1.0       5.0       4.0     3.0          5.0       4.0
2         0.0          1.0       4.0       4.0     3.0          5.0       2.0
3         0.0          1.0       3.0       4.0     2.0          3.0       3.0
4         1.0          2.0       3.0       3.0     4.0          6.0       5.0

   CAMEO_INTL_2015_wealth  CAMEO_INTL_2015_life_stage
0                     NaN                         NaN
1                     5.0                         1.0
2                     2.0                         4.0
3                     1.0                         2.0
4                     4.0                         3.0

[5 rows x 77 columns]
```

Discussion 1.2.2: Engineer Mixed-Type Features Decisions needed to be made on whether to keep or drop mixed-value features based on their meaningfulness and the possibility of engineering new informative variables from them. For some features, new variables were engineered. The engineering of new variables allowed us to represent the mixed-value features in a more structured manner, making them suitable for further analysis.

I created two dictionaries for the movement and decade found in PRAEGENDE_JUGENDJAHRE. This information was used to create two new columns new columns were created - PRAEGENDE_JUGENDJAHRE_Decade and PRAEGENDE_JUGENDJAHRE_Movement.

For CAMEO_INTL_2015, instead of using dictionaries, two new columns CAMEO_INTL_2015_wealth and CAMEO_INTL_2015_life_stage were created by utilizing a lambda function to split the ones and tens place into separate entities.

The PRAEGENDE_JUGENDJAHRE and CAMEO_INTL_2015 columns were dropped.

Step 1.2.3: Complete Feature Selection In order to finish this step up, you need to make sure that your data frame now only has the columns that you want to keep. To summarize, the dataframe should consist of the following: - All numeric, interval, and ordinal type columns from the original dataset. - Binary categorical features (all numerically-encoded). - Engineered features from other multi-level categorical features and mixed features.

Make sure that for any new columns that you have engineered, that you've excluded the original columns from the final dataset. Otherwise, their values will interfere with the analysis later on the project. For example, you should not keep "PRAEGENDE_JUGENDJAHRE", since its values won't be useful for the algorithm: only the values derived from it in the engineered features you created should be retained. As a reminder, your data should only be from **the subset with few or no missing values**.

In [32]:
```
# If there are other re-engineering tasks you need to perform, make sure you
# take care of them here. (Dealing with missing data will come in step 2.1.)
#Take a look at all the datatypes
azdias_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891221 entries, 0 to 891220
Data columns (total 77 columns):
ALTERSKATEGORIE_GROB          888340 non-null float64
ANREDE_KZ                     891221 non-null int64
CJT_GESAMTTYP                 886367 non-null float64
FINANZ_MINIMALIST             891221 non-null int64
FINANZ_SPARER                 891221 non-null int64
FINANZ_VORSORGER              891221 non-null int64
FINANZ_ANLEGER                891221 non-null int64
FINANZ_UNAUFFAELLIGER         891221 non-null int64
FINANZ_HAUSBAUER              891221 non-null int64
FINANZTYP                     891221 non-null int64
GFK_URLAUBERTYP               886367 non-null float64
GREEN_AVANTGARDE              891221 non-null int64
HEALTH_TYP                    780025 non-null float64
LP_LEBENSPHASE_FEIN           793589 non-null float64
```

```
LP_LEBENSPHASE_GROB          796649 non-null float64
LP_FAMILIE_FEIN              813429 non-null float64
LP_FAMILIE_GROB              813429 non-null float64
LP_STATUS_FEIN               886367 non-null float64
LP_STATUS_GROB               886367 non-null float64
NATIONALITAET_KZ             782906 non-null float64
RETOURTYP_BK_S               886367 non-null float64
SEMIO_SOZ                    891221 non-null int64
SEMIO_FAM                    891221 non-null int64
SEMIO_REL                    891221 non-null int64
SEMIO_MAT                    891221 non-null int64
SEMIO_VERT                   891221 non-null int64
SEMIO_LUST                   891221 non-null int64
SEMIO_ERL                    891221 non-null int64
SEMIO_KULT                   891221 non-null int64
SEMIO_RAT                    891221 non-null int64
SEMIO_KRIT                   891221 non-null int64
SEMIO_DOM                    891221 non-null int64
SEMIO_KAEM                   891221 non-null int64
SEMIO_PFLICHT                891221 non-null int64
SEMIO_TRADV                  891221 non-null int64
SHOPPER_TYP                  780025 non-null float64
SOHO_KZ                      817722 non-null float64
VERS_TYP                     780025 non-null float64
ZABEOTYP                     891221 non-null int64
ANZ_PERSONEN                 817722 non-null float64
ANZ_TITEL                    817722 non-null float64
HH_EINKOMMEN_SCORE           872873 non-null float64
W_KEIT_KIND_HH               743233 non-null float64
WOHNDAUER_2008               817722 non-null float64
ANZ_HAUSHALTE_AKTIV          791610 non-null float64
ANZ_HH_TITEL                 794213 non-null float64
GEBAEUDETYP                  798073 non-null float64
KONSUMNAEHE                  817252 non-null float64
MIN_GEBAEUDEJAHR             798073 non-null float64
OST_WEST_KZ                  780153 non-null float64
WOHNLAGE                     798073 non-null float64
KBA05_ANTG1                  757897 non-null float64
KBA05_ANTG2                  757897 non-null float64
KBA05_ANTG3                  757897 non-null float64
KBA05_ANTG4                  757897 non-null float64
KBA05_GBZ                    757897 non-null float64
BALLRAUM                     797481 non-null float64
EWDICHTE                     797481 non-null float64
INNENSTADT                   797481 non-null float64
GEBAEUDETYP_RASTER           798066 non-null float64
KKK                          733157 non-null float64
MOBI_REGIO                   757897 non-null float64
```

```
ONLINE_AFFINITAET              886367 non-null float64
REGIOTYP                       733157 non-null float64
KBA13_ANZAHL_PKW               785421 non-null float64
PLZ8_ANTG1                     774706 non-null float64
PLZ8_ANTG2                     774706 non-null float64
PLZ8_ANTG3                     774706 non-null float64
PLZ8_ANTG4                     774706 non-null float64
PLZ8_BAUMAX                    774706 non-null float64
PLZ8_HHZ                       774706 non-null float64
PLZ8_GBZ                       774706 non-null float64
ARBEIT                         793846 non-null float64
ORTSGR_KLS9                    793947 non-null float64
RELAT_AB                       793846 non-null float64
CAMEO_INTL_2015_wealth         791869 non-null float64
CAMEO_INTL_2015_life_stage     791869 non-null float64
dtypes: float64(53), int64(24)
memory usage: 523.6 MB
```

In [33]:
```
# Do whatever you need to in order to ensure that the dataframe only contains
# the columns that should be passed to the algorithm functions.
mixed_features = feat_info.loc[feat_info['type'] == 'mixed', 'attribute']
mixed_features = mixed_features[mixed_features.isin(azdias_cleaned.columns)]

azdias_cleaned.drop(columns = mixed_features, axis=1, inplace=True)
azdias_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891221 entries, 0 to 891220
Data columns (total 73 columns):
ALTERSKATEGORIE_GROB           888340 non-null float64
ANREDE_KZ                      891221 non-null int64
CJT_GESAMTTYP                  886367 non-null float64
FINANZ_MINIMALIST              891221 non-null int64
FINANZ_SPARER                  891221 non-null int64
FINANZ_VORSORGER               891221 non-null int64
FINANZ_ANLEGER                 891221 non-null int64
FINANZ_UNAUFFAELLIGER          891221 non-null int64
FINANZ_HAUSBAUER               891221 non-null int64
FINANZTYP                      891221 non-null int64
GFK_URLAUBERTYP                886367 non-null float64
GREEN_AVANTGARDE               891221 non-null int64
HEALTH_TYP                     780025 non-null float64
LP_FAMILIE_FEIN                813429 non-null float64
LP_FAMILIE_GROB                813429 non-null float64
LP_STATUS_FEIN                 886367 non-null float64
LP_STATUS_GROB                 886367 non-null float64
NATIONALITAET_KZ               782906 non-null float64
```

```
RETOURTYP_BK_S          886367 non-null float64
SEMIO_SOZ               891221 non-null int64
SEMIO_FAM               891221 non-null int64
SEMIO_REL               891221 non-null int64
SEMIO_MAT               891221 non-null int64
SEMIO_VERT              891221 non-null int64
SEMIO_LUST              891221 non-null int64
SEMIO_ERL               891221 non-null int64
SEMIO_KULT              891221 non-null int64
SEMIO_RAT               891221 non-null int64
SEMIO_KRIT              891221 non-null int64
SEMIO_DOM               891221 non-null int64
SEMIO_KAEM              891221 non-null int64
SEMIO_PFLICHT           891221 non-null int64
SEMIO_TRADV             891221 non-null int64
SHOPPER_TYP             780025 non-null float64
SOHO_KZ                 817722 non-null float64
VERS_TYP                780025 non-null float64
ZABEOTYP                891221 non-null int64
ANZ_PERSONEN            817722 non-null float64
ANZ_TITEL               817722 non-null float64
HH_EINKOMMEN_SCORE      872873 non-null float64
W_KEIT_KIND_HH          743233 non-null float64
WOHNDAUER_2008          817722 non-null float64
ANZ_HAUSHALTE_AKTIV     791610 non-null float64
ANZ_HH_TITEL            794213 non-null float64
GEBAEUDETYP             798073 non-null float64
KONSUMNAEHE             817252 non-null float64
MIN_GEBAEUDEJAHR        798073 non-null float64
OST_WEST_KZ             780153 non-null float64
KBA05_ANTG1             757897 non-null float64
KBA05_ANTG2             757897 non-null float64
KBA05_ANTG3             757897 non-null float64
KBA05_ANTG4             757897 non-null float64
KBA05_GBZ               757897 non-null float64
BALLRAUM                797481 non-null float64
EWDICHTE                797481 non-null float64
INNENSTADT              797481 non-null float64
GEBAEUDETYP_RASTER      798066 non-null float64
KKK                     733157 non-null float64
MOBI_REGIO              757897 non-null float64
ONLINE_AFFINITAET       886367 non-null float64
REGIOTYP                733157 non-null float64
KBA13_ANZAHL_PKW        785421 non-null float64
PLZ8_ANTG1              774706 non-null float64
PLZ8_ANTG2              774706 non-null float64
PLZ8_ANTG3              774706 non-null float64
PLZ8_ANTG4              774706 non-null float64
```

```
PLZ8_HHZ                       774706 non-null float64
PLZ8_GBZ                       774706 non-null float64
ARBEIT                         793846 non-null float64
ORTSGR_KLS9                    793947 non-null float64
RELAT_AB                       793846 non-null float64
CAMEO_INTL_2015_wealth         791869 non-null float64
CAMEO_INTL_2015_life_stage     791869 non-null float64
dtypes: float64(49), int64(24)
memory usage: 496.4 MB
```

1.1.3 Step 1.3: Create a Cleaning Function

Even though you've finished cleaning up the general population demographics data, it's important to look ahead to the future and realize that you'll need to perform the same cleaning steps on the customer demographics data. In this substep, complete the function below to execute the main feature selection, encoding, and re-engineering steps you performed above. Then, when it comes to looking at the customer data in Step 3, you can just run this function on that DataFrame to get the trimmed dataset in a single step.

```
In [34]: def clean_data(df):
             """
             Perform feature trimming, re-encoding, and engineering for demographics
             data

             INPUT: Demographics DataFrame
             OUTPUT: Trimmed and cleaned demographics DataFrame
             """
             #read in the feat_info file
             feat_info =  pd.read_csv('AZDIAS_Feature_Summary.csv',sep=';')

             # Put in code here to execute all main cleaning steps:

             # Convert missing value codes into NaNs
             for index, row in feat_info.iterrows():
                 missing_or_unknown_str = row['missing_or_unknown']
                 missing_or_unknown_list = missing_or_unknown_str.strip('[]').split(',')
                 missing_or_unknown_list = [int(value) if value not in ['X', 'XX', ''] else valu

                 if missing_or_unknown_list != ['']:
                     attribute = row['attribute']
                     if attribute in df.columns:
                         df[attribute] = df[attribute].replace(missing_or_unknown_list, np.nan)
             print('missing removed')

             # Remove selected columns and rows
             columns_to_drop = ['CAMEO_DEUG_2015', 'CAMEO_DEU_2015', 'AGER_TYP', 'GEBURTSJAHR',
                                'ALTER_HH', 'KK_KUNDENTYP', 'KBA05_BAUMAX', 'LP_LEBENSPHASE_FEIN
```

```
                     'LP_LEBENSPHASE_GROB', 'PLZ8_BAUMAX', 'WOHNLAGE'] #columns
    columns_to_drop_existing = [col for col in columns_to_drop if col in df.columns]
    df.drop(columns_to_drop_existing, axis=1, inplace=True)
    print('column drop successful')

    # Print the number of rows before dropping columns and rows
    print("Number of rows before dropping columns and rows:", df.shape[0])
    df = df[df.isnull().sum(axis=1) <= 10] #rows
    # Print the number of rows after filtering rows
    print("Number of rows after filtering rows:", df.shape[0])

    # Select, re-encode, and engineer column values
    #re-encoding
    # replace 'O' with 0 and 'W' with 1
    df['OST_WEST_KZ'] = df['OST_WEST_KZ'].replace({'O': 0, 'W': 1})
    df['OST_WEST_KZ'] = pd.to_numeric(df['OST_WEST_KZ'])

    #engineering
    # Engineer "PRAEGENDE_JUGENDJAHRE" features
    decade_dict = {1:1, 2:1, 3:2, 4:2, 5:3, 6:3, 7:3, 8:4, 9:4, 10:5, 11:5, 12:5, 13:5,
    movement_dict = {1:1, 2:0, 3:1, 4:0, 5:1, 6:0, 7:0, 8:1, 9:0, 10:1, 11:0, 12:1, 13:
    df['DECADE'] = df['PRAEGENDE_JUGENDJAHRE']
    df['MOVEMENT'] = df['PRAEGENDE_JUGENDJAHRE']
    df['DECADE'].replace(decade_dict, inplace=True)
    df['MOVEMENT'].replace(movement_dict, inplace=True)
    print('Step 1 engineering complete.')


    # Engineer "CAMEO_INTL_2015" features using the lambda function
    df['CAMEO_INTL_2015_wealth'] = df['CAMEO_INTL_2015'].apply(lambda x: x if pd.isnull
    df['CAMEO_INTL_2015_life_stage'] = df['CAMEO_INTL_2015'].apply(lambda x: x if pd.is
    print('Step 2 engineering complete.')

    # Drop CAMEO_INTL_2015 column
    df.drop('CAMEO_INTL_2015', axis=1, inplace=True)
    df.drop('PRAEGENDE_JUGENDJAHRE', axis=1, inplace=True)
    print('Columns re-engineered and dropped.')

    # Drop other mixed variables
    mixed_features = feat_info.loc[feat_info['type'] == 'mixed', 'attribute']
    mixed_features = mixed_features[mixed_features.isin(df.columns)]
    df = df.drop(columns=mixed_features)
    print('Mixed variables dropped.')

    # Return the cleaned dataframe.
    return df
```

```
In [35]: #azdias.head()
```

```
In [36]: #test clean_data function

         #print('Checking function:')
         #clean_data(azdias)
```

```
In [37]: azdias_cleaned_copy = pd.DataFrame(azdias_cleaned.values, index=azdias.index, columns=a
```

1.2 Step 2: Feature Transformation

1.2.1 Step 2.1: Apply Feature Scaling

Before we apply dimensionality reduction techniques to the data, we need to perform feature scaling so that the principal component vectors are not influenced by the natural differences in scale for features. Starting from this part of the project, you'll want to keep an eye on the API reference page for sklearn to help you navigate to all of the classes and functions that you'll need. In this substep, you'll need to check the following:

- sklearn requires that data not have missing values in order for its estimators to work properly. So, before applying the scaler to your data, make sure that you've cleaned the DataFrame of the remaining missing values. This can be as simple as just removing all data points with missing data, or applying an Imputer to replace all missing values. You might also try a more complicated procedure where you temporarily remove missing values in order to compute the scaling parameters before re-introducing those missing values and applying imputation. Think about how much missing data you have and what possible effects each approach might have on your analysis, and justify your decision in the discussion section below.
- For the actual scaling function, a StandardScaler instance is suggested, scaling each feature to mean 0 and standard deviation 1.
- For these classes, you can make use of the `.fit_transform()` method to both fit a procedure to the data as well as apply the transformation to the data at the same time. Don't forget to keep the fit sklearn objects handy, since you'll be applying them to the customer demographics data towards the end of the project.

```
In [38]: # If you've not yet cleaned the dataset of all NaN values, then investigate and
         # do that now.
         # Referenced: https://www.kdnuggets.com/2022/07/scikitlearn-imputer.html

         # Update the column names
         azdias_cleaned_columns = azdias_cleaned_copy.columns.tolist()
         imputer = Imputer(strategy='mean')

         # Fit and transform
         azdias_cleaned_copy = imputer.fit_transform(azdias_cleaned_copy)

         azdias_cleaned_copy = pd.DataFrame(azdias_cleaned_copy, columns=azdias_cleaned_columns)

         # Check number of NaN values
         print('NaNs after imputer:', np.isnan(azdias_cleaned_copy).sum())
```

```
NaNs after imputer: ALTERSKATEGORIE_GROB          0
ANREDE_KZ                    0
CJT_GESAMTTYP                0
FINANZ_MINIMALIST            0
FINANZ_SPARER                0
FINANZ_VORSORGER             0
FINANZ_ANLEGER               0
FINANZ_UNAUFFAELLIGER        0
FINANZ_HAUSBAUER             0
FINANZTYP                    0
GFK_URLAUBERTYP              0
GREEN_AVANTGARDE             0
HEALTH_TYP                   0
LP_FAMILIE_FEIN              0
LP_FAMILIE_GROB              0
LP_STATUS_FEIN               0
LP_STATUS_GROB               0
NATIONALITAET_KZ             0
RETOURTYP_BK_S               0
SEMIO_SOZ                    0
SEMIO_FAM                    0
SEMIO_REL                    0
SEMIO_MAT                    0
SEMIO_VERT                   0
SEMIO_LUST                   0
SEMIO_ERL                    0
SEMIO_KULT                   0
SEMIO_RAT                    0
SEMIO_KRIT                   0
SEMIO_DOM                    0
                            ..
ANZ_HH_TITEL                 0
GEBAEUDETYP                  0
KONSUMNAEHE                  0
MIN_GEBAEUDEJAHR             0
OST_WEST_KZ                  0
KBA05_ANTG1                  0
KBA05_ANTG2                  0
KBA05_ANTG3                  0
KBA05_ANTG4                  0
KBA05_GBZ                    0
BALLRAUM                     0
EWDICHTE                     0
INNENSTADT                   0
GEBAEUDETYP_RASTER           0
KKK                          0
MOBI_REGIO                   0
ONLINE_AFFINITAET            0
```

```
REGIOTYP                       0
KBA13_ANZAHL_PKW               0
PLZ8_ANTG1                     0
PLZ8_ANTG2                     0
PLZ8_ANTG3                     0
PLZ8_ANTG4                     0
PLZ8_HHZ                       0
PLZ8_GBZ                       0
ARBEIT                         0
ORTSGR_KLS9                    0
RELAT_AB                       0
CAMEO_INTL_2015_wealth         0
CAMEO_INTL_2015_life_stage     0
Length: 73, dtype: int64
```

In [39]:
```python
# Apply feature scaling to the general population demographics data.
# Scaler
scaler = StandardScaler()

# Fit and transform
azdias_scaled = scaler.fit_transform(azdias_cleaned_copy)

# Convert back to a DataFrame
azdias_scaled = pd.DataFrame(azdias_scaled, columns=azdias_cleaned_columns)
```

In [40]:
```python
azdias_scaled
```

Out[40]:

	ALTERSKATEGORIE_GROB	ANREDE_KZ	CJT_GESAMTTYP	FINANZ_MINIMALIST \
0	-0.750972	-1.045218	-1.026509	-0.056416
1	-1.742724	0.956738	0.859488	-1.570358
2	0.240781	0.956738	-0.397844	-1.570358
3	1.232533	0.956738	-1.026509	0.700556
4	0.240781	-1.045218	0.859488	0.700556
5	-1.742724	0.956738	-1.026509	-0.056416
6	-0.750972	0.956738	0.859488	-1.570358
7	-1.742724	-1.045218	-0.397844	-0.056416
8	0.240781	-1.045218	-0.397844	0.700556
9	0.240781	0.956738	0.230822	-0.813387
10	0.240781	0.956738	-1.655175	-0.813387
11	-0.750972	-1.045218	1.488153	-0.056416
12	0.240781	-1.045218	1.488153	1.457527
13	-1.742724	0.956738	0.859488	-1.570358
14	0.240781	-1.045218	1.488153	-0.056416
15	1.232533	0.956738	0.230822	0.700556
16	-1.742724	0.956738	-1.655175	0.700556
17	-0.750972	-1.045218	1.488153	-0.056416
18	-0.750972	0.956738	1.488153	-0.813387

19	0.240781	-1.045218	-0.397844	1.457527
20	-0.750972	0.956738	0.230822	0.700556
21	-0.750972	-1.045218	-0.397844	-0.056416
22	-1.742724	-1.045218	0.230822	-1.570358
23	0.240781	-1.045218	-0.397844	1.457527
24	0.240781	0.956738	1.488153	-0.056416
25	-1.742724	-1.045218	-0.397844	-0.056416
26	0.240781	-1.045218	-0.397844	1.457527
27	0.240781	-1.045218	0.230822	-0.056416
28	0.240781	-1.045218	-1.026509	-0.056416
29	1.232533	0.956738	-1.655175	1.457527
...	...	...	...	...
891191	1.232533	0.956738	-1.655175	0.700556
891192	-1.742724	0.956738	-0.397844	-1.570358
891193	1.232533	-1.045218	-0.397844	0.700556
891194	0.240781	-1.045218	0.230822	0.700556
891195	1.232533	0.956738	1.488153	-0.056416
891196	-0.750972	0.956738	1.488153	-1.570358
891197	0.240781	0.956738	-1.655175	-0.056416
891198	0.240781	-1.045218	0.859488	-0.813387
891199	-0.750972	-1.045218	-0.397844	-0.813387
891200	-1.742724	0.956738	-0.397844	-1.570358
891201	0.240781	-1.045218	-0.397844	0.700556
891202	-0.750972	0.956738	0.859488	-1.570358
891203	1.232533	0.956738	-1.655175	0.700556
891204	0.240781	-1.045218	0.859488	0.700556
891205	1.232533	-1.045218	-1.026509	0.700556
891206	-1.742724	0.956738	0.230822	-0.056416
891207	0.240781	0.956738	-1.655175	1.457527
891208	1.232533	-1.045218	-1.026509	1.457527
891209	-1.742724	0.956738	0.859488	-1.570358
891210	0.240781	-1.045218	0.859488	-0.056416
891211	0.240781	-1.045218	-1.026509	-0.056416
891212	1.232533	-1.045218	-1.655175	-0.056416
891213	1.232533	0.956738	0.859488	-0.056416
891214	-1.742724	0.956738	0.230822	-1.570358
891215	-0.750972	0.956738	1.488153	-1.570358
891216	0.240781	0.956738	0.859488	-1.570358
891217	-0.750972	-1.045218	0.230822	-0.056416
891218	-0.750972	0.956738	0.230822	-0.813387
891219	-1.742724	-1.045218	-0.397844	-1.570358
891220	1.232533	-1.045218	-1.655175	0.700556

	FINANZ_SPARER	FINANZ_VORSORGER	FINANZ_ANLEGER	\
0	0.804890	-0.303378	1.285741	
1	1.487601	-1.059731	1.285741	
2	0.804890	-1.816084	-0.675554	
3	-0.560532	1.209329	-0.675554	

4	0.122179	0.452976	-1.329319
5	-1.243244	1.209329	-0.675554
6	1.487601	-1.816084	1.285741
7	0.122179	0.452976	-1.329319
8	0.804890	-1.059731	0.631976
9	0.804890	-1.059731	-0.021789
10	-0.560532	1.209329	-0.021789
11	0.804890	-0.303378	1.285741
12	0.122179	0.452976	-0.675554
13	0.804890	-0.303378	1.285741
14	0.804890	-0.303378	1.285741
15	-1.243244	1.209329	-1.329319
16	0.122179	-1.816084	0.631976
17	0.804890	-0.303378	1.285741
18	0.804890	-1.816084	1.285741
19	-0.560532	-0.303378	-1.329319
20	0.122179	-1.816084	0.631976
21	0.804890	-1.816084	-0.675554
22	1.487601	-0.303378	1.285741
23	0.122179	-0.303378	-0.675554
24	0.804890	-0.303378	1.285741
25	1.487601	-0.303378	1.285741
26	-0.560532	0.452976	-0.675554
27	0.122179	0.452976	-1.329319
28	-0.560532	0.452976	-0.021789
29	-1.243244	1.209329	-0.675554
...	...	...	...
891191	-1.243244	1.209329	-1.329319
891192	1.487601	-1.059731	1.285741
891193	-1.243244	1.209329	-1.329319
891194	0.122179	0.452976	-0.675554
891195	-1.243244	1.209329	-1.329319
891196	1.487601	-1.059731	1.285741
891197	-0.560532	0.452976	-1.329319
891198	1.487601	-0.303378	0.631976
891199	1.487601	-1.059731	-0.021789
891200	1.487601	-0.303378	1.285741
891201	-0.560532	-0.303378	-0.675554
891202	0.804890	-1.816084	-0.675554
891203	-1.243244	1.209329	-1.329319
891204	0.122179	0.452976	0.631976
891205	-0.560532	0.452976	-0.021789
891206	0.804890	-1.059731	0.631976
891207	-1.243244	1.209329	-1.329319
891208	-0.560532	0.452976	-0.675554
891209	0.804890	-1.059731	-0.021789
891210	0.122179	0.452976	-0.675554
891211	-0.560532	0.452976	-0.021789

891212	-1.243244	1.209329	-1.329319
891213	0.122179	-0.303378	1.285741
891214	1.487601	-1.059731	-0.021789
891215	1.487601	-1.059731	0.631976
891216	0.804890	-1.059731	1.285741
891217	0.122179	-0.303378	-0.675554
891218	0.804890	-1.059731	1.285741
891219	1.487601	-0.303378	1.285741
891220	-0.560532	1.209329	-0.675554

	FINANZ_UNAUFFAELLIGER	FINANZ_HAUSBAUER	FINANZTYP	\
0	1.429871	-0.055511	0.105346	
1	0.757254	1.422415	-1.403804	
2	0.084637	1.422415	-1.403804	
3	-1.260597	-0.794475	1.111445	
4	0.084637	-0.794475	0.608395	
5	-0.587980	1.422415	-0.900754	
6	0.757254	-0.055511	0.105346	
7	0.084637	-0.794475	0.608395	
8	-0.587980	-0.794475	1.111445	
9	1.429871	0.683452	-1.403804	
10	-1.260597	1.422415	1.111445	
11	1.429871	-0.055511	0.105346	
12	0.757254	-1.533438	-0.397704	
13	1.429871	-0.794475	-1.403804	
14	1.429871	-0.055511	0.105346	
15	-1.260597	0.683452	-0.900754	
16	1.429871	-1.533438	-0.397704	
17	1.429871	-0.055511	0.105346	
18	0.757254	-1.533438	-1.403804	
19	0.084637	-1.533438	0.608395	
20	1.429871	-1.533438	-0.397704	
21	1.429871	-1.533438	-0.397704	
22	1.429871	1.422415	-1.403804	
23	-0.587980	-1.533438	1.111445	
24	1.429871	-0.055511	0.105346	
25	0.757254	-0.055511	0.105346	
26	0.084637	-1.533438	-0.397704	
27	-0.587980	-0.794475	0.608395	
28	0.084637	-0.794475	-0.397704	
29	-1.260597	-0.055511	1.111445	
...	...	...	...	
891191	-1.260597	0.683452	-0.900754	
891192	0.084637	1.422415	-1.403804	
891193	-0.587980	1.422415	0.608395	
891194	0.084637	-1.533438	-0.397704	
891195	-1.260597	1.422415	-0.900754	
891196	0.757254	-0.055511	-1.403804	

```
891197           0.084637        -0.055511   0.608395
891198           1.429871        -0.055511  -1.403804
891199           0.084637         0.683452  -1.403804
891200           1.429871         1.422415  -1.403804
891201          -0.587980        -0.055511   1.111445
891202           1.429871         0.683452  -1.403804
891203           0.084637         0.683452   0.608395
891204          -0.587980        -1.533438   1.111445
891205           0.084637         0.683452  -0.900754
891206           0.757254        -0.794475   0.105346
891207          -0.587980         0.683452  -0.900754
891208           0.084637        -1.533438   1.111445
891209           0.757254         0.683452  -1.403804
891210          -0.587980        -0.794475   1.111445
891211           0.084637        -0.794475  -0.900754
891212          -1.260597         1.422415   0.608395
891213           0.084637        -0.794475   1.111445
891214           0.084637         0.683452  -1.403804
891215           1.429871         0.683452  -1.403804
891216           0.757254         0.683452  -1.403804
891217          -0.587980        -0.055511   1.111445
891218           0.757254        -0.055511  -1.403804
891219           1.429871         1.422415  -1.403804
891220          -1.260597         1.422415   1.111445

                   ...           PLZ8_ANTG2    PLZ8_ANTG3    PLZ8_ANTG4  \
0                  ...        -1.035121e-15 -7.240775e-16  1.637641e-16
1                  ...         2.309234e-01  4.397660e-01  4.437471e-01
2                  ...         2.309234e-01 -6.472190e-01 -1.031309e+00
3                  ...        -9.345192e-01  4.397660e-01 -1.031309e+00
4                  ...         1.396366e+00  4.397660e-01  4.437471e-01
5                  ...         2.309234e-01 -6.472190e-01  4.437471e-01
6                  ...         2.309234e-01 -6.472190e-01 -1.031309e+00
7                  ...         2.309234e-01 -6.472190e-01 -1.031309e+00
8                  ...         2.309234e-01  4.397660e-01  4.437471e-01
9                  ...         2.309234e-01  4.397660e-01  4.437471e-01
10                 ...         1.396366e+00  4.397660e-01 -1.031309e+00
11                 ...        -1.035121e-15 -7.240775e-16  1.637641e-16
12                 ...         2.309234e-01 -6.472190e-01 -1.031309e+00
13                 ...        -2.099962e+00 -6.472190e-01  4.437471e-01
14                 ...        -1.035121e-15 -7.240775e-16  1.637641e-16
15                 ...        -1.035121e-15 -7.240775e-16  1.637641e-16
16                 ...         2.309234e-01 -6.472190e-01 -1.031309e+00
17                 ...        -1.035121e-15 -7.240775e-16  1.637641e-16
18                 ...         2.309234e-01  4.397660e-01  4.437471e-01
19                 ...         1.396366e+00  4.397660e-01  4.437471e-01
20                 ...        -1.035121e-15 -7.240775e-16  1.637641e-16
21                 ...         1.396366e+00  1.526751e+00  1.918803e+00
```

22	...	1.396366e+00	4.397660e-01	4.437471e-01
23	...	-1.035121e-15	-7.240775e-16	1.637641e-16
24	...	-1.035121e-15	-7.240775e-16	1.637641e-16
25	...	-9.345192e-01	-1.734204e+00	-1.031309e+00
26	...	-1.035121e-15	-7.240775e-16	1.637641e-16
27	...	2.309234e-01	1.526751e+00	1.918803e+00
28	...	2.309234e-01	-6.472190e-01	-1.031309e+00
29	...	-9.345192e-01	4.397660e-01	-1.031309e+00
...	...	...	...	...
891191	...	2.309234e-01	-6.472190e-01	-1.031309e+00
891192	...	-9.345192e-01	-6.472190e-01	-1.031309e+00
891193	...	1.396366e+00	-6.472190e-01	-1.031309e+00
891194	...	-9.345192e-01	4.397660e-01	-1.031309e+00
891195	...	2.309234e-01	4.397660e-01	4.437471e-01
891196	...	2.309234e-01	4.397660e-01	4.437471e-01
891197	...	2.309234e-01	4.397660e-01	4.437471e-01
891198	...	-9.345192e-01	-6.472190e-01	-1.031309e+00
891199	...	1.396366e+00	1.526751e+00	1.918803e+00
891200	...	1.396366e+00	4.397660e-01	4.437471e-01
891201	...	1.396366e+00	1.526751e+00	4.437471e-01
891202	...	1.396366e+00	1.526751e+00	1.918803e+00
891203	...	2.309234e-01	4.397660e-01	4.437471e-01
891204	...	2.309234e-01	-6.472190e-01	-1.031309e+00
891205	...	1.396366e+00	1.526751e+00	4.437471e-01
891206	...	-9.345192e-01	-6.472190e-01	-1.031309e+00
891207	...	2.309234e-01	4.397660e-01	1.918803e+00
891208	...	-9.345192e-01	-1.734204e+00	-1.031309e+00
891209	...	1.396366e+00	4.397660e-01	4.437471e-01
891210	...	2.309234e-01	4.397660e-01	4.437471e-01
891211	...	1.396366e+00	4.397660e-01	-1.031309e+00
891212	...	2.309234e-01	1.526751e+00	1.918803e+00
891213	...	1.396366e+00	4.397660e-01	4.437471e-01
891214	...	1.396366e+00	1.526751e+00	1.918803e+00
891215	...	1.396366e+00	4.397660e-01	4.437471e-01
891216	...	-9.345192e-01	-1.734204e+00	-1.031309e+00
891217	...	2.309234e-01	1.526751e+00	1.918803e+00
891218	...	-9.345192e-01	-1.734204e+00	-1.031309e+00
891219	...	1.396366e+00	1.526751e+00	4.437471e-01
891220	...	2.309234e-01	-6.472190e-01	-1.031309e+00

	PLZ8_HHZ	PLZ8_GBZ	ARBEIT	ORTSGR_KLS9	RELAT_AB	\
0	-4.890471e-16	-4.284963e-16	-4.709754e-16	2.042688e-15	-3.458487e-16	
1	1.527612e+00	5.971822e-01	-1.767775e-01	-1.349507e-01	7.234631e-01	
2	4.263757e-01	5.971822e-01	-1.767775e-01	-1.349507e-01	-8.341011e-01	
3	-6.748605e-01	5.971822e-01	-1.237320e+00	-1.054896e+00	-5.531897e-02	
4	-6.748605e-01	-3.677058e-01	8.837647e-01	3.250218e-01	1.502245e+00	
5	1.527612e+00	1.562070e+00	-1.237320e+00	-1.054896e+00	-5.531897e-02	
6	1.527612e+00	1.562070e+00	8.837647e-01	3.250218e-01	-5.531897e-02	

7	4.263757e-01	5.971822e-01	-1.237320e+00	-1.349507e-01	-8.341011e-01
8	-6.748605e-01	-3.677058e-01	-1.237320e+00	-5.949232e-01	-5.531897e-02
9	-6.748605e-01	-3.677058e-01	-1.237320e+00	-1.054896e+00	-1.612883e+00
10	-6.748605e-01	-3.677058e-01	8.837647e-01	3.250218e-01	1.502245e+00
11	-4.890471e-16	-4.284963e-16	-4.709754e-16	2.042688e-15	-3.458487e-16
12	1.527612e+00	1.562070e+00	-1.767775e-01	3.250218e-01	7.234631e-01
13	-6.748605e-01	-3.677058e-01	-1.767775e-01	3.250218e-01	7.234631e-01
14	-4.890471e-16	-4.284963e-16	-4.709754e-16	2.042688e-15	-3.458487e-16
15	-4.890471e-16	-4.284963e-16	8.837647e-01	1.244967e+00	1.502245e+00
16	-6.748605e-01	5.971822e-01	-2.297862e+00	-1.514868e+00	-1.612883e+00
17	-4.890471e-16	-4.284963e-16	-4.709754e-16	2.042688e-15	-3.458487e-16
18	-6.748605e-01	-3.677058e-01	-1.767775e-01	-5.949232e-01	-5.531897e-02
19	1.527612e+00	5.971822e-01	8.837647e-01	3.250218e-01	-5.531897e-02
20	-4.890471e-16	-4.284963e-16	-1.767775e-01	-5.949232e-01	-1.612883e+00
21	4.263757e-01	-3.677058e-01	1.944307e+00	7.849944e-01	1.502245e+00
22	4.263757e-01	-3.677058e-01	8.837647e-01	-1.349507e-01	1.502245e+00
23	-4.890471e-16	-4.284963e-16	-1.767775e-01	3.250218e-01	-8.341011e-01
24	-4.890471e-16	-4.284963e-16	-4.709754e-16	2.042688e-15	-3.458487e-16
25	4.263757e-01	1.562070e+00	-1.767775e-01	-1.514868e+00	1.502245e+00
26	-4.890471e-16	-4.284963e-16	8.837647e-01	-1.054896e+00	1.502245e+00
27	1.527612e+00	-3.677058e-01	8.837647e-01	7.849944e-01	1.502245e+00
28	1.527612e+00	1.562070e+00	-1.767775e-01	-5.949232e-01	-1.612883e+00
29	-6.748605e-01	5.971822e-01	-1.237320e+00	-5.949232e-01	-5.531897e-02
...	...	...	...	...	...
891191	4.263757e-01	1.562070e+00	-2.297862e+00	-1.054896e+00	-1.612883e+00
891192	-6.748605e-01	-3.677058e-01	-1.767775e-01	-1.349507e-01	-1.612883e+00
891193	-1.776097e+00	-1.332594e+00	8.837647e-01	3.250218e-01	7.234631e-01
891194	4.263757e-01	5.971822e-01	8.837647e-01	1.244967e+00	1.502245e+00
891195	4.263757e-01	-3.677058e-01	8.837647e-01	7.849944e-01	1.502245e+00
891196	-6.748605e-01	-3.677058e-01	-1.767775e-01	-5.949232e-01	1.502245e+00
891197	4.263757e-01	5.971822e-01	-1.767775e-01	1.704939e+00	1.502245e+00
891198	4.263757e-01	1.562070e+00	-1.767775e-01	1.704939e+00	1.502245e+00
891199	1.527612e+00	-3.677058e-01	-1.767775e-01	7.849944e-01	-5.531897e-02
891200	-6.748605e-01	-3.677058e-01	8.837647e-01	3.250218e-01	1.502245e+00
891201	4.263757e-01	-3.677058e-01	-1.767775e-01	7.849944e-01	1.502245e+00
891202	4.263757e-01	-1.332594e+00	8.837647e-01	1.244967e+00	1.502245e+00
891203	-6.748605e-01	-3.677058e-01	8.837647e-01	1.244967e+00	1.502245e+00
891204	-6.748605e-01	5.971822e-01	8.837647e-01	7.849944e-01	-5.531897e-02
891205	-6.748605e-01	-1.332594e+00	-1.767775e-01	7.849944e-01	1.502245e+00
891206	4.263757e-01	1.562070e+00	-1.767775e-01	-5.949232e-01	-5.531897e-02
891207	4.263757e-01	-1.332594e+00	-2.297862e+00	-1.349507e-01	-1.612883e+00
891208	-6.748605e-01	-3.677058e-01	8.837647e-01	-1.514868e+00	1.502245e+00
891209	1.527612e+00	5.971822e-01	8.837647e-01	-1.349507e-01	1.502245e+00
891210	-6.748605e-01	-3.677058e-01	-1.767775e-01	1.704939e+00	1.502245e+00
891211	-6.748605e-01	-1.332594e+00	-1.767775e-01	-1.349507e-01	1.502245e+00
891212	4.263757e-01	-2.297482e+00	-1.767775e-01	1.704939e+00	1.502245e+00
891213	1.527612e+00	1.562070e+00	-1.767775e-01	-5.949232e-01	7.234631e-01
891214	1.527612e+00	-1.332594e+00	-1.767775e-01	7.849944e-01	-5.531897e-02

891215	4.263757e-01	5.971822e-01	-1.237320e+00	-1.349507e-01	-8.341011e-01
891216	-1.776097e+00	-3.677058e-01	-4.709754e-16	2.042688e-15	-3.458487e-16
891217	1.527612e+00	-3.677058e-01	8.837647e-01	3.250218e-01	1.502245e+00
891218	-6.748605e-01	5.971822e-01	-1.237320e+00	-1.514868e+00	-5.531897e-02
891219	-2.877333e+00	-2.297482e+00	8.837647e-01	7.849944e-01	1.502245e+00
891220	4.263757e-01	5.971822e-01	-1.767775e-01	-5.949232e-01	1.502245e+00

	CAMEO_INTL_2015_wealth	CAMEO_INTL_2015_life_stage
0	3.216984e-16	6.346104e-16
1	1.258937e+00	-1.338297e+00
2	-9.142640e-01	8.052265e-01
3	-1.638664e+00	-6.237891e-01
4	5.345367e-01	9.071870e-02
5	1.258937e+00	8.052265e-01
6	-9.142640e-01	-6.237891e-01
7	-1.638664e+00	8.052265e-01
8	-1.638664e+00	9.071870e-02
9	-1.638664e+00	1.519734e+00
10	1.258937e+00	-1.338297e+00
11	3.216984e-16	6.346104e-16
12	5.345367e-01	9.071870e-02
13	-1.898636e-01	9.071870e-02
14	3.216984e-16	6.346104e-16
15	5.345367e-01	-1.338297e+00
16	5.345367e-01	-1.338297e+00
17	3.216984e-16	6.346104e-16
18	-9.142640e-01	8.052265e-01
19	-1.898636e-01	8.052265e-01
20	-9.142640e-01	8.052265e-01
21	1.258937e+00	1.519734e+00
22	1.258937e+00	-1.338297e+00
23	5.345367e-01	9.071870e-02
24	3.216984e-16	6.346104e-16
25	-1.898636e-01	9.071870e-02
26	3.216984e-16	6.346104e-16
27	1.258937e+00	-1.338297e+00
28	-1.638664e+00	9.071870e-02
29	-1.638664e+00	-6.237891e-01
...	...	...
891191	5.345367e-01	8.052265e-01
891192	-1.898636e-01	-6.237891e-01
891193	5.345367e-01	9.071870e-02
891194	-9.142640e-01	8.052265e-01
891195	1.258937e+00	1.519734e+00
891196	-9.142640e-01	9.071870e-02
891197	-9.142640e-01	8.052265e-01
891198	-1.638664e+00	8.052265e-01
891199	1.258937e+00	-1.338297e+00

```
891200          1.258937e+00              -6.237891e-01
891201          5.345367e-01              -1.338297e+00
891202          5.345367e-01              -1.338297e+00
891203         -9.142640e-01               1.519734e+00
891204         -9.142640e-01               8.052265e-01
891205         -1.898636e-01               8.052265e-01
891206         -9.142640e-01               1.519734e+00
891207          5.345367e-01              -1.338297e+00
891208         -1.638664e+00               8.052265e-01
891209          1.258937e+00              -1.338297e+00
891210          5.345367e-01              -1.338297e+00
891211          1.258937e+00              -1.338297e+00
891212          1.258937e+00              -1.338297e+00
891213         -1.898636e-01               8.052265e-01
891214         -9.142640e-01               9.071870e-02
891215         -1.898636e-01              -1.338297e+00
891216          5.345367e-01              -1.338297e+00
891217          1.258937e+00              -1.338297e+00
891218         -9.142640e-01               8.052265e-01
891219          1.258937e+00              -1.338297e+00
891220          5.345367e-01               9.071870e-02

[891221 rows x 73 columns]
```

1.2.2 Discussion 2.1: Apply Feature Scaling

After cleaning and transforming the dataset, I used the imputer function to remove any remaining NaNs and then the StandardScaler to perform feature scaling.

1.2.3 Step 2.2: Perform Dimensionality Reduction

On your scaled data, you are now ready to apply dimensionality reduction techniques.

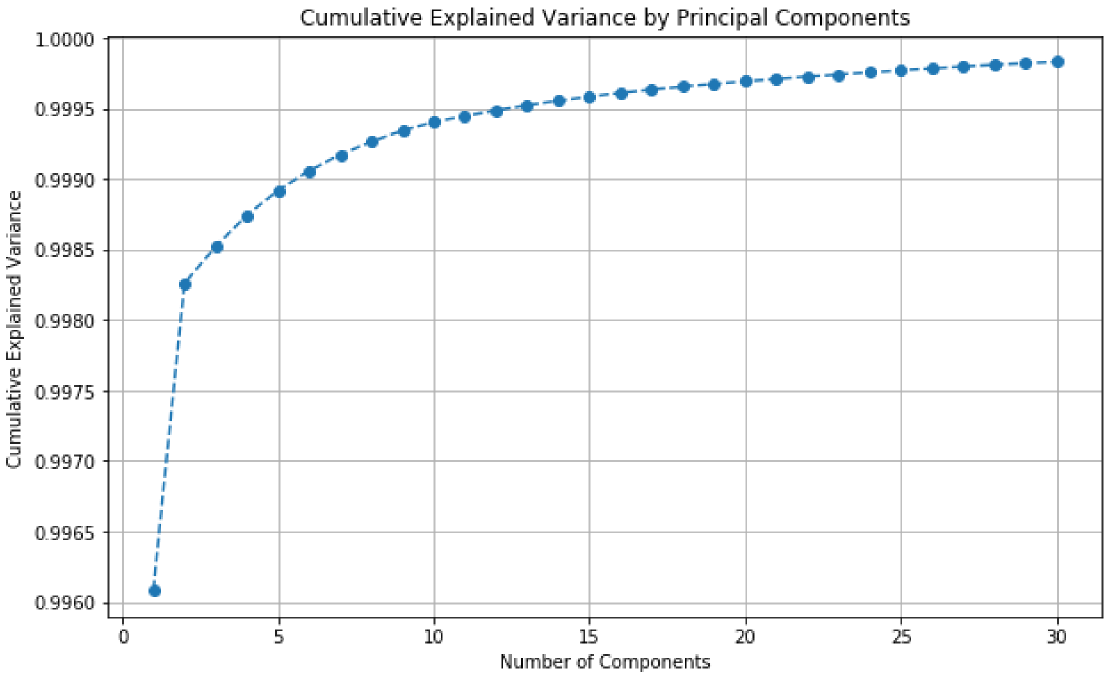
- Use sklearn's PCA class to apply principal component analysis on the data, thus finding the vectors of maximal variance in the data. To start, you should not set any parameters (so all components are computed) or set a number of components that is at least half the number of features (so there's enough features to see the general trend in variability).
- Check out the ratio of variance explained by each principal component as well as the cumulative variance explained. Try plotting the cumulative or sequential values using matplotlib's `plot()` function. Based on what you find, select a value for the number of transformed features you'll retain for the clustering part of the project.
- Once you've made a choice for the number of components to keep, make sure you re-fit a PCA instance to perform the decided-on transformation.

In [41]:
```python
# Apply PCA to the data.
# Referenced: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.P

from sklearn.decomposition import PCA
```

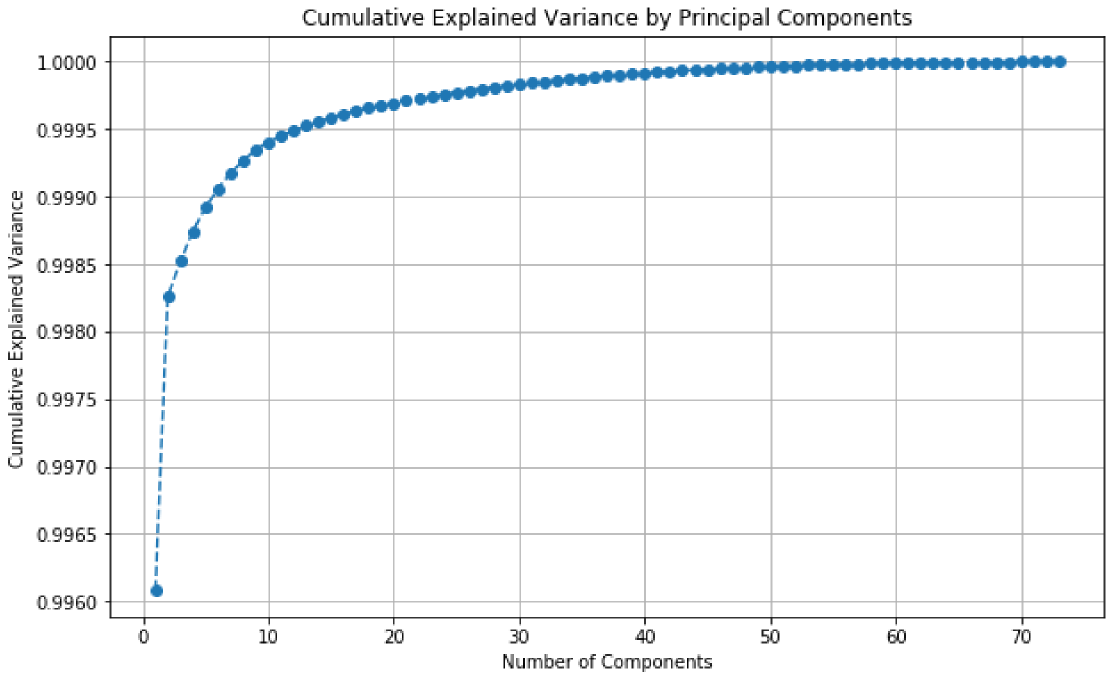
```
# Initialize PCA instance
pca = PCA()

# Fit PCA to the scaled data
transformed_pca = pca.fit(azdias_cleaned_copy)
```

In [90]:
```
# Investigate the variance accounted for by each principal component.
# Referenced: https://vitalflux.com/pca-explained-variance-concept-python-example/
# Get eplained variance ratio
explained_variance = pca.explained_variance_ratio_

print(explained_variance)

cumulative_var = np.cumsum(explained_variance)

# Plot the cumulative explained variance
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_var) + 1), cumulative_var, marker='o', linestyle='--')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Cumulative Explained Variance by Principal Components')
plt.grid(True)
plt.show()
```

```
[ 9.96082891e-01   2.17774224e-03   2.60370941e-04   2.23467237e-04
  1.74662543e-04   1.40043983e-04   1.15223549e-04   9.17216625e-05
  8.08838220e-05   5.79596604e-05   4.33842728e-05   3.98594760e-05
  3.63269757e-05   3.20105142e-05   2.92201331e-05   2.68347793e-05
  2.39965661e-05   2.06084149e-05   1.89065543e-05   1.83706783e-05
  1.70697061e-05   1.61749242e-05   1.56931881e-05   1.49891938e-05
  1.46475182e-05   1.40835114e-05   1.32556312e-05   1.22324548e-05
  1.08888559e-05   1.01437847e-05]
```



In [43]:
```
# Re-apply PCA to the data while selecting for number of components to retain.
# Number of components to retain
num_components_to_retain = 30  # line appears to significantly slow in rise at this poi
pca = PCA(n_components=num_components_to_retain)

# Fit PCA to scaled data and transform
transformed_pca = pca.fit_transform(azdias_cleaned_copy)

# Calculate cumulative explained variance
cumulative_var = np.cumsum(explained_variance)

# Plot the cumulative explained variance
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_var) + 1), cumulative_var, marker='o', linestyle='--')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Cumulative Explained Variance by Principal Components')
plt.grid(True)
plt.show()
```



1.2.4 Discussion 2.2: Perform Dimensionality Reduction

It appears that at around 50, the variance is almost nonexistent, there was no change when selecting 50. I tried again with 30 and there is little to no change.

1.2.5 Step 2.3: Interpret Principal Components

Now that we have our transformed principal components, it's a nice idea to check out the weight of each variable on the first few components to see if they can be interpreted in some fashion.

As a reminder, each principal component is a unit vector that points in the direction of highest variance (after accounting for the variance captured by earlier principal components). The further a weight is from zero, the more the principal component is in the direction of the corresponding feature. If two features have large weights of the same sign (both positive or both negative), then increases in one tend expect to be associated with increases in the other. To contrast, features with different signs can be expected to show a negative correlation: increases in one variable should result in a decrease in the other.

- To investigate the features, you should map each weight to their corresponding feature name, then sort the features according to weight. The most interesting features for each principal component, then, will be those at the beginning and end of the sorted list. Use the data dictionary document to help you understand these most prominent features, their relationships, and what a positive or negative value on the principal component might indicate.
- You should investigate and interpret feature associations from the first three principal components in this substep. To help facilitate this, you should write a function that you can call at any time to print the sorted list of feature weights, for the *i*-th principal component. This

might come in handy in the next step of the project, when you interpret the tendencies of the discovered clusters.

In [44]:
```
# Map weights for the first principal component to corresponding feature names
# and then print the linked values, sorted by weight.
# HINT: Try defining a function here or in a new cell that you can reuse in the
# other cells.

def map_weights(pca, component_number, df):
    """
    Print the sorted list of feature weights for a specific principal component sorted
    """
    # Get the weights of first principal component.
    weights = pca.components_[component_number]

    # Get the names of features.
    features = df.columns

    # Create a dataframe with the weights and features.
    weights = pd.DataFrame({'weights': weights, 'features': features})

    # Sort the dataframe by the weights.
    weights.sort_values(by='weights', inplace=True)

    # Print the dataframe.
    print(weights)

    return weights


# first principal component
first_component_mapped = map_weights(pca, 0, azdias_cleaned_copy)
```

```
     weights                features
42 -0.003928       ANZ_HAUSHALTE_AKTIV
69 -0.001534              ORTSGR_KLS9
54 -0.001069                 EWDICHTE
70 -0.001028                 RELAT_AB
71 -0.000842    CAMEO_INTL_2015_wealth
68 -0.000836                   ARBEIT
39 -0.000791         HH_EINKOMMEN_SCORE
8  -0.000505           FINANZ_HAUSBAUER
60 -0.000468                 REGIOTYP
64 -0.000384               PLZ8_ANTG3
36 -0.000363                 ZABEOTYP
44 -0.000361              GEBAEUDETYP
65 -0.000351               PLZ8_ANTG4
9  -0.000300                FINANZTYP
```

40	-0.000255	W_KEIT_KIND_HH
10	-0.000252	GFK_URLAUBERTYP
57	-0.000242	KKK
50	-0.000218	KBA05_ANTG3
19	-0.000193	SEMIO_SOZ
51	-0.000190	KBA05_ANTG4
26	-0.000182	SEMIO_KULT
30	-0.000171	SEMIO_KAEM
63	-0.000159	PLZ8_ANTG2
20	-0.000140	SEMIO_FAM
4	-0.000137	FINANZ_SPARER
21	-0.000130	SEMIO_REL
31	-0.000102	SEMIO_PFLICHT
18	-0.000098	RETOURTYP_BK_S
22	-0.000040	SEMIO_MAT
35	-0.000039	VERS_TYP
..	...	...
6	0.000050	FINANZ_ANLEGER
25	0.000052	SEMIO_ERL
24	0.000054	SEMIO_LUST
23	0.000080	SEMIO_VERT
11	0.000098	GREEN_AVANTGARDE
41	0.000100	WOHNDAUER_2008
49	0.000118	KBA05_ANTG2
37	0.000181	ANZ_PERSONEN
7	0.000200	FINANZ_UNAUFFAELLIGER
5	0.000214	FINANZ_VORSORGER
47	0.000230	OST_WEST_KZ
45	0.000236	KONSUMNAEHE
56	0.000247	GEBAEUDETYP_RASTER
14	0.000270	LP_FAMILIE_GROB
59	0.000278	ONLINE_AFFINITAET
3	0.000344	FINANZ_MINIMALIST
46	0.000407	MIN_GEBAEUDEJAHR
52	0.000423	KBA05_GBZ
72	0.000434	CAMEO_INTL_2015_life_stage
48	0.000585	KBA05_ANTG1
16	0.000592	LP_STATUS_GROB
53	0.000598	BALLRAUM
13	0.000599	LP_FAMILIE_FEIN
62	0.000601	PLZ8_ANTG1
58	0.000603	MOBI_REGIO
55	0.001124	INNENSTADT
15	0.001420	LP_STATUS_FEIN
66	0.002018	PLZ8_HHZ
67	0.002328	PLZ8_GBZ
61	0.999980	KBA13_ANZAHL_PKW

```
[73 rows x 2 columns]
```

```
In [45]: print("Most Influential Features for First Principal Component:")
         print(first_component_mapped.head(5))  # Top 5 positive weights
         print(first_component_mapped.tail(5))  # Top 5 negative weights
```

```
Most Influential Features for First Principal Component:
     weights               features
42 -0.003928     ANZ_HAUSHALTE_AKTIV
69 -0.001534            ORTSGR_KLS9
54 -0.001069               EWDICHTE
70 -0.001028               RELAT_AB
71 -0.000842  CAMEO_INTL_2015_wealth
     weights         features
55  0.001124       INNENSTADT
15  0.001420    LP_STATUS_FEIN
66  0.002018         PLZ8_HHZ
67  0.002328         PLZ8_GBZ
61  0.999980  KBA13_ANZAHL_PKW
```

```
In [46]: # Map weights for the second principal component to corresponding feature names
         # and then print the linked values, sorted by weight.

         second_component_mapped = map_weights(pca, 0, azdias_cleaned_copy)

         second_component_mapped
```

```
     weights                features
42 -0.003928       ANZ_HAUSHALTE_AKTIV
69 -0.001534              ORTSGR_KLS9
54 -0.001069                 EWDICHTE
70 -0.001028                 RELAT_AB
71 -0.000842    CAMEO_INTL_2015_wealth
68 -0.000836                   ARBEIT
39 -0.000791        HH_EINKOMMEN_SCORE
8  -0.000505          FINANZ_HAUSBAUER
60 -0.000468                 REGIOTYP
64 -0.000384               PLZ8_ANTG3
36 -0.000363                 ZABEOTYP
44 -0.000361               GEBAEUDETYP
65 -0.000351               PLZ8_ANTG4
9  -0.000300                FINANZTYP
40 -0.000255            W_KEIT_KIND_HH
10 -0.000252           GFK_URLAUBERTYP
57 -0.000242                      KKK
50 -0.000218               KBA05_ANTG3
19 -0.000193                SEMIO_SOZ
```

```
51 -0.000190                 KBA05_ANTG4
26 -0.000182                  SEMIO_KULT
30 -0.000171                  SEMIO_KAEM
63 -0.000159                  PLZ8_ANTG2
20 -0.000140                   SEMIO_FAM
4  -0.000137               FINANZ_SPARER
21 -0.000130                   SEMIO_REL
31 -0.000102               SEMIO_PFLICHT
18 -0.000098              RETOURTYP_BK_S
22 -0.000040                   SEMIO_MAT
35 -0.000039                    VERS_TYP
..       ...                         ...
6   0.000050              FINANZ_ANLEGER
25  0.000052                   SEMIO_ERL
24  0.000054                  SEMIO_LUST
23  0.000080                  SEMIO_VERT
11  0.000098            GREEN_AVANTGARDE
41  0.000100              WOHNDAUER_2008
49  0.000118                 KBA05_ANTG2
37  0.000181                ANZ_PERSONEN
7   0.000200        FINANZ_UNAUFFAELLIGER
5   0.000214            FINANZ_VORSORGER
47  0.000230                 OST_WEST_KZ
45  0.000236                 KONSUMNAEHE
56  0.000247           GEBAEUDETYP_RASTER
14  0.000270             LP_FAMILIE_GROB
59  0.000278            ONLINE_AFFINITAET
3   0.000344            FINANZ_MINIMALIST
46  0.000407            MIN_GEBAEUDEJAHR
52  0.000423                   KBA05_GBZ
72  0.000434  CAMEO_INTL_2015_life_stage
48  0.000585                 KBA05_ANTG1
16  0.000592              LP_STATUS_GROB
53  0.000598                    BALLRAUM
13  0.000599             LP_FAMILIE_FEIN
62  0.000601                  PLZ8_ANTG1
58  0.000603                  MOBI_REGIO
55  0.001124                  INNENSTADT
15  0.001420              LP_STATUS_FEIN
66  0.002018                    PLZ8_HHZ
67  0.002328                    PLZ8_GBZ
61  0.999980              KBA13_ANZAHL_PKW

[73 rows x 2 columns]
```

```
Out[46]:        weights                     features
         42 -0.003928          ANZ_HAUSHALTE_AKTIV
```

69	-0.001534	ORTSGR_KLS9
54	-0.001069	EWDICHTE
70	-0.001028	RELAT_AB
71	-0.000842	CAMEO_INTL_2015_wealth
68	-0.000836	ARBEIT
39	-0.000791	HH_EINKOMMEN_SCORE
8	-0.000505	FINANZ_HAUSBAUER
60	-0.000468	REGIOTYP
64	-0.000384	PLZ8_ANTG3
36	-0.000363	ZABEOTYP
44	-0.000361	GEBAEUDETYP
65	-0.000351	PLZ8_ANTG4
9	-0.000300	FINANZTYP
40	-0.000255	W_KEIT_KIND_HH
10	-0.000252	GFK_URLAUBERTYP
57	-0.000242	KKK
50	-0.000218	KBA05_ANTG3
19	-0.000193	SEMIO_SOZ
51	-0.000190	KBA05_ANTG4
26	-0.000182	SEMIO_KULT
30	-0.000171	SEMIO_KAEM
63	-0.000159	PLZ8_ANTG2
20	-0.000140	SEMIO_FAM
4	-0.000137	FINANZ_SPARER
21	-0.000130	SEMIO_REL
31	-0.000102	SEMIO_PFLICHT
18	-0.000098	RETOURTYP_BK_S
22	-0.000040	SEMIO_MAT
35	-0.000039	VERS_TYP
..	...	...
6	0.000050	FINANZ_ANLEGER
25	0.000052	SEMIO_ERL
24	0.000054	SEMIO_LUST
23	0.000080	SEMIO_VERT
11	0.000098	GREEN_AVANTGARDE
41	0.000100	WOHNDAUER_2008
49	0.000118	KBA05_ANTG2
37	0.000181	ANZ_PERSONEN
7	0.000200	FINANZ_UNAUFFAELLIGER
5	0.000214	FINANZ_VORSORGER
47	0.000230	OST_WEST_KZ
45	0.000236	KONSUMNAEHE
56	0.000247	GEBAEUDETYP_RASTER
14	0.000270	LP_FAMILIE_GROB
59	0.000278	ONLINE_AFFINITAET
3	0.000344	FINANZ_MINIMALIST
46	0.000407	MIN_GEBAEUDEJAHR
52	0.000423	KBA05_GBZ

```
72  0.000434  CAMEO_INTL_2015_life_stage
48  0.000585                 KBA05_ANTG1
16  0.000592              LP_STATUS_GROB
53  0.000598                    BALLRAUM
13  0.000599             LP_FAMILIE_FEIN
62  0.000601                 PLZ8_ANTG1
58  0.000603                 MOBI_REGIO
55  0.001124                 INNENSTADT
15  0.001420              LP_STATUS_FEIN
66  0.002018                    PLZ8_HHZ
67  0.002328                    PLZ8_GBZ
61  0.999980             KBA13_ANZAHL_PKW

[73 rows x 2 columns]
```

In [47]:
```python
print("Most Influential Features for Second Principal Component:")
print(second_component_mapped.head(5))  # Top 5 positive weights
print(second_component_mapped.tail(5))  # Top 5 negative weights
```

```
Most Influential Features for Second Principal Component:
     weights                features
42 -0.003928     ANZ_HAUSHALTE_AKTIV
69 -0.001534             ORTSGR_KLS9
54 -0.001069                EWDICHTE
70 -0.001028                RELAT_AB
71 -0.000842  CAMEO_INTL_2015_wealth
     weights          features
55  0.001124        INNENSTADT
15  0.001420    LP_STATUS_FEIN
66  0.002018          PLZ8_HHZ
67  0.002328          PLZ8_GBZ
61  0.999980  KBA13_ANZAHL_PKW
```

In [48]:
```python
# Map weights for the third principal component to corresponding feature names
# and then print the linked values, sorted by weight.

third_component_mapped = map_weights(pca, 2, azdias_cleaned_copy)

third_component_mapped
```

```
     weights                  features
10 -0.230277           GFK_URLAUBERTYP
21 -0.176024                 SEMIO_REL
69 -0.170431               ORTSGR_KLS9
31 -0.163639              SEMIO_PFLICHT
4  -0.154389              FINANZ_SPARER
39 -0.146118          HH_EINKOMMEN_SCORE
32 -0.130798                SEMIO_TRADV
```

```
54 -0.128486                   EWDICHTE
27 -0.119277                  SEMIO_RAT
71 -0.117623     CAMEO_INTL_2015_wealth
22 -0.117556                  SEMIO_MAT
20 -0.113562                  SEMIO_FAM
36 -0.110987                   ZABEOTYP
8  -0.108926            FINANZ_HAUSBAUER
26 -0.108357                 SEMIO_KULT
7  -0.106773      FINANZ_UNAUFFAELLIGER
44 -0.102133                 GEBAEUDETYP
6  -0.094772              FINANZ_ANLEGER
64 -0.084656                  PLZ8_ANTG3
50 -0.073734                 KBA05_ANTG3
70 -0.064950                    RELAT_AB
63 -0.063736                  PLZ8_ANTG2
2  -0.059750               CJT_GESAMTTYP
40 -0.053513              W_KEIT_KIND_HH
65 -0.051459                  PLZ8_ANTG4
60 -0.049909                    REGIOTYP
68 -0.049319                      ARBEIT
19 -0.041560                   SEMIO_SOZ
49 -0.040493                 KBA05_ANTG2
66 -0.037365                    PLZ8_HHZ
..       ...                         ...
43  0.005486                ANZ_HH_TITEL
33  0.019884                 SHOPPER_TYP
11  0.023990             GREEN_AVANTGARDE
59  0.032958            ONLINE_AFFINITAET
56  0.033573          GEBAEUDETYP_RASTER
67  0.037614                    PLZ8_GBZ
18  0.041042              RETOURTYP_BK_S
23  0.052128                  SEMIO_VERT
62  0.076060                  PLZ8_ANTG1
37  0.080996                 ANZ_PERSONEN
72  0.083452  CAMEO_INTL_2015_life_stage
41  0.093742               WOHNDAUER_2008
0   0.094290          ALTERSKATEGORIE_GROB
46  0.095786             MIN_GEBAEUDEJAHR
52  0.101258                   KBA05_GBZ
45  0.107300                 KONSUMNAEHE
53  0.108315                    BALLRAUM
5   0.110855             FINANZ_VORSORGER
9   0.118068                  FINANZTYP
25  0.119526                   SEMIO_ERL
24  0.124528                  SEMIO_LUST
55  0.125263                  INNENSTADT
48  0.127021                 KBA05_ANTG1
58  0.129415                  MOBI_REGIO
```

```
14  0.137737             LP_FAMILIE_GROB
42  0.145813         ANZ_HAUSHALTE_AKTIV
3   0.176195           FINANZ_MINIMALIST
16  0.200974              LP_STATUS_GROB
13  0.317554             LP_FAMILIE_FEIN
15  0.460532              LP_STATUS_FEIN

[73 rows x 2 columns]
```

```
Out[48]:      weights                    features
         10 -0.230277             GFK_URLAUBERTYP
         21 -0.176024                   SEMIO_REL
         69 -0.170431                 ORTSGR_KLS9
         31 -0.163639               SEMIO_PFLICHT
         4  -0.154389               FINANZ_SPARER
         39 -0.146118          HH_EINKOMMEN_SCORE
         32 -0.130798                 SEMIO_TRADV
         54 -0.128486                    EWDICHTE
         27 -0.119277                   SEMIO_RAT
         71 -0.117623      CAMEO_INTL_2015_wealth
         22 -0.117556                   SEMIO_MAT
         20 -0.113562                   SEMIO_FAM
         36 -0.110987                    ZABEOTYP
         8  -0.108926            FINANZ_HAUSBAUER
         26 -0.108357                  SEMIO_KULT
         7  -0.106773       FINANZ_UNAUFFAELLIGER
         44 -0.102133                 GEBAEUDETYP
         6  -0.094772               FINANZ_ANLEGER
         64 -0.084656                  PLZ8_ANTG3
         50 -0.073734                 KBA05_ANTG3
         70 -0.064950                    RELAT_AB
         63 -0.063736                  PLZ8_ANTG2
         2  -0.059750               CJT_GESAMTTYP
         40 -0.053513               W_KEIT_KIND_HH
         65 -0.051459                  PLZ8_ANTG4
         60 -0.049909                   REGIOTYP
         68 -0.049319                      ARBEIT
         19 -0.041560                   SEMIO_SOZ
         49 -0.040493                 KBA05_ANTG2
         66 -0.037365                    PLZ8_HHZ
         ..       ...                         ...
         43  0.005486                ANZ_HH_TITEL
         33  0.019884                 SHOPPER_TYP
         11  0.023990             GREEN_AVANTGARDE
         59  0.032958            ONLINE_AFFINITAET
         56  0.033573          GEBAEUDETYP_RASTER
         67  0.037614                    PLZ8_GBZ
```

```
18  0.041042              RETOURTYP_BK_S
23  0.052128                  SEMIO_VERT
62  0.076060                  PLZ8_ANTG1
37  0.080996                ANZ_PERSONEN
72  0.083452  CAMEO_INTL_2015_life_stage
41  0.093742              WOHNDAUER_2008
0   0.094290         ALTERSKATEGORIE_GROB
46  0.095786            MIN_GEBAEUDEJAHR
52  0.101258                   KBA05_GBZ
45  0.107300                 KONSUMNAEHE
53  0.108315                    BALLRAUM
5   0.110855            FINANZ_VORSORGER
9   0.118068                   FINANZTYP
25  0.119526                   SEMIO_ERL
24  0.124528                  SEMIO_LUST
55  0.125263                  INNENSTADT
48  0.127021                 KBA05_ANTG1
58  0.129415                  MOBI_REGIO
14  0.137737             LP_FAMILIE_GROB
42  0.145813          ANZ_HAUSHALTE_AKTIV
3   0.176195            FINANZ_MINIMALIST
16  0.200974              LP_STATUS_GROB
13  0.317554             LP_FAMILIE_FEIN
15  0.460532              LP_STATUS_FEIN

[73 rows x 2 columns]
```

In [49]:
```python
print("Most Influential Features for Third Principal Component:")
print(third_component_mapped.head(5))  # Top 5 positive weights
print(third_component_mapped.tail(5))  # Top 5 negative weights
```

```
Most Influential Features for Third Principal Component:
     weights         features
10 -0.230277  GFK_URLAUBERTYP
21 -0.176024        SEMIO_REL
69 -0.170431      ORTSGR_KLS9
31 -0.163639    SEMIO_PFLICHT
4  -0.154389    FINANZ_SPARER
     weights             features
42  0.145813  ANZ_HAUSHALTE_AKTIV
3   0.176195    FINANZ_MINIMALIST
16  0.200974       LP_STATUS_GROB
13  0.317554      LP_FAMILIE_FEIN
15  0.460532       LP_STATUS_FEIN
```

1.2.6 Discussion 2.3: Interpret Principal Components

First Principal Component: The most influential negative features are associated with characteristics related to the number of households, size of the community, wealth, and socio-economic status. The most influential positive feature is KBA13_ANZAHL_PKW, which represents the number of cars in the microcell. This indicates that the presence of many cars in the community is significant.

Second Principal Component: The most influential negative features are associated with status, mobility, and consumer behavior. The most influential positive features are associated with characteristics related to wealth, density, and the number of households.

Third Principal Component: The most influential negative features are associated with travel and consumer behavior, suggesting a tendency towards less interest in travel and less consumer affinity.The most influential positive features are associated with characteristics related to household activity, minimalism, and status.

1.3 Step 3: Clustering

1.3.1 Step 3.1: Apply Clustering to General Population

You've assessed and cleaned the demographics data, then scaled and transformed them. Now, it's time to see how the data clusters in the principal components space. In this substep, you will apply k-means clustering to the dataset and use the average within-cluster distances from each point to their assigned cluster's centroid to decide on a number of clusters to keep.

- Use sklearn's KMeans class to perform k-means clustering on the PCA-transformed data.
- Then, compute the average difference from each point to its assigned cluster's center. **Hint**: The KMeans object's `.score()` method might be useful here, but note that in sklearn, scores tend to be defined so that larger is better. Try applying it to a small, toy dataset, or use an internet search to help your understanding.
- Perform the above two steps for a number of different cluster counts. You can then see how the average distance decreases with an increasing number of clusters. However, each additional cluster provides a smaller net benefit. Use this fact to select a final number of clusters in which to group the data. **Warning**: because of the large size of the dataset, it can take a long time for the algorithm to resolve. The more clusters to fit, the longer the algorithm will take. You should test for cluster counts through at least 10 clusters to get the full picture, but you shouldn't need to test for a number of clusters above about 30.
- Once you've selected a final number of clusters to use, re-fit a KMeans instance to perform the clustering operation. Make sure that you also obtain the cluster assignments for the general demographics data, since you'll be using them in the final Step 3.3.

In [52]:
```
from sklearn.cluster import MiniBatchKMeans
from sklearn.cluster import KMeans
# Referenced: https://www.statology.org/k-means-clustering-in-python/

scores = []
num_clusters = list(range(10, 25))

for k in num_clusters:
```

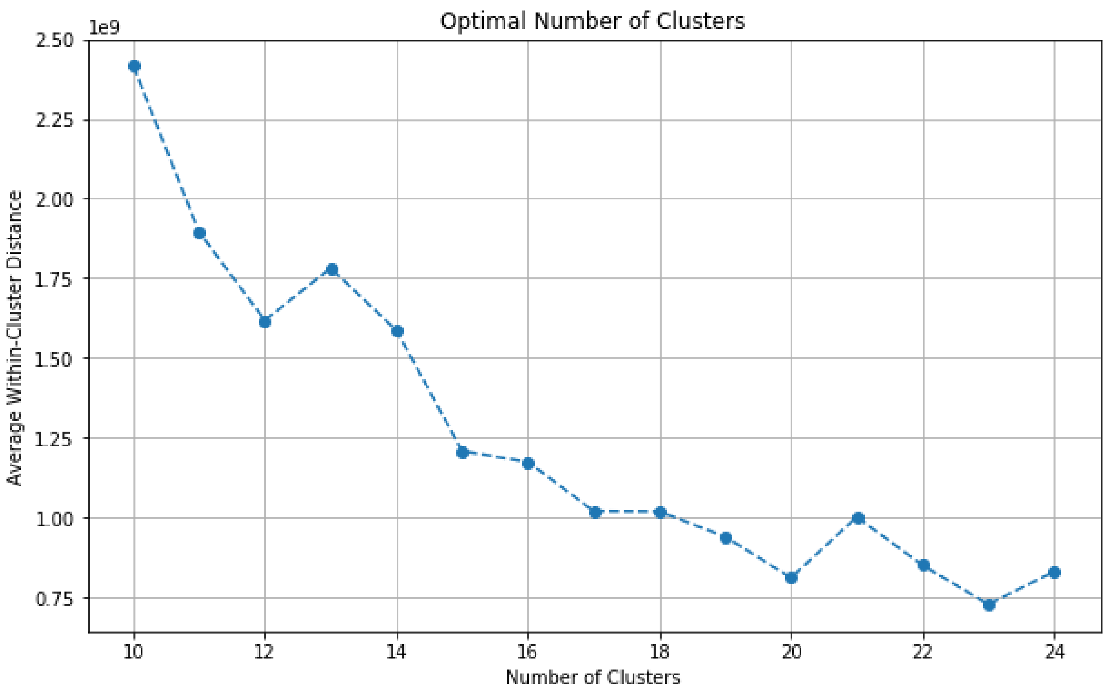
```
    # run k-means clustering on the data and...
    # use MiniBatchKMeans to speed up run time
    kmeans = MiniBatchKMeans(n_clusters=k)
    kmeans_model = kmeans.fit(transformed_pca)
    # compute the average within-cluster distances.
    score = np.abs(kmeans_model.score(transformed_pca))
    scores.append(score)

print('Clustering complete.')
```

Clustering complete.

In [66]:
```
# Was struggling to get the clustering to work so tried with a subset
# Maximum number of clusters
#clusters = 20
# Try different cluster counts from 2 up to the maximum number of clusters
#cluster_counts = range(2, clusters + 1)
# Create a subset of the transformed data for faster processing - Workspace is timing o
#subset_size = 10000
#subset_data = transformed_pca[:subset_size]
# Initialize a list to store the average within-cluster distances for each cluster coun
#avg_distances = []

#for num_clusters in cluster_counts:
#    # Initialize KMeans model with the current number of clusters
#    kmeans_model = KMeans(n_clusters=num_clusters, n_init=10)

#    # Fit the KMeans model to the subset of the PCA-transformed data
#    kmeans_model.fit(subset_data)

    # Compute the average within-cluster distances and store it in the list
#    avg_distance = -kmeans_model.score(subset_data) / subset_data.shape[0]  # Convert
#    avg_distances.append(avg_distance)

# Find the cluster count with the smallest average distance (largest score value)
#selected_num_clusters = cluster_counts[avg_distances.index(max(avg_distances))]

# Re-initialize KMeans model with the selected number of clusters
#kmeans_model = KMeans(n_clusters=selected_num_clusters, random_state=42)

# Fit the KMeans model to the PCA-transformed data
#kmeans_model.fit(transformed_pca)

# Obtain cluster predictions for the general population demographics data
#cluster_predictions = kmeans_model.predict(transformed_pca)

#print(selected_num_clusters)
```

2

In [53]:
```python
# Investigate the change in within-cluster distance across number of clusters.
# HINT: Use matplotlib's plot function to visualize this relationship.

# Plot the average within-cluster distances against the number of clusters
num_clusters = list(range(10, 25))

plt.figure(figsize=(10, 6))
plt.plot(num_clusters, scores, marker='o', linestyle='--')
plt.xlabel('Number of Clusters')
plt.ylabel('Average Within-Cluster Distance')
plt.title('Optimal Number of Clusters')
plt.grid(True)
plt.show()
```



In [54]:
```python
# Re-fit the k-means model with the selected number of clusters and obtain
# cluster predictions for the general population demographics data.
kmeans = KMeans(15)
kmeans_model = kmeans.fit(transformed_pca)
general_predictions = kmeans_model.predict(transformed_pca)

print("cluster predictions complete")
```

```
cluster predictions complete
```

1.3.2 Discussion 3.1: Apply Clustering to General Population

A subset of data was utilized to perform the predictions with 10 clusters, as the workspace kept timing out. An elbow method was used to plot the number of clusters against the scores.

1.3.3 Step 3.2: Apply All Steps to the Customer Data

Now that you have clusters and cluster centers for the general population, it's time to see how the customer data maps on to those clusters. Take care to not confuse this for re-fitting all of the models to the customer data. Instead, you're going to use the fits from the general population to clean, transform, and cluster the customer data. In the last step of the project, you will interpret how the general population fits apply to the customer data.

- Don't forget when loading in the customers data, that it is semicolon (`;`) delimited.
- Apply the same feature wrangling, selection, and engineering steps to the customer demographics using the `clean_data()` function you created earlier. (You can assume that the customer demographics data has similar meaning behind missing data patterns as the general demographics data.)
- Use the sklearn objects from the general demographics data, and apply their transformations to the customers data. That is, you should not be using a `.fit()` or `.fit_transform()` method to re-fit the old objects, nor should you be creating new sklearn objects! Carry the data through the feature scaling, PCA, and clustering steps, obtaining cluster assignments for all of the data in the customer demographics data.

In [58]:
```
# Load in the customer demographics data.
customers = pd.read_csv('Udacity_CUSTOMERS_Subset.csv', sep=';')
customers
```

Out[58]:
```
    AGER_TYP  ALTERSKATEGORIE_GROB  ANREDE_KZ  CJT_GESAMTTYP  \
0          2                     4          1            5.0
1         -1                     4          1            NaN
2         -1                     4          2            2.0
3          1                     4          1            2.0
4         -1                     3          1            6.0
5          1                     3          1            4.0
6          2                     4          1            2.0
7          1                     4          1            2.0
8          2                     4          2            1.0
9          1                     3          1            3.0
10        -1                     3          2            5.0
11         1                     4          1            3.0
12        -1                     4          1            5.0
13        -1                     3          1            6.0
14         2                     4          2            2.0
15         2                     3          1            3.0
16         1                     4          1            2.0
17        -1                     4          1            5.0
18        -1                     2          1            4.0
19        -1                     4          2            4.0
```

20	-1	4	1	6.0
21	2	4	1	1.0
22	2	1	2	2.0
23	-1	3	1	6.0
24	2	4	2	2.0
25	-1	3	2	3.0
26	1	4	2	2.0
27	-1	3	1	3.0
28	2	4	2	2.0
29	0	3	2	4.0
...	...	...	...	...
191622	2	4	1	2.0
191623	1	4	1	2.0
191624	1	4	1	2.0
191625	2	4	2	4.0
191626	-1	2	1	4.0
191627	3	3	2	2.0
191628	1	4	1	2.0
191629	2	4	2	1.0
191630	2	3	1	2.0
191631	2	4	2	1.0
191632	0	3	1	5.0
191633	1	4	1	3.0
191634	2	3	1	1.0
191635	2	4	2	2.0
191636	2	4	1	1.0
191637	-1	3	2	6.0
191638	1	4	1	6.0
191639	-1	3	1	5.0
191640	3	3	1	4.0
191641	1	4	1	2.0
191642	2	4	2	2.0
191643	2	4	1	5.0
191644	2	4	2	6.0
191645	2	4	1	5.0
191646	3	2	2	2.0
191647	1	3	1	4.0
191648	-1	4	2	2.0
191649	2	4	1	2.0
191650	3	3	2	4.0
191651	3	2	1	2.0

	FINANZ_MINIMALIST	FINANZ_SPARER	FINANZ_VORSORGER	FINANZ_ANLEGER	\
0	5	1	5	1	
1	5	1	5	1	
2	5	1	5	1	
3	5	1	5	2	
4	3	1	4	4	

5	5	1	5	1
6	5	1	5	1
7	5	1	5	1
8	2	2	5	1
9	5	2	4	1
10	4	2	4	4
11	5	1	5	1
12	5	2	4	3
13	5	2	4	2
14	3	1	5	1
15	5	1	5	1
16	5	1	5	1
17	4	3	1	4
18	2	4	2	2
19	3	2	4	3
20	5	3	4	2
21	3	1	5	2
22	5	1	5	1
23	3	3	4	1
24	4	1	5	2
25	5	2	4	4
26	5	1	5	2
27	5	2	3	1
28	3	1	5	1
29	3	1	5	1
...	...	...	...	...
191622	5	1	5	1
191623	5	1	5	1
191624	5	1	5	1
191625	5	1	5	2
191626	3	3	3	1
191627	2	1	5	1
191628	5	1	5	1
191629	2	1	5	1
191630	5	1	5	1
191631	2	1	5	1
191632	4	1	5	1
191633	5	1	5	1
191634	5	1	5	1
191635	5	1	5	2
191636	5	1	5	1
191637	5	2	4	2
191638	3	1	5	1
191639	4	3	3	2
191640	4	1	5	1
191641	5	1	5	1
191642	2	1	5	2
191643	5	1	5	1

191644	2	1	5	1
191645	5	1	5	1
191646	2	1	5	1
191647	5	1	5	1
191648	5	1	5	2
191649	5	1	5	1
191650	2	1	5	1
191651	5	1	5	1

	FINANZ_UNAUFFAELLIGER	FINANZ_HAUSBAUER	...	PLZ8_ANTG1 \
0	2	2	...	3.0
1	3	2	...	NaN
2	4	4	...	2.0
3	1	2	...	3.0
4	5	2	...	2.0
5	2	3	...	2.0
6	1	2	...	3.0
7	2	2	...	3.0
8	1	5	...	1.0
9	3	1	...	3.0
10	3	1	...	NaN
11	3	2	...	4.0
12	2	1	...	3.0
13	4	1	...	3.0
14	2	5	...	1.0
15	1	2	...	3.0
16	3	2	...	4.0
17	5	1	...	2.0
18	3	3	...	1.0
19	3	3	...	4.0
20	3	1	...	4.0
21	1	4	...	1.0
22	1	2	...	3.0
23	2	2	...	1.0
24	2	3	...	3.0
25	2	1	...	3.0
26	1	3	...	3.0
27	2	1	...	2.0
28	1	5	...	1.0
29	2	5	...	1.0
...	...	...	...	...
191622	1	2	...	3.0
191623	3	2	...	2.0
191624	4	2	...	2.0
191625	1	2	...	3.0
191626	3	2	...	2.0
191627	2	5	...	1.0
191628	1	2	...	3.0

```
191629                      2                    5  ...              2.0
191630                      1                    2  ...              2.0
191631                      2                    5  ...              2.0
191632                      1                    4  ...              4.0
191633                      4                    2  ...              4.0
191634                      2                    3  ...              3.0
191635                      1                    3  ...              3.0
191636                      1                    2  ...              3.0
191637                      3                    1  ...              2.0
191638                      1                    4  ...              0.0
191639                      4                    1  ...              4.0
191640                      2                    4  ...              2.0
191641                      2                    2  ...              4.0
191642                      1                    5  ...              0.0
191643                      3                    2  ...              4.0
191644                      2                    5  ...              1.0
191645                      3                    2  ...              2.0
191646                      2                    5  ...              1.0
191647                      1                    2  ...              2.0
191648                      2                    3  ...              NaN
191649                      1                    2  ...              3.0
191650                      2                    5  ...              3.0
191651                      1                    2  ...              3.0

        PLZ8_ANTG2  PLZ8_ANTG3  PLZ8_ANTG4  PLZ8_BAUMAX  PLZ8_HHZ  PLZ8_GBZ  \
0              3.0         1.0         0.0          1.0       5.0       5.0
1              NaN         NaN         NaN          NaN       NaN       NaN
2              3.0         3.0         1.0          3.0       3.0       2.0
3              2.0         1.0         0.0          1.0       3.0       4.0
4              4.0         2.0         1.0          2.0       3.0       3.0
5              3.0         2.0         1.0          1.0       5.0       5.0
6              2.0         1.0         0.0          1.0       5.0       5.0
7              3.0         1.0         1.0          1.0       3.0       3.0
8              4.0         2.0         1.0          5.0       5.0       4.0
9              3.0         1.0         0.0          1.0       5.0       5.0
10             NaN         NaN         NaN          NaN       NaN       NaN
11             2.0         1.0         0.0          1.0       3.0       3.0
12             1.0         0.0         0.0          1.0       4.0       5.0
13             3.0         1.0         0.0          1.0       4.0       4.0
14             4.0         3.0         1.0          3.0       5.0       3.0
15             2.0         1.0         0.0          1.0       5.0       5.0
16             1.0         0.0         0.0          1.0       3.0       4.0
17             3.0         2.0         1.0          1.0       5.0       5.0
18             3.0         3.0         2.0          5.0       3.0       1.0
19             2.0         1.0         0.0          1.0       3.0       4.0
20             1.0         0.0         0.0          1.0       3.0       4.0
21             4.0         3.0         1.0          3.0       4.0       3.0
22             2.0         0.0         0.0          1.0       4.0       5.0
```

23	3.0	3.0	2.0	5.0	5.0	2.0
24	3.0	1.0	0.0	1.0	3.0	3.0
25	2.0	1.0	0.0	1.0	3.0	4.0
26	3.0	1.0	0.0	1.0	5.0	5.0
27	4.0	2.0	1.0	2.0	4.0	3.0
28	3.0	3.0	2.0	4.0	5.0	3.0
29	4.0	3.0	2.0	5.0	3.0	2.0
...	...	...	...	...	...	...
191622	3.0	1.0	0.0	1.0	3.0	3.0
191623	4.0	2.0	1.0	2.0	4.0	3.0
191624	3.0	2.0	1.0	1.0	4.0	4.0
191625	2.0	1.0	1.0	1.0	5.0	5.0
191626	3.0	2.0	1.0	1.0	4.0	4.0
191627	3.0	2.0	2.0	4.0	5.0	3.0
191628	2.0	1.0	1.0	1.0	5.0	5.0
191629	4.0	2.0	1.0	5.0	3.0	2.0
191630	3.0	1.0	1.0	1.0	5.0	5.0
191631	4.0	2.0	1.0	2.0	4.0	3.0
191632	2.0	0.0	0.0	1.0	3.0	4.0
191633	2.0	0.0	0.0	1.0	4.0	4.0
191634	3.0	0.0	0.0	1.0	3.0	4.0
191635	3.0	1.0	1.0	1.0	3.0	3.0
191636	2.0	1.0	0.0	1.0	3.0	3.0
191637	3.0	1.0	0.0	1.0	3.0	3.0
191638	3.0	3.0	2.0	5.0	4.0	1.0
191639	1.0	0.0	0.0	1.0	2.0	3.0
191640	3.0	2.0	1.0	1.0	5.0	5.0
191641	2.0	1.0	0.0	1.0	4.0	5.0
191642	0.0	0.0	2.0	5.0	3.0	1.0
191643	2.0	0.0	0.0	1.0	5.0	5.0
191644	4.0	3.0	1.0	5.0	4.0	3.0
191645	3.0	2.0	1.0	1.0	3.0	3.0
191646	4.0	2.0	0.0	5.0	1.0	1.0
191647	4.0	2.0	1.0	2.0	5.0	4.0
191648	NaN	NaN	NaN	NaN	NaN	NaN
191649	2.0	2.0	1.0	1.0	5.0	5.0
191650	2.0	1.0	1.0	1.0	2.0	3.0
191651	2.0	0.0	0.0	1.0	4.0	5.0

	ARBEIT	ORTSGR_KLS9	RELAT_AB
0	1.0	2.0	1.0
1	NaN	NaN	NaN
2	3.0	5.0	3.0
3	1.0	3.0	1.0
4	3.0	5.0	1.0
5	3.0	7.0	5.0
6	2.0	3.0	2.0
7	3.0	4.0	3.0

8	3.0	8.0	3.0
9	3.0	6.0	4.0
10	NaN	NaN	NaN
11	2.0	5.0	1.0
12	1.0	1.0	1.0
13	4.0	5.0	5.0
14	3.0	8.0	5.0
15	3.0	6.0	1.0
16	3.0	5.0	3.0
17	1.0	3.0	1.0
18	3.0	8.0	5.0
19	3.0	5.0	5.0
20	4.0	6.0	5.0
21	1.0	3.0	1.0
22	1.0	2.0	1.0
23	3.0	9.0	5.0
24	2.0	6.0	3.0
25	1.0	3.0	1.0
26	1.0	2.0	1.0
27	3.0	7.0	4.0
28	3.0	7.0	5.0
29	3.0	8.0	5.0
...	...	...	...
191622	2.0	2.0	2.0
191623	2.0	4.0	3.0
191624	2.0	5.0	1.0
191625	2.0	4.0	3.0
191626	4.0	8.0	5.0
191627	3.0	7.0	3.0
191628	2.0	4.0	3.0
191629	2.0	5.0	1.0
191630	4.0	7.0	5.0
191631	4.0	7.0	5.0
191632	3.0	1.0	3.0
191633	3.0	5.0	2.0
191634	1.0	2.0	1.0
191635	3.0	5.0	3.0
191636	4.0	5.0	4.0
191637	3.0	7.0	5.0
191638	3.0	7.0	3.0
191639	4.0	5.0	4.0
191640	3.0	7.0	4.0
191641	2.0	1.0	2.0
191642	4.0	9.0	3.0
191643	2.0	4.0	1.0
191644	3.0	5.0	5.0
191645	3.0	5.0	3.0
191646	3.0	9.0	5.0

```
191647     3.0          8.0        5.0
191648     1.0          4.0        1.0
191649     3.0          7.0        5.0
191650     3.0          4.0        4.0
191651     1.0          3.0        1.0

[191652 rows x 85 columns]
```

In [59]: customers_cleaned = clean_data(customers)

```
missing removed
column drop successful
Number of rows before dropping columns and rows: 191652
Number of rows after filtering rows: 139434
Step 1 engineering complete.
Step 2 engineering complete.
Columns re-engineered and dropped.
Mixed variables dropped.
```

In [60]: customers_cleaned

Out[60]:

	ALTERSKATEGORIE_GROB	ANREDE_KZ	CJT_GESAMTTYP	FINANZ_MINIMALIST \
0	4.0	1	5.0	5
2	4.0	2	2.0	5
3	4.0	1	2.0	5
4	3.0	1	6.0	3
5	3.0	1	4.0	5
6	4.0	1	2.0	5
7	4.0	1	2.0	5
8	4.0	2	1.0	2
9	3.0	1	3.0	5
11	4.0	1	3.0	5
12	4.0	1	5.0	5
13	3.0	1	6.0	5
14	4.0	2	2.0	3
15	3.0	1	3.0	5
16	4.0	1	2.0	5
17	4.0	1	5.0	4
18	2.0	1	4.0	2
19	4.0	2	4.0	3
20	4.0	1	6.0	5
21	4.0	1	1.0	3
22	1.0	2	2.0	5
23	3.0	1	6.0	3
24	4.0	2	2.0	4
25	3.0	2	3.0	5
26	4.0	2	2.0	5
27	3.0	1	3.0	5

28	4.0	2	2.0	3
29	3.0	2	4.0	3
30	3.0	2	2.0	4
31	3.0	1	2.0	5
...	...	...	...	...
191621	4.0	2	2.0	3
191622	4.0	1	2.0	5
191623	4.0	1	2.0	5
191624	4.0	1	2.0	5
191625	4.0	2	4.0	5
191626	2.0	1	4.0	3
191627	3.0	2	2.0	2
191628	4.0	1	2.0	5
191629	4.0	2	1.0	2
191630	3.0	1	2.0	5
191631	4.0	2	1.0	2
191632	3.0	1	5.0	4
191633	4.0	1	3.0	5
191634	3.0	1	1.0	5
191635	4.0	2	2.0	5
191636	4.0	1	1.0	5
191637	3.0	2	6.0	5
191638	4.0	1	6.0	3
191639	3.0	1	5.0	4
191640	3.0	1	4.0	4
191641	4.0	1	2.0	5
191642	4.0	2	2.0	2
191643	4.0	1	5.0	5
191644	4.0	2	6.0	2
191645	4.0	1	5.0	5
191646	2.0	2	2.0	2
191647	3.0	1	4.0	5
191649	4.0	1	2.0	5
191650	3.0	2	4.0	2
191651	2.0	1	2.0	5

	FINANZ_SPARER	FINANZ_VORSORGER	FINANZ_ANLEGER	\
0	1	5	1	
2	1	5	1	
3	1	5	2	
4	1	4	4	
5	1	5	1	
6	1	5	1	
7	1	5	1	
8	2	5	1	
9	2	4	1	
11	1	5	1	
12	2	4	3	

13	2	4	2
14	1	5	1
15	1	5	1
16	1	5	1
17	3	1	4
18	4	2	2
19	2	4	3
20	3	4	2
21	1	5	2
22	1	5	1
23	3	4	1
24	1	5	2
25	2	4	4
26	1	5	2
27	2	3	1
28	1	5	1
29	1	5	1
30	1	5	1
31	2	3	3
...	...	...	...
191621	1	5	2
191622	1	5	1
191623	1	5	1
191624	1	5	1
191625	1	5	2
191626	3	3	1
191627	1	5	1
191628	1	5	1
191629	1	5	1
191630	1	5	1
191631	1	5	1
191632	1	5	1
191633	1	5	1
191634	1	5	1
191635	1	5	2
191636	1	5	1
191637	2	4	2
191638	1	5	1
191639	3	3	2
191640	1	5	1
191641	1	5	1
191642	1	5	2
191643	1	5	1
191644	1	5	1
191645	1	5	1
191646	1	5	1
191647	1	5	1
191649	1	5	1

```
191650             1              5              1
191651             1              5              1

        FINANZ_UNAUFFAELLIGER  FINANZ_HAUSBAUER  FINANZTYP  \
0                           2                 2          2
2                           4                 4          2
3                           1                 2          6
4                           5                 2          2
5                           2                 3          5
6                           1                 2          2
7                           2                 2          5
8                           1                 5          5
9                           3                 1          2
11                          3                 2          2
12                          2                 1          6
13                          4                 1          3
14                          2                 5          5
15                          1                 2          5
16                          3                 2          5
17                          5                 1          3
18                          3                 3          1
19                          3                 3          2
20                          3                 1          3
21                          1                 4          6
22                          1                 2          2
23                          2                 2          5
24                          2                 3          2
25                          2                 1          6
26                          1                 3          2
27                          2                 1          5
28                          1                 5          6
29                          2                 5          5
30                          2                 2          2
31                          2                 1          6
...                       ...               ...        ...
191621                      1                 5          6
191622                      1                 2          2
191623                      3                 2          5
191624                      4                 2          5
191625                      1                 2          2
191626                      3                 2          5
191627                      2                 5          2
191628                      1                 2          5
191629                      2                 5          5
191630                      1                 2          5
191631                      2                 5          5
191632                      1                 4          5
191633                      4                 2          5
```

```
191634                 2               3          2
191635                 1               3          2
191636                 1               2          5
191637                 3               1          6
191638                 1               4          5
191639                 4               1          3
191640                 2               4          5
191641                 2               2          2
191642                 1               5          6
191643                 3               2          5
191644                 2               5          5
191645                 3               2          5
191646                 2               5          5
191647                 1               2          5
191649                 1               2          5
191650                 2               5          2
191651                 1               2          6

               ...          PLZ8_ANTG4  PLZ8_HHZ  PLZ8_GBZ  ARBEIT  \
0              ...                 0.0       5.0       5.0     1.0
2              ...                 1.0       3.0       2.0     3.0
3              ...                 0.0       3.0       4.0     1.0
4              ...                 1.0       3.0       3.0     3.0
5              ...                 1.0       5.0       5.0     3.0
6              ...                 0.0       5.0       5.0     2.0
7              ...                 1.0       3.0       3.0     3.0
8              ...                 1.0       5.0       4.0     3.0
9              ...                 0.0       5.0       5.0     3.0
11             ...                 0.0       3.0       3.0     2.0
12             ...                 0.0       4.0       5.0     1.0
13             ...                 0.0       4.0       4.0     4.0
14             ...                 1.0       5.0       3.0     3.0
15             ...                 0.0       5.0       5.0     3.0
16             ...                 0.0       3.0       4.0     3.0
17             ...                 1.0       5.0       5.0     1.0
18             ...                 2.0       3.0       1.0     3.0
19             ...                 0.0       3.0       4.0     3.0
20             ...                 0.0       3.0       4.0     4.0
21             ...                 1.0       4.0       3.0     1.0
22             ...                 0.0       4.0       5.0     1.0
23             ...                 2.0       5.0       2.0     3.0
24             ...                 0.0       3.0       3.0     2.0
25             ...                 0.0       3.0       4.0     1.0
26             ...                 0.0       5.0       5.0     1.0
27             ...                 1.0       4.0       3.0     3.0
28             ...                 2.0       5.0       3.0     3.0
29             ...                 2.0       3.0       2.0     3.0
30             ...                 0.0       5.0       5.0     2.0
```

```
31                  ...                    0.0       4.0       5.0     2.0
...                 ...                    ...       ...       ...     ...
191621              ...                    1.0       5.0       4.0     4.0
191622              ...                    0.0       3.0       3.0     2.0
191623              ...                    1.0       4.0       3.0     2.0
191624              ...                    1.0       4.0       4.0     2.0
191625              ...                    1.0       5.0       5.0     2.0
191626              ...                    1.0       4.0       4.0     4.0
191627              ...                    2.0       5.0       3.0     3.0
191628              ...                    1.0       5.0       5.0     2.0
191629              ...                    1.0       3.0       2.0     2.0
191630              ...                    1.0       5.0       5.0     4.0
191631              ...                    1.0       4.0       3.0     4.0
191632              ...                    0.0       3.0       4.0     3.0
191633              ...                    0.0       4.0       4.0     3.0
191634              ...                    0.0       3.0       4.0     1.0
191635              ...                    1.0       3.0       3.0     3.0
191636              ...                    0.0       3.0       3.0     4.0
191637              ...                    0.0       3.0       3.0     3.0
191638              ...                    2.0       4.0       1.0     3.0
191639              ...                    0.0       2.0       3.0     4.0
191640              ...                    1.0       5.0       5.0     3.0
191641              ...                    0.0       4.0       5.0     2.0
191642              ...                    2.0       3.0       1.0     4.0
191643              ...                    0.0       5.0       5.0     2.0
191644              ...                    1.0       4.0       3.0     3.0
191645              ...                    1.0       3.0       3.0     3.0
191646              ...                    0.0       1.0       1.0     3.0
191647              ...                    1.0       5.0       4.0     3.0
191649              ...                    1.0       5.0       5.0     3.0
191650              ...                    1.0       2.0       3.0     3.0
191651              ...                    0.0       4.0       5.0     1.0

        ORTSGR_KLS9  RELAT_AB  DECADE  MOVEMENT  CAMEO_INTL_2015_wealth  \
0               2.0       1.0     2.0       0.0                     1.0
2               5.0       3.0     2.0       0.0                     3.0
3               3.0       1.0     1.0       1.0                     2.0
4               5.0       1.0     4.0       1.0                     4.0
5               7.0       5.0     2.0       0.0                     3.0
6               3.0       2.0     2.0       0.0                     2.0
7               4.0       3.0     2.0       0.0                     1.0
8               8.0       3.0     1.0       1.0                     5.0
9               6.0       4.0     4.0       0.0                     1.0
11              5.0       1.0     2.0       0.0                     1.0
12              1.0       1.0     4.0       1.0                     2.0
13              5.0       5.0     4.0       1.0                     4.0
14              8.0       5.0     1.0       1.0                     5.0
15              6.0       1.0     3.0       0.0                     1.0
```

16	5.0	3.0	2.0	0.0	1.0
17	3.0	1.0	6.0	0.0	1.0
18	8.0	5.0	6.0	1.0	4.0
19	5.0	5.0	4.0	1.0	2.0
20	6.0	5.0	5.0	0.0	3.0
21	3.0	1.0	3.0	1.0	4.0
22	2.0	1.0	1.0	1.0	2.0
23	9.0	5.0	4.0	1.0	5.0
24	6.0	3.0	2.0	1.0	3.0
25	3.0	1.0	5.0	0.0	2.0
26	2.0	1.0	2.0	1.0	2.0
27	7.0	4.0	4.0	0.0	4.0
28	7.0	5.0	1.0	1.0	5.0
29	8.0	5.0	3.0	1.0	5.0
30	3.0	3.0	3.0	0.0	3.0
31	1.0	2.0	4.0	0.0	2.0
...	...	...	...	...	...
191621	6.0	5.0	3.0	1.0	4.0
191622	2.0	2.0	1.0	1.0	4.0
191623	4.0	3.0	3.0	0.0	1.0
191624	5.0	1.0	4.0	0.0	5.0
191625	4.0	3.0	2.0	0.0	2.0
191626	8.0	5.0	4.0	1.0	2.0
191627	7.0	3.0	2.0	1.0	5.0
191628	4.0	3.0	3.0	0.0	2.0
191629	5.0	1.0	1.0	1.0	5.0
191630	7.0	5.0	2.0	0.0	5.0
191631	7.0	5.0	2.0	1.0	4.0
191632	1.0	3.0	4.0	1.0	1.0
191633	5.0	2.0	3.0	0.0	1.0
191634	2.0	1.0	3.0	1.0	3.0
191635	5.0	3.0	1.0	1.0	4.0
191636	5.0	4.0	1.0	0.0	3.0
191637	7.0	5.0	4.0	0.0	2.0
191638	7.0	3.0	3.0	1.0	5.0
191639	5.0	4.0	5.0	1.0	2.0
191640	7.0	4.0	4.0	0.0	1.0
191641	1.0	2.0	2.0	0.0	1.0
191642	9.0	3.0	2.0	1.0	5.0
191643	4.0	1.0	3.0	0.0	1.0
191644	5.0	5.0	4.0	0.0	4.0
191645	5.0	3.0	4.0	1.0	4.0
191646	9.0	5.0	4.0	1.0	1.0
191647	8.0	5.0	2.0	0.0	1.0
191649	7.0	5.0	2.0	0.0	2.0
191650	4.0	4.0	4.0	1.0	2.0
191651	3.0	1.0	2.0	1.0	3.0

	CAMEO_INTL_2015_life_stage
0	3.0
2	4.0
3	4.0
4	1.0
5	4.0
6	3.0
7	5.0
8	5.0
9	5.0
11	4.0
12	2.0
13	3.0
14	1.0
15	5.0
16	5.0
17	4.0
18	1.0
19	4.0
20	3.0
21	3.0
22	4.0
23	1.0
24	3.0
25	2.0
26	5.0
27	4.0
28	4.0
29	1.0
30	2.0
31	4.0
...	...
191621	1.0
191622	3.0
191623	4.0
191624	4.0
191625	5.0
191626	4.0
191627	5.0
191628	2.0
191629	5.0
191630	1.0
191631	4.0
191632	4.0
191633	4.0
191634	3.0
191635	5.0
191636	4.0

```
191637                        5.0
191638                        2.0
191639                        2.0
191640                        5.0
191641                        4.0
191642                        5.0
191643                        5.0
191644                        1.0
191645                        3.0
191646                        4.0
191647                        4.0
191649                        4.0
191650                        4.0
191651                        3.0

[139434 rows x 75 columns]
```

In [61]:
```
col_difference = list(set.difference(set(customers_cleaned.columns),set(azdias_cleaned.
col_difference
```

Out[61]:
```
['MOVEMENT', 'DECADE']
```

In [62]:
```
customers_cleaned.drop(col_difference, axis=1, inplace=True)
customers_cleaned
```

Out[62]:
```
        ALTERSKATEGORIE_GROB  ANREDE_KZ  CJT_GESAMTTYP  FINANZ_MINIMALIST  \
0                        4.0          1            5.0                  5
2                        4.0          2            2.0                  5
3                        4.0          1            2.0                  5
4                        3.0          1            6.0                  3
5                        3.0          1            4.0                  5
6                        4.0          1            2.0                  5
7                        4.0          1            2.0                  5
8                        4.0          2            1.0                  2
9                        3.0          1            3.0                  5
11                       4.0          1            3.0                  5
12                       4.0          1            5.0                  5
13                       3.0          1            6.0                  5
14                       4.0          2            2.0                  3
15                       3.0          1            3.0                  5
16                       4.0          1            2.0                  5
17                       4.0          1            5.0                  4
18                       2.0          1            4.0                  2
19                       4.0          2            4.0                  3
20                       4.0          1            6.0                  5
21                       4.0          1            1.0                  3
22                       1.0          2            2.0                  5
23                       3.0          1            6.0                  3
24                       4.0          2            2.0                  4
```

25	3.0	2	3.0	5
26	4.0	2	2.0	5
27	3.0	1	3.0	5
28	4.0	2	2.0	3
29	3.0	2	4.0	3
30	3.0	2	2.0	4
31	3.0	1	2.0	5
...	...	...	...	...
191621	4.0	2	2.0	3
191622	4.0	1	2.0	5
191623	4.0	1	2.0	5
191624	4.0	1	2.0	5
191625	4.0	2	4.0	5
191626	2.0	1	4.0	3
191627	3.0	2	2.0	2
191628	4.0	1	2.0	5
191629	4.0	2	1.0	2
191630	3.0	1	2.0	5
191631	4.0	2	1.0	2
191632	3.0	1	5.0	4
191633	4.0	1	3.0	5
191634	3.0	1	1.0	5
191635	4.0	2	2.0	5
191636	4.0	1	1.0	5
191637	3.0	2	6.0	5
191638	4.0	1	6.0	3
191639	3.0	1	5.0	4
191640	3.0	1	4.0	4
191641	4.0	1	2.0	5
191642	4.0	2	2.0	2
191643	4.0	1	5.0	5
191644	4.0	2	6.0	2
191645	4.0	1	5.0	5
191646	2.0	2	2.0	2
191647	3.0	1	4.0	5
191649	4.0	1	2.0	5
191650	3.0	2	4.0	2
191651	2.0	1	2.0	5

	FINANZ_SPARER	FINANZ_VORSORGER	FINANZ_ANLEGER	\
0	1	5	1	
2	1	5	1	
3	1	5	2	
4	1	4	4	
5	1	5	1	
6	1	5	1	
7	1	5	1	
8	2	5	1	

9	2	4	1
11	1	5	1
12	2	4	3
13	2	4	2
14	1	5	1
15	1	5	1
16	1	5	1
17	3	1	4
18	4	2	2
19	2	4	3
20	3	4	2
21	1	5	2
22	1	5	1
23	3	4	1
24	1	5	2
25	2	4	4
26	1	5	2
27	2	3	1
28	1	5	1
29	1	5	1
30	1	5	1
31	2	3	3
...	...	...	...
191621	1	5	2
191622	1	5	1
191623	1	5	1
191624	1	5	1
191625	1	5	2
191626	3	3	1
191627	1	5	1
191628	1	5	1
191629	1	5	1
191630	1	5	1
191631	1	5	1
191632	1	5	1
191633	1	5	1
191634	1	5	1
191635	1	5	2
191636	1	5	1
191637	2	4	2
191638	1	5	1
191639	3	3	2
191640	1	5	1
191641	1	5	1
191642	1	5	2
191643	1	5	1
191644	1	5	1
191645	1	5	1

191646	1	5	1
191647	1	5	1
191649	1	5	1
191650	1	5	1
191651	1	5	1

	FINANZ_UNAUFFAELLIGER	FINANZ_HAUSBAUER	FINANZTYP \
0	2	2	2
2	4	4	2
3	1	2	6
4	5	2	2
5	2	3	5
6	1	2	2
7	2	2	5
8	1	5	5
9	3	1	2
11	3	2	2
12	2	1	6
13	4	1	3
14	2	5	5
15	1	2	5
16	3	2	5
17	5	1	3
18	3	3	1
19	3	3	2
20	3	1	3
21	1	4	6
22	1	2	2
23	2	2	5
24	2	3	2
25	2	1	6
26	1	3	2
27	2	1	5
28	1	5	6
29	2	5	5
30	2	2	2
31	2	1	6
...	...	...	...
191621	1	5	6
191622	1	2	2
191623	3	2	5
191624	4	2	5
191625	1	2	2
191626	3	2	5
191627	2	5	2
191628	1	2	5
191629	2	5	5
191630	1	2	5

191631	2	5	5
191632	1	4	5
191633	4	2	5
191634	2	3	2
191635	1	3	2
191636	1	2	5
191637	3	1	6
191638	1	4	5
191639	4	1	3
191640	2	4	5
191641	2	2	2
191642	1	5	6
191643	3	2	5
191644	2	5	5
191645	3	2	5
191646	2	5	5
191647	1	2	5
191649	1	2	5
191650	2	5	2
191651	1	2	6

	...	PLZ8_ANTG2	PLZ8_ANTG3	PLZ8_ANTG4	\
0	...	3.0	1.0	0.0	
2	...	3.0	3.0	1.0	
3	...	2.0	1.0	0.0	
4	...	4.0	2.0	1.0	
5	...	3.0	2.0	1.0	
6	...	2.0	1.0	0.0	
7	...	3.0	1.0	1.0	
8	...	4.0	2.0	1.0	
9	...	3.0	1.0	0.0	
11	...	2.0	1.0	0.0	
12	...	1.0	0.0	0.0	
13	...	3.0	1.0	0.0	
14	...	4.0	3.0	1.0	
15	...	2.0	1.0	0.0	
16	...	1.0	0.0	0.0	
17	...	3.0	2.0	1.0	
18	...	3.0	3.0	2.0	
19	...	2.0	1.0	0.0	
20	...	1.0	0.0	0.0	
21	...	4.0	3.0	1.0	
22	...	2.0	0.0	0.0	
23	...	3.0	3.0	2.0	
24	...	3.0	1.0	0.0	
25	...	2.0	1.0	0.0	
26	...	3.0	1.0	0.0	
27	...	4.0	2.0	1.0	

```
28                ...                 3.0        3.0        2.0
29                ...                 4.0        3.0        2.0
30                ...                 2.0        1.0        0.0
31                ...                 2.0        1.0        0.0
...               ...                 ...        ...        ...
191621            ...                 2.0        2.0        1.0
191622            ...                 3.0        1.0        0.0
191623            ...                 4.0        2.0        1.0
191624            ...                 3.0        2.0        1.0
191625            ...                 2.0        1.0        1.0
191626            ...                 3.0        2.0        1.0
191627            ...                 3.0        2.0        2.0
191628            ...                 2.0        1.0        1.0
191629            ...                 4.0        2.0        1.0
191630            ...                 3.0        1.0        1.0
191631            ...                 4.0        2.0        1.0
191632            ...                 2.0        0.0        0.0
191633            ...                 2.0        0.0        0.0
191634            ...                 3.0        0.0        0.0
191635            ...                 3.0        1.0        1.0
191636            ...                 2.0        1.0        0.0
191637            ...                 3.0        1.0        0.0
191638            ...                 3.0        3.0        2.0
191639            ...                 1.0        0.0        0.0
191640            ...                 3.0        2.0        1.0
191641            ...                 2.0        1.0        0.0
191642            ...                 0.0        0.0        2.0
191643            ...                 2.0        0.0        0.0
191644            ...                 4.0        3.0        1.0
191645            ...                 3.0        2.0        1.0
191646            ...                 4.0        2.0        0.0
191647            ...                 4.0        2.0        1.0
191649            ...                 2.0        2.0        1.0
191650            ...                 2.0        1.0        1.0
191651            ...                 2.0        0.0        0.0

        PLZ8_HHZ  PLZ8_GBZ  ARBEIT  ORTSGR_KLS9  RELAT_AB  \
0            5.0       5.0     1.0          2.0       1.0
2            3.0       2.0     3.0          5.0       3.0
3            3.0       4.0     1.0          3.0       1.0
4            3.0       3.0     3.0          5.0       1.0
5            5.0       5.0     3.0          7.0       5.0
6            5.0       5.0     2.0          3.0       2.0
7            3.0       3.0     3.0          4.0       3.0
8            5.0       4.0     3.0          8.0       3.0
9            5.0       5.0     3.0          6.0       4.0
11           3.0       3.0     2.0          5.0       1.0
12           4.0       5.0     1.0          1.0       1.0
```

13	4.0	4.0	4.0	5.0	5.0
14	5.0	3.0	3.0	8.0	5.0
15	5.0	5.0	3.0	6.0	1.0
16	3.0	4.0	3.0	5.0	3.0
17	5.0	5.0	1.0	3.0	1.0
18	3.0	1.0	3.0	8.0	5.0
19	3.0	4.0	3.0	5.0	5.0
20	3.0	4.0	4.0	6.0	5.0
21	4.0	3.0	1.0	3.0	1.0
22	4.0	5.0	1.0	2.0	1.0
23	5.0	2.0	3.0	9.0	5.0
24	3.0	3.0	2.0	6.0	3.0
25	3.0	4.0	1.0	3.0	1.0
26	5.0	5.0	1.0	2.0	1.0
27	4.0	3.0	3.0	7.0	4.0
28	5.0	3.0	3.0	7.0	5.0
29	3.0	2.0	3.0	8.0	5.0
30	5.0	5.0	2.0	3.0	3.0
31	4.0	5.0	2.0	1.0	2.0
...	...	...	...	...	...
191621	5.0	4.0	4.0	6.0	5.0
191622	3.0	3.0	2.0	2.0	2.0
191623	4.0	3.0	2.0	4.0	3.0
191624	4.0	4.0	2.0	5.0	1.0
191625	5.0	5.0	2.0	4.0	3.0
191626	4.0	4.0	4.0	8.0	5.0
191627	5.0	3.0	3.0	7.0	3.0
191628	5.0	5.0	2.0	4.0	3.0
191629	3.0	2.0	2.0	5.0	1.0
191630	5.0	5.0	4.0	7.0	5.0
191631	4.0	3.0	4.0	7.0	5.0
191632	3.0	4.0	3.0	1.0	3.0
191633	4.0	4.0	3.0	5.0	2.0
191634	3.0	4.0	1.0	2.0	1.0
191635	3.0	3.0	3.0	5.0	3.0
191636	3.0	3.0	4.0	5.0	4.0
191637	3.0	3.0	3.0	7.0	5.0
191638	4.0	1.0	3.0	7.0	3.0
191639	2.0	3.0	4.0	5.0	4.0
191640	5.0	5.0	3.0	7.0	4.0
191641	4.0	5.0	2.0	1.0	2.0
191642	3.0	1.0	4.0	9.0	3.0
191643	5.0	5.0	2.0	4.0	1.0
191644	4.0	3.0	3.0	5.0	5.0
191645	3.0	3.0	3.0	5.0	3.0
191646	1.0	1.0	3.0	9.0	5.0
191647	5.0	4.0	3.0	8.0	5.0
191649	5.0	5.0	3.0	7.0	5.0

191650	2.0	3.0	3.0	4.0	4.0
191651	4.0	5.0	1.0	3.0	1.0

	CAMEO_INTL_2015_wealth	CAMEO_INTL_2015_life_stage
0	1.0	3.0
2	3.0	4.0
3	2.0	4.0
4	4.0	1.0
5	3.0	4.0
6	2.0	3.0
7	1.0	5.0
8	5.0	5.0
9	1.0	5.0
11	1.0	4.0
12	2.0	2.0
13	4.0	3.0
14	5.0	1.0
15	1.0	5.0
16	1.0	5.0
17	1.0	4.0
18	4.0	1.0
19	2.0	4.0
20	3.0	3.0
21	4.0	3.0
22	2.0	4.0
23	5.0	1.0
24	3.0	3.0
25	2.0	2.0
26	2.0	5.0
27	4.0	4.0
28	5.0	4.0
29	5.0	1.0
30	3.0	2.0
31	2.0	4.0
...	...	...
191621	4.0	1.0
191622	4.0	3.0
191623	1.0	4.0
191624	5.0	4.0
191625	2.0	5.0
191626	2.0	4.0
191627	5.0	5.0
191628	2.0	2.0
191629	5.0	5.0
191630	5.0	1.0
191631	4.0	4.0
191632	1.0	4.0
191633	1.0	4.0

```
191634                    3.0                         3.0
191635                    4.0                         5.0
191636                    3.0                         4.0
191637                    2.0                         5.0
191638                    5.0                         2.0
191639                    2.0                         2.0
191640                    1.0                         5.0
191641                    1.0                         4.0
191642                    5.0                         5.0
191643                    1.0                         5.0
191644                    4.0                         1.0
191645                    4.0                         3.0
191646                    1.0                         4.0
191647                    1.0                         4.0
191649                    2.0                         4.0
191650                    2.0                         4.0
191651                    3.0                         3.0

[139434 rows x 73 columns]
```

In [63]:
```python
# Apply preprocessing, feature transformation, and clustering from the general
# demographics onto the customer data, obtaining cluster predictions for the
# customer demographics data.
print('Number of NaNs remaining in data before imputer:', customers_cleaned.isnull().su
values = imputer.transform(customers_cleaned)
customers_cleaned = pd.DataFrame(values, columns=customers_cleaned.columns)
print('Number of NaNs remaining in data after imputer:', customers_cleaned.isnull().sum
```

```
Number of NaNs remaining in data before imputer: 105463
Number of NaNs remaining in data after imputer: 0
```

In [64]:
```python
# Scale the customers data
cust_scaled = scaler.transform(customers_cleaned)
cust_scaled = pd.DataFrame(cust_scaled, columns=customers_cleaned.columns)
# PCA
cust_pca = pd.DataFrame(pca.transform(cust_scaled))
# Predictions
cust_pred = kmeans_model.predict(cust_pca)
```

In [65]:
```python
cust_scaled
```

Out[65]:
```
   ALTERSKATEGORIE_GROB  ANREDE_KZ  CJT_GESAMTTYP  FINANZ_MINIMALIST  \
0              1.232533  -1.045218       0.859488           1.457527
1              1.232533   0.956738      -1.026509           1.457527
2              1.232533  -1.045218      -1.026509           1.457527
3              0.240781  -1.045218       1.488153          -0.056416
4              0.240781  -1.045218       0.230822           1.457527
5              1.232533  -1.045218      -1.026509           1.457527
```

6	1.232533	-1.045218	-1.026509	1.457527
7	1.232533	0.956738	-1.655175	-0.813387
8	0.240781	-1.045218	-0.397844	1.457527
9	1.232533	-1.045218	-0.397844	1.457527
10	1.232533	-1.045218	0.859488	1.457527
11	0.240781	-1.045218	1.488153	1.457527
12	1.232533	0.956738	-1.026509	-0.056416
13	0.240781	-1.045218	-0.397844	1.457527
14	1.232533	-1.045218	-1.026509	1.457527
15	1.232533	-1.045218	0.859488	0.700556
16	-0.750972	-1.045218	0.230822	-0.813387
17	1.232533	0.956738	0.230822	-0.056416
18	1.232533	-1.045218	1.488153	1.457527
19	1.232533	-1.045218	-1.655175	-0.056416
20	-1.742724	0.956738	-1.026509	1.457527
21	0.240781	-1.045218	1.488153	-0.056416
22	1.232533	0.956738	-1.026509	0.700556
23	0.240781	0.956738	-0.397844	1.457527
24	1.232533	0.956738	-1.026509	1.457527
25	0.240781	-1.045218	-0.397844	1.457527
26	1.232533	0.956738	-1.026509	-0.056416
27	0.240781	0.956738	0.230822	-0.056416
28	0.240781	0.956738	-1.026509	0.700556
29	0.240781	-1.045218	-1.026509	1.457527
...	...	...	...	...
139404	1.232533	0.956738	-1.026509	-0.056416
139405	1.232533	-1.045218	-1.026509	1.457527
139406	1.232533	-1.045218	-1.026509	1.457527
139407	1.232533	-1.045218	-1.026509	1.457527
139408	1.232533	0.956738	0.230822	1.457527
139409	-0.750972	-1.045218	0.230822	-0.056416
139410	0.240781	0.956738	-1.026509	-0.813387
139411	1.232533	-1.045218	-1.026509	1.457527
139412	1.232533	0.956738	-1.655175	-0.813387
139413	0.240781	-1.045218	-1.026509	1.457527
139414	1.232533	0.956738	-1.655175	-0.813387
139415	0.240781	-1.045218	0.859488	0.700556
139416	1.232533	-1.045218	-0.397844	1.457527
139417	0.240781	-1.045218	-1.655175	1.457527
139418	1.232533	0.956738	-1.026509	1.457527
139419	1.232533	-1.045218	-1.655175	1.457527
139420	0.240781	0.956738	1.488153	1.457527
139421	1.232533	-1.045218	1.488153	-0.056416
139422	0.240781	-1.045218	0.859488	0.700556
139423	0.240781	-1.045218	0.230822	0.700556
139424	1.232533	-1.045218	-1.026509	1.457527
139425	1.232533	0.956738	-1.026509	-0.813387
139426	1.232533	-1.045218	0.859488	1.457527

139427	1.232533	0.956738	1.488153	-0.813387
139428	1.232533	-1.045218	0.859488	1.457527
139429	-0.750972	0.956738	-1.026509	-0.813387
139430	0.240781	-1.045218	0.230822	1.457527
139431	1.232533	-1.045218	-1.026509	1.457527
139432	0.240781	0.956738	0.230822	-0.813387
139433	-0.750972	-1.045218	-1.026509	1.457527

	FINANZ_SPARER	FINANZ_VORSORGER	FINANZ_ANLEGER	\
0	-1.243244	1.209329	-1.329319	
1	-1.243244	1.209329	-1.329319	
2	-1.243244	1.209329	-0.675554	
3	-1.243244	0.452976	0.631976	
4	-1.243244	1.209329	-1.329319	
5	-1.243244	1.209329	-1.329319	
6	-1.243244	1.209329	-1.329319	
7	-0.560532	1.209329	-1.329319	
8	-0.560532	0.452976	-1.329319	
9	-1.243244	1.209329	-1.329319	
10	-0.560532	0.452976	-0.021789	
11	-0.560532	0.452976	-0.675554	
12	-1.243244	1.209329	-1.329319	
13	-1.243244	1.209329	-1.329319	
14	-1.243244	1.209329	-1.329319	
15	0.122179	-1.816084	0.631976	
16	0.804890	-1.059731	-0.675554	
17	-0.560532	0.452976	-0.021789	
18	0.122179	0.452976	-0.675554	
19	-1.243244	1.209329	-0.675554	
20	-1.243244	1.209329	-1.329319	
21	0.122179	0.452976	-1.329319	
22	-1.243244	1.209329	-0.675554	
23	-0.560532	0.452976	0.631976	
24	-1.243244	1.209329	-0.675554	
25	-0.560532	-0.303378	-1.329319	
26	-1.243244	1.209329	-1.329319	
27	-1.243244	1.209329	-1.329319	
28	-1.243244	1.209329	-1.329319	
29	-0.560532	-0.303378	-0.021789	
...	...	...	...	
139404	-1.243244	1.209329	-0.675554	
139405	-1.243244	1.209329	-1.329319	
139406	-1.243244	1.209329	-1.329319	
139407	-1.243244	1.209329	-1.329319	
139408	-1.243244	1.209329	-0.675554	
139409	0.122179	-0.303378	-1.329319	
139410	-1.243244	1.209329	-1.329319	
139411	-1.243244	1.209329	-1.329319	

139412	-1.243244	1.209329	-1.329319
139413	-1.243244	1.209329	-1.329319
139414	-1.243244	1.209329	-1.329319
139415	-1.243244	1.209329	-1.329319
139416	-1.243244	1.209329	-1.329319
139417	-1.243244	1.209329	-1.329319
139418	-1.243244	1.209329	-0.675554
139419	-1.243244	1.209329	-1.329319
139420	-0.560532	0.452976	-0.675554
139421	-1.243244	1.209329	-1.329319
139422	0.122179	-0.303378	-0.675554
139423	-1.243244	1.209329	-1.329319
139424	-1.243244	1.209329	-1.329319
139425	-1.243244	1.209329	-0.675554
139426	-1.243244	1.209329	-1.329319
139427	-1.243244	1.209329	-1.329319
139428	-1.243244	1.209329	-1.329319
139429	-1.243244	1.209329	-1.329319
139430	-1.243244	1.209329	-1.329319
139431	-1.243244	1.209329	-1.329319
139432	-1.243244	1.209329	-1.329319
139433	-1.243244	1.209329	-1.329319

	FINANZ_UNAUFFAELLIGER	FINANZ_HAUSBAUER	FINANZTYP	\
0	-0.587980	-0.794475	-0.900754	
1	0.757254	0.683452	-0.900754	
2	-1.260597	-0.794475	1.111445	
3	1.429871	-0.794475	-0.900754	
4	-0.587980	-0.055511	0.608395	
5	-1.260597	-0.794475	-0.900754	
6	-0.587980	-0.794475	0.608395	
7	-1.260597	1.422415	0.608395	
8	0.084637	-1.533438	-0.900754	
9	0.084637	-0.794475	-0.900754	
10	-0.587980	-1.533438	1.111445	
11	0.757254	-1.533438	-0.397704	
12	-0.587980	1.422415	0.608395	
13	-1.260597	-0.794475	0.608395	
14	0.084637	-0.794475	0.608395	
15	1.429871	-1.533438	-0.397704	
16	0.084637	-0.055511	-1.403804	
17	0.084637	-0.055511	-0.900754	
18	0.084637	-1.533438	-0.397704	
19	-1.260597	0.683452	1.111445	
20	-1.260597	-0.794475	-0.900754	
21	-0.587980	-0.794475	0.608395	
22	-0.587980	-0.055511	-0.900754	
23	-0.587980	-1.533438	1.111445	

24	-1.260597	-0.055511	-0.900754
25	-0.587980	-1.533438	0.608395
26	-1.260597	1.422415	1.111445
27	-0.587980	1.422415	0.608395
28	-0.587980	-0.794475	-0.900754
29	-0.587980	-1.533438	1.111445
...	...	...	...
139404	-1.260597	1.422415	1.111445
139405	-1.260597	-0.794475	-0.900754
139406	0.084637	-0.794475	0.608395
139407	0.757254	-0.794475	0.608395
139408	-1.260597	-0.794475	-0.900754
139409	0.084637	-0.794475	0.608395
139410	-0.587980	1.422415	-0.900754
139411	-1.260597	-0.794475	0.608395
139412	-0.587980	1.422415	0.608395
139413	-1.260597	-0.794475	0.608395
139414	-0.587980	1.422415	0.608395
139415	-1.260597	0.683452	0.608395
139416	0.757254	-0.794475	0.608395
139417	-0.587980	-0.055511	-0.900754
139418	-1.260597	-0.055511	-0.900754
139419	-1.260597	-0.794475	0.608395
139420	0.084637	-1.533438	1.111445
139421	-1.260597	0.683452	0.608395
139422	0.757254	-1.533438	-0.397704
139423	-0.587980	0.683452	0.608395
139424	-0.587980	-0.794475	-0.900754
139425	-1.260597	1.422415	1.111445
139426	0.084637	-0.794475	0.608395
139427	-0.587980	1.422415	0.608395
139428	0.084637	-0.794475	0.608395
139429	-0.587980	1.422415	0.608395
139430	-1.260597	-0.794475	0.608395
139431	-1.260597	-0.794475	0.608395
139432	-0.587980	1.422415	-0.900754
139433	-1.260597	-0.794475	1.111445

	...	PLZ8_ANTG2	PLZ8_ANTG3	PLZ8_ANTG4	\
0	...	0.230923	-0.647219	-1.031309	
1	...	0.230923	1.526751	0.443747	
2	...	-0.934519	-0.647219	-1.031309	
3	...	1.396366	0.439766	0.443747	
4	...	0.230923	0.439766	0.443747	
5	...	-0.934519	-0.647219	-1.031309	
6	...	0.230923	-0.647219	0.443747	
7	...	1.396366	0.439766	0.443747	
8	...	0.230923	-0.647219	-1.031309	

9	...	-0.934519	-0.647219	-1.031309
10	...	-2.099962	-1.734204	-1.031309
11	...	0.230923	-0.647219	-1.031309
12	...	1.396366	1.526751	0.443747
13	...	-0.934519	-0.647219	-1.031309
14	...	-2.099962	-1.734204	-1.031309
15	...	0.230923	0.439766	0.443747
16	...	0.230923	1.526751	1.918803
17	...	-0.934519	-0.647219	-1.031309
18	...	-2.099962	-1.734204	-1.031309
19	...	1.396366	1.526751	0.443747
20	...	-0.934519	-1.734204	-1.031309
21	...	0.230923	1.526751	1.918803
22	...	0.230923	-0.647219	-1.031309
23	...	-0.934519	-0.647219	-1.031309
24	...	0.230923	-0.647219	-1.031309
25	...	1.396366	0.439766	0.443747
26	...	0.230923	1.526751	1.918803
27	...	1.396366	1.526751	1.918803
28	...	-0.934519	-0.647219	-1.031309
29	...	-0.934519	-0.647219	-1.031309
...	...	...	...	...
139404	...	-0.934519	0.439766	0.443747
139405	...	0.230923	-0.647219	-1.031309
139406	...	1.396366	0.439766	0.443747
139407	...	0.230923	0.439766	0.443747
139408	...	-0.934519	-0.647219	0.443747
139409	...	0.230923	0.439766	0.443747
139410	...	0.230923	0.439766	1.918803
139411	...	-0.934519	-0.647219	0.443747
139412	...	1.396366	0.439766	0.443747
139413	...	0.230923	-0.647219	0.443747
139414	...	1.396366	0.439766	0.443747
139415	...	-0.934519	-1.734204	-1.031309
139416	...	-0.934519	-1.734204	-1.031309
139417	...	0.230923	-1.734204	-1.031309
139418	...	0.230923	-0.647219	0.443747
139419	...	-0.934519	-0.647219	-1.031309
139420	...	0.230923	-0.647219	-1.031309
139421	...	0.230923	1.526751	1.918803
139422	...	-2.099962	-1.734204	-1.031309
139423	...	0.230923	0.439766	0.443747
139424	...	-0.934519	-0.647219	-1.031309
139425	...	-3.265404	-1.734204	1.918803
139426	...	-0.934519	-1.734204	-1.031309
139427	...	1.396366	1.526751	0.443747
139428	...	0.230923	0.439766	0.443747
139429	...	1.396366	0.439766	-1.031309

139430	...	1.396366	0.439766	0.443747
139431	...	-0.934519	0.439766	0.443747
139432	...	-0.934519	-0.647219	0.443747
139433	...	-0.934519	-1.734204	-1.031309

	PLZ8_HHZ	PLZ8_GBZ	ARBEIT	ORTSGR_KLS9	RELAT_AB	\
0	1.527612	1.562070	-2.297862	-1.514868	-1.612883	
1	-0.674861	-1.332594	-0.176778	-0.134951	-0.055319	
2	-0.674861	0.597182	-2.297862	-1.054896	-1.612883	
3	-0.674861	-0.367706	-0.176778	-0.134951	-1.612883	
4	1.527612	1.562070	-0.176778	0.784994	1.502245	
5	1.527612	1.562070	-1.237320	-1.054896	-0.834101	
6	-0.674861	-0.367706	-0.176778	-0.594923	-0.055319	
7	1.527612	0.597182	-0.176778	1.244967	-0.055319	
8	1.527612	1.562070	-0.176778	0.325022	0.723463	
9	-0.674861	-0.367706	-1.237320	-0.134951	-1.612883	
10	0.426376	1.562070	-2.297862	-1.974841	-1.612883	
11	0.426376	0.597182	0.883765	-0.134951	1.502245	
12	1.527612	-0.367706	-0.176778	1.244967	1.502245	
13	1.527612	1.562070	-0.176778	0.325022	-1.612883	
14	-0.674861	0.597182	-0.176778	-0.134951	-0.055319	
15	1.527612	1.562070	-2.297862	-1.054896	-1.612883	
16	-0.674861	-2.297482	-0.176778	1.244967	1.502245	
17	-0.674861	0.597182	-0.176778	-0.134951	1.502245	
18	-0.674861	0.597182	0.883765	0.325022	1.502245	
19	0.426376	-0.367706	-2.297862	-1.054896	-1.612883	
20	0.426376	1.562070	-2.297862	-1.514868	-1.612883	
21	1.527612	-1.332594	-0.176778	1.704939	1.502245	
22	-0.674861	-0.367706	-1.237320	0.325022	-0.055319	
23	-0.674861	0.597182	-2.297862	-1.054896	-1.612883	
24	1.527612	1.562070	-2.297862	-1.514868	-1.612883	
25	0.426376	-0.367706	-0.176778	0.784994	0.723463	
26	1.527612	-0.367706	-0.176778	0.784994	1.502245	
27	-0.674861	-1.332594	-0.176778	1.244967	1.502245	
28	1.527612	1.562070	-1.237320	-1.054896	-0.055319	
29	0.426376	1.562070	-1.237320	-1.974841	-0.834101	
...	...	...	...	...	...	
139404	1.527612	0.597182	0.883765	0.325022	1.502245	
139405	-0.674861	-0.367706	-1.237320	-1.514868	-0.834101	
139406	0.426376	-0.367706	-1.237320	-0.594923	-0.055319	
139407	0.426376	0.597182	-1.237320	-0.134951	-1.612883	
139408	1.527612	1.562070	-1.237320	-0.594923	-0.055319	
139409	0.426376	0.597182	0.883765	1.244967	1.502245	
139410	1.527612	-0.367706	-0.176778	0.784994	-0.055319	
139411	1.527612	1.562070	-1.237320	-0.594923	-0.055319	
139412	-0.674861	-1.332594	-1.237320	-0.134951	-1.612883	
139413	1.527612	1.562070	0.883765	0.784994	1.502245	
139414	0.426376	-0.367706	0.883765	0.784994	1.502245	

139415	-0.674861	0.597182	-0.176778	-1.974841	-0.055319
139416	0.426376	0.597182	-0.176778	-0.134951	-0.834101
139417	-0.674861	0.597182	-2.297862	-1.514868	-1.612883
139418	-0.674861	-0.367706	-0.176778	-0.134951	-0.055319
139419	-0.674861	-0.367706	0.883765	-0.134951	0.723463
139420	-0.674861	-0.367706	-0.176778	0.784994	1.502245
139421	0.426376	-2.297482	-0.176778	0.784994	-0.055319
139422	-1.776097	-0.367706	0.883765	-0.134951	0.723463
139423	1.527612	1.562070	-0.176778	0.784994	0.723463
139424	0.426376	1.562070	-1.237320	-1.974841	-0.834101
139425	-0.674861	-2.297482	0.883765	1.704939	-0.055319
139426	1.527612	1.562070	-1.237320	-0.594923	-1.612883
139427	0.426376	-0.367706	-0.176778	-0.134951	1.502245
139428	-0.674861	-0.367706	-0.176778	-0.134951	-0.055319
139429	-2.877333	-2.297482	-0.176778	1.704939	1.502245
139430	1.527612	0.597182	-0.176778	1.244967	1.502245
139431	1.527612	1.562070	-0.176778	0.784994	1.502245
139432	-1.776097	-0.367706	-0.176778	-0.594923	0.723463
139433	0.426376	1.562070	-2.297862	-1.054896	-1.612883

	CAMEO_INTL_2015_wealth	CAMEO_INTL_2015_life_stage
0	-1.638664	0.090719
1	-0.189864	0.805226
2	-0.914264	0.805226
3	0.534537	-1.338297
4	-0.189864	0.805226
5	-0.914264	0.090719
6	-1.638664	1.519734
7	1.258937	1.519734
8	-1.638664	1.519734
9	-1.638664	0.805226
10	-0.914264	-0.623789
11	0.534537	0.090719
12	1.258937	-1.338297
13	-1.638664	1.519734
14	-1.638664	1.519734
15	-1.638664	0.805226
16	0.534537	-1.338297
17	-0.914264	0.805226
18	-0.189864	0.090719
19	0.534537	0.090719
20	-0.914264	0.805226
21	1.258937	-1.338297
22	-0.189864	0.090719
23	-0.914264	-0.623789
24	-0.914264	1.519734
25	0.534537	0.805226
26	1.258937	0.805226

```
27                   1.258937                     -1.338297
28                  -0.189864                     -0.623789
29                  -0.914264                      0.805226
...                       ...                           ...
139404               0.534537                     -1.338297
139405               0.534537                      0.090719
139406              -1.638664                      0.805226
139407               1.258937                      0.805226
139408              -0.914264                      1.519734
139409              -0.914264                      0.805226
139410               1.258937                      1.519734
139411              -0.914264                     -0.623789
139412               1.258937                      1.519734
139413               1.258937                     -1.338297
139414               0.534537                      0.805226
139415              -1.638664                      0.805226
139416              -1.638664                      0.805226
139417              -0.189864                      0.090719
139418               0.534537                      1.519734
139419              -0.189864                      0.805226
139420              -0.914264                      1.519734
139421               1.258937                     -0.623789
139422              -0.914264                     -0.623789
139423              -1.638664                      1.519734
139424              -1.638664                      0.805226
139425               1.258937                      1.519734
139426              -1.638664                      1.519734
139427               0.534537                     -1.338297
139428               0.534537                      0.090719
139429              -1.638664                      0.805226
139430              -1.638664                      0.805226
139431              -0.914264                      0.805226
139432              -0.914264                      0.805226
139433              -0.189864                      0.090719

[139434 rows x 73 columns]
```

```
In [66]: cust_pca
```

```
Out[66]:                 0          1           2           3          4           5  \
         0     -618.626340  44.274942 -185.368133 -229.248290 -56.094502 -672.458091
         1     -621.059453  44.890282 -187.412650 -233.712232 -51.776794 -673.063599
         2     -620.034686  44.864235 -186.177667 -228.581244 -55.325193 -671.986848
         3     -620.812255  45.514345 -190.494582 -227.095178 -51.345643 -675.987831
         4     -618.748007  44.774911 -188.280141 -229.858496 -54.641085 -673.553235
         5     -618.317749  44.261463 -185.700252 -229.121051 -55.251533 -671.930984
         6     -620.895520  44.358159 -185.708131 -229.575194 -56.212117 -672.905491
         7     -621.103526  50.743515 -190.093967 -235.368431 -51.231721 -674.938750
```

8	-618.930225	44.233540	-185.662980	-227.147028	-53.186641	-673.793274
9	-620.445448	44.266660	-186.271313	-227.918585	-56.075123	-671.147773
10	-618.617673	44.120142	-185.163098	-226.854789	-54.225563	-672.225250
11	-620.349555	44.503584	-187.009003	-227.979803	-52.654559	-673.179590
12	-620.611586	46.340431	-191.531803	-235.123589	-52.070207	-676.018345
13	-617.065624	44.229981	-185.069357	-226.589493	-53.753623	-674.816945
14	-620.655154	44.195214	-185.029225	-226.783690	-54.624054	-674.313225
15	-617.378477	44.206415	-185.478243	-226.112271	-52.169724	-674.504810
16	-621.959073	47.049622	-195.159045	-227.683989	-54.021360	-675.364873
17	-620.991517	45.625082	-188.604908	-233.239097	-50.204921	-672.329104
18	-621.124560	44.534519	-186.483565	-228.806754	-52.970486	-674.403326
19	-620.541990	47.078628	-189.877781	-230.630331	-55.567953	-673.720322
20	-619.304436	44.332155	-187.034565	-228.376006	-51.371029	-669.515055
21	-621.015555	48.806840	-192.262440	-231.342457	-53.571506	-675.555923
22	-620.608003	44.559277	-185.466169	-230.532162	-50.804730	-674.910972
23	-620.154385	43.971606	-184.724838	-227.418963	-50.162085	-671.175742
24	-619.013056	44.301052	-184.511665	-230.840374	-50.784545	-672.857314
25	-620.835065	44.756728	-188.151520	-227.021855	-52.882114	-676.345734
26	-620.608947	46.117895	-189.575187	-233.405192	-50.693318	-676.840601
27	-621.397466	46.493582	-191.880148	-232.956072	-51.386399	-674.968304
28	-619.212047	44.520403	-186.486989	-231.465607	-51.696386	-671.661490
29	-619.397387	44.200482	-185.495396	-227.031692	-52.814495	-673.583149
...	...	...	...	...	...	...
139404	-620.061935	47.545843	-189.966792	-234.549846	-52.320020	-674.803527
139405	-621.073440	44.684744	-187.324152	-229.650768	-55.552322	-672.862636
139406	-620.264203	44.197928	-184.852759	-227.906407	-53.633982	-674.415405
139407	-620.124082	44.546812	-187.145010	-228.972730	-55.936288	-672.621083
139408	-619.265204	44.443920	-185.585012	-232.584610	-51.789255	-672.376655
139409	-619.992296	45.694807	-192.264536	-227.738123	-55.097030	-674.070547
139410	-620.133203	46.299463	-191.430249	-233.752998	-51.374908	-674.619093
139411	-619.190527	44.408945	-185.845265	-229.966677	-56.233275	-672.463329
139412	-620.968874	51.800150	-187.720485	-232.894360	-50.644755	-676.761934
139413	-619.405410	44.936514	-189.035364	-229.918180	-55.819898	-673.951876
139414	-620.917836	47.240593	-190.452418	-234.002317	-51.927342	-676.216497
139415	-620.878138	44.338249	-184.955133	-228.045393	-52.731546	-674.126670
139416	-620.382872	44.073418	-183.521746	-229.071492	-52.765580	-674.558016
139417	-619.661858	44.292568	-184.991042	-227.945240	-52.716266	-673.620668
139418	-621.174489	44.560399	-185.456486	-232.014290	-51.369480	-674.647791
139419	-621.158235	44.577668	-186.591750	-229.984608	-55.911134	-673.560646
139420	-621.325791	44.521210	-186.499252	-229.540596	-50.094076	-674.204799
139421	-620.710488	46.728174	-189.489377	-230.147034	-53.676194	-677.794167
139422	-621.542333	44.666033	-188.756439	-227.180280	-53.659646	-670.637149
139423	-619.394440	44.701517	-187.014664	-227.374265	-53.530241	-675.780108
139424	-619.762139	44.208908	-185.741680	-228.331082	-56.647599	-671.497436
139425	-621.513027	47.914615	-190.193390	-234.964310	-52.172409	-675.531475
139426	-617.999432	44.006204	-183.674650	-227.487519	-53.463521	-674.023073
139427	-621.608584	45.623396	-188.948136	-232.006645	-50.277437	-676.225022
139428	-620.893001	44.844951	-187.834278	-227.179238	-53.397494	-676.662351

```
139429 -622.226756  45.796679 -193.295033 -230.854282 -50.883516 -673.527272
139430 -620.386196  44.898012 -188.849510 -230.057544 -54.765025 -673.984637
139431 -620.386013  44.319453 -184.506793 -229.914154 -53.593562 -675.267942
139432 -621.622080  45.189128 -188.347942 -230.925768 -49.339489 -675.149548
139433 -618.806388  44.264750 -186.498256 -225.356581 -53.805926 -673.037700

                6           7           8           9    ...           20  \
0      -195.277587 -1655.374857 -625.726035 -146.452601    ...    -37.444693
1      -195.892498 -1654.460177 -626.343590 -144.483623    ...    -37.723762
2      -193.260407 -1656.680572 -624.400597 -146.317191    ...    -37.033639
3      -196.411551 -1654.506537 -623.262154 -146.724018    ...    -37.604654
4      -193.734962 -1655.109677 -626.061020 -147.058865    ...    -36.626951
5      -193.374813 -1656.086413 -625.830131 -146.811517    ...    -36.544172
6      -194.305796 -1655.188944 -626.443503 -146.491554    ...    -36.795970
7      -195.632356 -1652.532280 -624.355927 -146.573776    ...    -36.593107
8      -196.153128 -1653.068730 -626.466605 -147.081040    ...    -36.523152
9      -195.545601 -1655.675045 -626.469683 -146.327381    ...    -37.525047
10     -192.990198 -1656.119134 -623.628462 -146.560683    ...    -36.989909
11     -195.165352 -1653.729382 -624.325572 -147.080611    ...    -36.850202
12     -194.005744 -1652.911223 -625.068060 -147.711960    ...    -37.089889
13     -195.104642 -1654.630285 -627.224310 -146.560449    ...    -36.512442
14     -193.619227 -1655.505908 -626.748270 -143.760803    ...    -37.381601
15     -194.355494 -1654.733125 -626.696721 -146.911481    ...    -38.268637
16     -195.530271 -1653.502046 -625.685879 -145.163419    ...    -36.277897
17     -193.620924 -1655.571614 -624.600693 -147.719908    ...    -36.420603
18     -193.902884 -1653.389967 -626.839580 -147.274019    ...    -35.762666
19     -195.259456 -1655.641550 -622.927525 -144.436697    ...    -37.110835
20     -193.917171 -1656.614193 -626.052066 -145.679770    ...    -37.239116
21     -197.339773 -1652.836565 -625.607458 -146.602443    ...    -37.190847
22     -192.957124 -1655.047783 -624.896734 -147.123915    ...    -38.357811
23     -193.248169 -1655.681159 -625.150300 -143.656824    ...    -37.944380
24     -192.430265 -1656.659133 -623.691803 -146.629925    ...    -36.560292
25     -193.872458 -1653.889696 -626.702523 -146.070957    ...    -36.684754
26     -194.117467 -1653.699523 -623.958684 -147.471117    ...    -36.049296
27     -194.369092 -1653.453462 -626.365636 -147.757688    ...    -35.952725
28     -193.281309 -1655.609978 -626.213145 -146.877329    ...    -37.677792
29     -194.454254 -1656.247041 -623.840778 -145.985225    ...    -37.471554
...            ...          ...         ...         ...    ...          ...
139404 -193.710854 -1654.463442 -624.325704 -144.886817    ...    -36.941933
139405 -195.146931 -1655.794843 -623.914501 -147.204149    ...    -36.426653
139406 -193.719952 -1655.434944 -625.325732 -146.814156    ...    -37.011993
139407 -195.638822 -1655.086740 -626.877745 -146.749351    ...    -37.854025
139408 -194.083276 -1655.909008 -626.681578 -147.001446    ...    -37.751734
139409 -196.825844 -1653.401493 -625.405707 -147.591162    ...    -36.869952
139410 -194.553469 -1654.383036 -623.922029 -147.812509    ...    -37.016490
139411 -195.190287 -1655.295055 -626.666908 -146.573267    ...    -37.903713
139412 -195.049705 -1655.398387 -623.442207 -146.364007    ...    -37.416952
139413 -194.749823 -1654.632838 -626.414678 -146.585898    ...    -36.365553
```

139414	-193.868426	-1653.379668	-624.947971	-146.558752	...	-36.150621
139415	-195.238333	-1656.624883	-623.241489	-146.743340	...	-36.768043
139416	-194.303205	-1655.126252	-625.889967	-146.981198	...	-37.454176
139417	-195.131543	-1656.034471	-622.651764	-146.921243	...	-37.545363
139418	-194.543096	-1655.302287	-624.693661	-144.628415	...	-37.401972
139419	-194.541725	-1655.166581	-625.865159	-146.920454	...	-35.936561
139420	-194.956788	-1654.035803	-628.167219	-146.755862	...	-37.238236
139421	-194.852121	-1653.849824	-623.138928	-147.549157	...	-36.614273
139422	-194.763191	-1652.492118	-625.635117	-145.994486	...	-37.419991
139423	-195.377704	-1654.623999	-626.752965	-144.232630	...	-36.792117
139424	-194.573252	-1656.133563	-625.819499	-146.571541	...	-37.227973
139425	-195.328835	-1652.662450	-624.140663	-146.739750	...	-35.307479
139426	-192.701831	-1655.559760	-625.922813	-146.566097	...	-38.447338
139427	-195.207154	-1654.760721	-623.164388	-144.391152	...	-36.138543
139428	-193.427644	-1654.268817	-625.444751	-146.210438	...	-38.040321
139429	-195.000505	-1653.882986	-626.632470	-147.875568	...	-36.065290
139430	-196.753345	-1653.457375	-628.340280	-147.015322	...	-37.289058
139431	-193.578904	-1655.107387	-625.305333	-147.088972	...	-36.058602
139432	-194.108456	-1655.200781	-624.225194	-146.586598	...	-37.824508
139433	-196.302671	-1656.405373	-624.739758	-146.698983	...	-37.515107

	21	22	23	24	25	26 \
0	-124.968316	-47.081781	-185.911753	-41.607452	15.081172	-71.169816
1	-124.387283	-47.970706	-186.901404	-39.300335	16.031936	-71.733048
2	-123.785200	-46.526957	-187.210697	-40.193582	17.358723	-72.721053
3	-123.622017	-47.254082	-186.510915	-40.522505	18.120418	-70.581521
4	-124.517618	-45.246685	-185.045106	-40.397039	17.081728	-72.708418
5	-122.624629	-47.670885	-185.723553	-41.502972	16.443306	-72.475451
6	-125.108126	-46.813603	-185.994658	-40.630995	16.318999	-71.604805
7	-125.449637	-48.636057	-186.070361	-40.516517	17.044643	-73.343870
8	-124.669456	-46.273195	-185.455846	-41.024793	17.242054	-72.032977
9	-122.929406	-47.132092	-186.522634	-40.817909	17.614746	-72.679300
10	-123.019242	-46.628326	-186.907726	-41.835626	16.833582	-71.650273
11	-124.901701	-46.642426	-183.903108	-40.775444	17.289858	-73.290811
12	-125.179797	-46.954401	-185.145649	-42.635013	16.514438	-72.492422
13	-124.288317	-47.170257	-186.279037	-41.705428	17.177215	-72.525590
14	-124.390099	-47.143660	-185.142269	-42.020838	17.961164	-72.193291
15	-123.649240	-48.216707	-187.085509	-40.940858	17.050526	-72.174029
16	-124.314063	-47.302435	-185.646144	-41.368939	17.818730	-74.235147
17	-123.178226	-47.401650	-185.982519	-40.910700	14.624849	-71.665193
18	-124.930966	-46.932124	-187.236684	-42.488598	16.021265	-71.375239
19	-123.379418	-46.200477	-185.803238	-39.813975	16.978496	-71.826657
20	-123.495415	-45.901627	-185.743384	-41.035630	15.513628	-72.919511
21	-124.079791	-47.079426	-183.802533	-40.903769	16.047394	-74.166112
22	-124.485894	-46.206273	-186.244245	-40.738712	16.425937	-72.391853
23	-123.368574	-46.244691	-187.680444	-43.331624	17.404825	-71.799488
24	-125.033163	-46.642791	-185.188304	-39.869225	16.165821	-72.468106
25	-122.964808	-48.181857	-186.058721	-40.516823	18.198381	-71.162349

26	-124.540386	-48.063272	-186.525269	-40.611352	15.944228	-72.960302
27	-124.175866	-45.736838	-186.291114	-42.058558	16.353149	-70.981204
28	-124.268903	-45.965210	-184.509863	-41.796363	15.271617	-71.765174
29	-123.433733	-47.959645	-186.057267	-40.711765	17.643428	-71.204146
...	...	...	...	...	...	...
139404	-124.662674	-47.484508	-186.328014	-43.091892	15.872151	-72.580578
139405	-124.530848	-47.034059	-186.023188	-40.198420	15.758190	-71.415720
139406	-123.845643	-45.903276	-186.434285	-40.136787	18.004445	-72.197307
139407	-123.582675	-46.800317	-185.831294	-41.189029	17.472006	-72.114495
139408	-125.300118	-46.228088	-185.903541	-40.546128	16.555170	-71.307343
139409	-125.128007	-46.986514	-182.937395	-40.937042	17.161105	-70.903472
139410	-125.251766	-48.578831	-186.578812	-40.439281	17.778153	-72.499587
139411	-123.080506	-46.974256	-185.750955	-42.370813	15.646674	-72.505604
139412	-126.579607	-47.680795	-185.525289	-41.476360	17.038868	-71.400850
139413	-121.881865	-45.396676	-185.346070	-41.817240	16.484826	-71.932253
139414	-125.435138	-46.434475	-186.479341	-41.018763	16.631016	-72.780956
139415	-122.786961	-48.302252	-184.185332	-42.227152	16.833810	-71.672911
139416	-123.314235	-47.378162	-185.890767	-41.140035	17.673611	-72.607181
139417	-123.837348	-47.888791	-184.318538	-40.842933	16.655123	-72.677710
139418	-124.615428	-46.973722	-185.661293	-40.435545	15.736596	-71.403557
139419	-123.797932	-45.663376	-185.104526	-41.390502	17.105976	-72.123439
139420	-124.122407	-46.853891	-187.412379	-40.089789	16.845969	-71.093131
139421	-123.961982	-47.944081	-186.004093	-41.799704	15.745529	-73.460205
139422	-124.139183	-48.682753	-186.325820	-42.693360	15.678225	-72.125827
139423	-123.717578	-48.022382	-185.936315	-42.255183	15.576660	-71.208962
139424	-123.857479	-47.689042	-186.923032	-41.179700	16.342066	-71.974779
139425	-125.873360	-48.661724	-186.780796	-41.711681	16.563727	-73.655085
139426	-124.726538	-47.358217	-186.745727	-41.681665	16.825868	-72.424178
139427	-124.305642	-45.530431	-184.806213	-41.116710	16.795247	-71.747805
139428	-124.600186	-46.219082	-183.819622	-39.818757	16.741434	-72.897690
139429	-124.942883	-46.874653	-184.541343	-39.874880	16.773390	-70.814258
139430	-124.955564	-46.285264	-185.074219	-41.251644	15.685008	-71.788351
139431	-123.806900	-46.202747	-184.672880	-40.861294	17.383078	-72.655415
139432	-124.885022	-46.439897	-185.082543	-40.353894	17.357771	-71.752891
139433	-124.686612	-45.715196	-185.457645	-42.406869	16.573512	-71.656200

	27	28	29
0	15.514160	47.153178	61.156201
1	15.621666	46.467161	61.396764
2	15.157594	46.539265	60.093078
3	16.917885	47.716557	62.231513
4	16.225010	48.202380	62.109326
5	15.762920	46.567709	62.502313
6	15.384060	46.513248	61.966705
7	14.766944	46.745940	61.229019
8	17.703096	47.618863	61.819820
9	14.386529	46.439094	61.195420
10	14.626869	46.965691	60.129872

11	16.109470	46.790900	62.318942
12	15.155451	46.842626	61.702152
13	16.160310	46.062443	60.591997
14	15.018639	45.101997	61.139150
15	15.548394	48.155100	62.266637
16	15.242880	47.000305	62.645651
17	15.061879	46.301116	61.560614
18	14.622533	47.735153	61.855293
19	14.632348	46.340945	61.353837
20	16.200208	47.265843	60.312097
21	13.124826	47.524966	62.091601
22	15.855838	45.704255	60.042005
23	16.153393	46.637676	60.454859
24	16.076879	47.354070	60.090750
25	16.421710	47.880151	62.205573
26	14.123971	45.618723	62.139863
27	15.851085	46.753459	60.892736
28	15.738420	47.244137	62.574830
29	16.247189	47.455190	61.517133
...	...	...	...
139404	14.454200	45.972332	61.336480
139405	16.562703	47.389858	60.498565
139406	17.026850	48.135779	61.780565
139407	15.508691	47.808205	61.407922
139408	15.130182	46.518202	61.420364
139409	16.551566	47.999817	62.252611
139410	14.897107	47.010775	60.982652
139411	14.492447	46.433502	61.239006
139412	16.016894	46.702648	60.190889
139413	14.602959	45.759515	61.249485
139414	14.495257	45.127423	60.624181
139415	16.254478	46.742034	62.295737
139416	14.822296	47.434229	61.531469
139417	18.288949	47.339905	60.411257
139418	14.613866	45.963542	61.449113
139419	15.650786	45.954559	62.197720
139420	15.676814	47.578856	61.398703
139421	14.280376	47.385773	61.502690
139422	15.638623	47.128892	62.461044
139423	15.736127	47.333814	62.003805
139424	15.925309	45.610932	62.353867
139425	13.557755	45.930030	61.032690
139426	14.981100	47.378951	60.474154
139427	16.376872	46.069696	61.264789
139428	15.978675	46.525296	62.460804
139429	16.858158	47.582975	61.769698
139430	15.595998	47.552400	62.686687
139431	14.711578	46.083012	60.325292

```
139432  14.538147  45.672355  61.435588
139433  16.624485  46.543587  59.744549

[139434 rows x 30 columns]
```

In [71]:
```
cust_pred
```

Out[71]:
```
array([10, 10, 10, ..., 10, 10, 10], dtype=int32)
```

1.3.4 Step 3.3: Compare Customer Data to Demographics Data

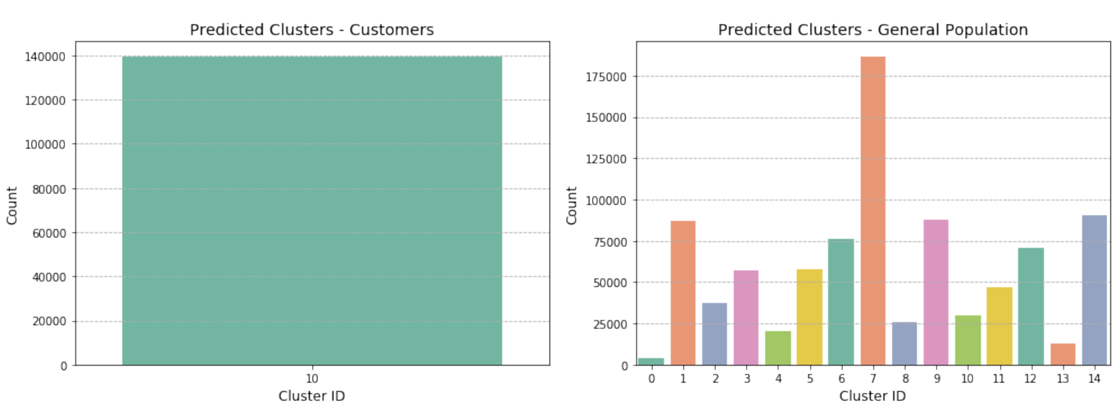
At this point, you have clustered data based on demographics of the general population of Germany, and seen how the customer data for a mail-order sales company maps onto those demographic clusters. In this final substep, you will compare the two cluster distributions to see where the strongest customer base for the company is.

Consider the proportion of persons in each cluster for the general population, and the proportions for the customers. If we think the company's customer base to be universal, then the cluster assignment proportions should be fairly similar between the two. If there are only particular segments of the population that are interested in the company's products, then we should see a mismatch from one to the other. If there is a higher proportion of persons in a cluster for the customer data compared to the general population (e.g. 5% of persons are assigned to a cluster for the general population, but 15% of the customer data is closest to that cluster's centroid) then that suggests the people in that cluster to be a target audience for the company. On the other hand, the proportion of the data in a cluster being larger in the general population than the customer data (e.g. only 2% of customers closest to a population centroid that captures 6% of the data) suggests that group of persons to be outside of the target demographics.

Take a look at the following points in this step:

- Compute the proportion of data points in each cluster for the general population and the customer data. Visualizations will be useful here: both for the individual dataset proportions, but also to visualize the ratios in cluster representation between groups. Seaborn's `countplot()` or `barplot()` function could be handy.
- Recall the analysis you performed in step 1.1.3 of the project, where you separated out certain data points from the dataset if they had more than a specified threshold of missing values. If you found that this group was qualitatively different from the main bulk of the data, you should treat this as an additional data cluster in this analysis. Make sure that you account for the number of data points in this subset, for both the general population and customer datasets, when making your computations!
- Which cluster or clusters are overrepresented in the customer dataset compared to the general population? Select at least one such cluster and infer what kind of people might be represented by that cluster. Use the principal component interpretations from step 2.3 or look at additional components to help you make this inference. Alternatively, you can use the `.inverse_transform()` method of the PCA and StandardScaler objects to transform centroids back to the original data space and interpret the retrieved values directly.
- Perform a similar investigation for the underrepresented clusters. Which cluster or clusters are underrepresented in the customer dataset compared to the general population, and what kinds of people are typified by these clusters?

In [76]:
```python
# Compare the proportion of data in each cluster for the customer data to the
# proportion of data in each cluster for the general population.
# Scale the customers data

# Create a custom color palette
custom_palette = sns.color_palette("Set2")

# Create subplots and adjust spacing
figure, axs = plt.subplots(nrows=1, ncols=2, figsize=(14, 5))
figure.subplots_adjust(hspace=0.5, wspace=0.3)

# Plot countplot for 'cust_pred'
sns.countplot(cust_pred, ax=axs[0], palette=custom_palette)
axs[0].set_title('Predicted Clusters - Customers', fontsize=14)
axs[0].set_xlabel('Cluster ID', fontsize=12)
axs[0].set_ylabel('Count', fontsize=12)
axs[0].grid(axis='y', linestyle='--')

# Customize ticks and labels for better readability
axs[0].tick_params(axis='both', which='major', labelsize=10)
axs[0].tick_params(axis='x')

# Plot countplot for 'general_predictions'
sns.countplot(general_predictions, ax=axs[1], palette=custom_palette)
axs[1].set_title('Predicted Clusters - General Population', fontsize=14)
axs[1].set_xlabel('Cluster ID', fontsize=12)
axs[1].set_ylabel('Count', fontsize=12)
axs[1].grid(axis='y', linestyle='--')

# Customize ticks and labels for better readability
axs[1].tick_params(axis='both', which='major', labelsize=10)
axs[1].tick_params(axis='x')

# Show the plot
plt.tight_layout()
plt.show()
```



```
In [83]: # What kinds of people are part of a cluster that is overrepresented in the
         # customer data compared to the general population?
         # Referenced: https://stackoverflow.com/questions/59771061/using-inverse-transform-minm

         popular = scaler.inverse_transform(pca.inverse_transform(kmeans_model.cluster_centers_[
         overrepresented_cluster = pd.Series(data = popular, index = customers.columns)

         overrepresented_cluster
```

```
Out[83]: ALTERSKATEGORIE_GROB         5.550208
         ANREDE_KZ                    2.275930
         CJT_GESAMTTYP                9.109683
         FINANZ_MINIMALIST            7.026407
         FINANZ_SPARER                6.918240
         FINANZ_VORSORGER             7.718749
         FINANZ_ANLEGER               7.743746
         FINANZ_UNAUFFAELLIGER        6.532687
         FINANZ_HAUSBAUER             7.507296
         FINANZTYP                   12.400200
         GFK_URLAUBERTYP             34.006055
         GREEN_AVANTGARDE             0.244297
         HEALTH_TYP                   3.713727
         LP_FAMILIE_FEIN             17.869335
         LP_FAMILIE_GROB              6.093701
         LP_STATUS_FEIN              20.336713
         LP_STATUS_GROB               5.841434
         NATIONALITAET_KZ             1.691088
         PRAEGENDE_JUGENDJAHRE        8.196298
         RETOURTYP_BK_S              12.225064
         SEMIO_SOZ                   12.224091
         SEMIO_FAM                   12.454938
         SEMIO_REL                   11.308440
         SEMIO_MAT                   12.826185
         SEMIO_VERT                  13.081485
         SEMIO_LUST                  12.766014
         SEMIO_ERL                   12.029692
         SEMIO_KULT                  10.112271
         SEMIO_RAT                   12.964144
         SEMIO_KRIT                  12.412673
                                        ...
         ANZ_HAUSHALTE_AKTIV          0.053344
         ANZ_HH_TITEL                10.235618
         GEBAEUDETYP                  8.023550
         KONSUMNAEHE               8280.552261
         MIN_GEBAEUDEJAHR             1.011914
```

```
OST_WEST_KZ                  3.223794
CAMEO_INTL_2015              2.334188
KBA05_ANTG1                  1.246258
KBA05_ANTG2                  0.551280
KBA05_ANTG3                  6.934162
KBA05_ANTG4                 12.478783
KBA05_GBZ                   10.257341
BALLRAUM                    12.621078
EWDICHTE                     6.979739
INNENSTADT                   5.291557
GEBAEUDETYP_RASTER           6.688717
KKK                          6.736960
MOBI_REGIO                  12.216858
ONLINE_AFFINITAET        42164.319741
REGIOTYP                     4.149696
KBA13_ANZAHL_PKW             5.060716
PLZ8_ANTG1                   3.045872
PLZ8_ANTG2                   1.223079
PLZ8_ANTG3                   5.870205
PLZ8_ANTG4                   5.720693
PLZ8_HHZ                     6.429666
PLZ8_GBZ                    16.847763
ARBEIT                       7.231672
ORTSGR_KLS9                  8.254031
RELAT_AB                     6.517874
Length: 73, dtype: float64
```

In [84]:
```python
# What kinds of people are part of a cluster that is underrepresented in the
# customer data compared to the general population?

unpopular = scaler.inverse_transform(pca.inverse_transform(kmeans_model.cluster_centers
cust_under_represented = pd.Series(data = unpopular, index = customers.columns)

cust_under_represented
```

Out[84]:
```
ALTERSKATEGORIE_GROB          5.607416
ANREDE_KZ                     2.285383
CJT_GESAMTTYP                 9.026964
FINANZ_MINIMALIST             7.487114
FINANZ_SPARER                 6.691441
FINANZ_VORSORGER              8.209147
FINANZ_ANLEGER                7.765754
FINANZ_UNAUFFAELLIGER         7.043692
FINANZ_HAUSBAUER              6.663519
FINANZTYP                    11.293628
GFK_URLAUBERTYP              32.682877
GREEN_AVANTGARDE              0.306414
HEALTH_TYP                    3.717129
```

```
LP_FAMILIE_FEIN              21.039059
LP_FAMILIE_GROB               6.711023
LP_STATUS_FEIN               25.983858
LP_STATUS_GROB                6.881639
NATIONALITAET_KZ              1.679703
PRAEGENDE_JUGENDJAHRE         7.895775
RETOURTYP_BK_S               11.742168
SEMIO_SOZ                    11.893880
SEMIO_FAM                    12.034191
SEMIO_REL                    11.097447
SEMIO_MAT                    13.065234
SEMIO_VERT                   13.215982
SEMIO_LUST                   12.892521
SEMIO_ERL                    11.537428
SEMIO_KULT                   10.001623
SEMIO_RAT                    12.869439
SEMIO_KRIT                   12.489390
                               ...
ANZ_HAUSHALTE_AKTIV           0.041223
ANZ_HH_TITEL                  8.941425
GEBAEUDETYP                   7.905517
KONSUMNAEHE                8281.571175
MIN_GEBAEUDEJAHR              1.172554
OST_WEST_KZ                   4.077262
CAMEO_INTL_2015               2.605488
KBA05_ANTG1                   1.065778
KBA05_ANTG2                   0.418911
KBA05_ANTG3                   7.378287
KBA05_ANTG4                  13.790442
KBA05_GBZ                     8.110462
BALLRAUM                     15.299383
EWDICHTE                      7.241520
INNENSTADT                    4.979639
GEBAEUDETYP_RASTER            7.521053
KKK                           7.364750
MOBI_REGIO                   11.321146
ONLINE_AFFINITAET        651518.950575
REGIOTYP                      4.861229
KBA13_ANZAHL_PKW              5.022249
PLZ8_ANTG1                    2.739858
PLZ8_ANTG2                    0.955753
PLZ8_ANTG3                    9.550412
PLZ8_ANTG4                   10.110956
PLZ8_HHZ                      5.167088
PLZ8_GBZ                     12.670352
ARBEIT                        5.122434
ORTSGR_KLS9                   6.718658
RELAT_AB                      7.207905
```

```
Length: 73, dtype: float64
```

```
In [89]: popular = pd.Series(popular)
         unpopular = pd.Series(unpopular)

         # Calculate the difference between overrepresented and underrepresented clusters for ea
         difference = popular - unpopular

         # Create two lists to hold features that are relatively popular or unpopular
         popular_features = []
         unpopular_features = []

         # Define a threshold for the difference to consider it significant
         threshold = 5

         # Identify the features that are relatively popular or unpopular
         for feature, diff in difference.items():
             if diff > threshold:
                 popular_features.append(feature)
             elif diff < -threshold:
                 unpopular_features.append(feature)

         # Print the descriptions of the segments
         print("Segments of the population that are relatively popular with the mail order compa
         for feature in popular_features:
             print(f"- {customers.columns[feature]}")

         print("\nSegments of the population that are relatively unpopular with the mail order c
         for feature in unpopular_features:
             print(f"- {customers.columns[feature]}")
```

```
Segments of the population that are relatively popular with the mail order company:
- WOHNDAUER_2008

Segments of the population that are relatively unpopular with the mail order company:
- LP_STATUS_FEIN
- ONLINE_AFFINITAET
```

1.3.5 Discussion 3.3: Compare Customer Data to Demographics Data

It looks like the over-represented cluster results are related to WOHNDAUER_2008. This is related to the length of residence of an individual. It may be true that the longer someone is in their residence, it could indicate they would be a potentially good candidate for the mail order offers.

On the other hand, the under-represented clusters seem to refer to LP_STATUS_FEIN and ONLINE_AFFINITAET. LP_STATUS_FEIN is related to the social status of an individual. Perhaps there are 'lower' status individuals and thus the cluster did not perform as highly as some others. ONLINE_AFFINITAET relates to the online presence of individuals. It's possible that a segment of the population with low online affinity may not respond well to mail order catalogs since they prefer traditional shopping methods.

> Congratulations on making it this far in the project! Before you finish, make sure to check through the entire notebook from top to bottom to make sure that your analysis follows a logical flow and all of your findings are documented in **Discussion** cells. Once you've checked over all of your work, you should export the notebook as an HTML document to submit for evaluation. You can do this from the menu, navigating to **File -> Download as -> HTML (.html)**. You will submit both that document and this notebook for your project submission.

```
In [ ]:
```