In [1]:
```
import pandas as pd
import pyarrow.parquet as pq
import pyarrow as pa
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import json
from typing import Union, List
import matplotlib.patches as mpatches
import scipy.stats as stats
import numpy as np
from scipy.stats import shapiro
import math
import matplotlib.ticker as ticker
import plotly.express as px
```

1. Laden des Rohdatensatzes

1.1 Laden der Frage-Codes

Zur besseren Verarbeitung der Datensätze werden die zugehörigen Codes der Fragen geladen und verwendet.

In [2]:
```
with open('code.json', 'r') as f:
  code_data = json.load(f)
```

In [3]:
```
df_code = pd.DataFrame(code_data)
codes = df_code["Variable"].tolist()
```

In [4]:
```
df_code.head()
```

Out[4]:

	Variable	Fragetyp	Fragetext	Werte	Zeichenlimit	Wert
0	60371597	Skalafrage	2.1 ein hohes Einkommen	{'<leer>': 'Ungültig / Keine Antwort', '1': 'u...	None	None
1	60371598	Skalafrage	2.2 gute Aufstiegsmöglichkeiten	{'<leer>': 'Ungültig / Keine Antwort', '1': 'u...	None	None
2	60371599	Skalafrage	2.3 einen sicheren Arbeitsplatz	{'<leer>': 'Ungültig / Keine Antwort', '1': 'u...	None	None
3	60371600	Skalafrage	2.4 viele Kontakte zu anderen Menschen	{'<leer>': 'Ungültig / Keine Antwort', '1': 'u...	None	None
4	60371601	Skalafrage	2.5 eine gute/kollegiale Arbeitsatmosphäre	{'<leer>': 'Ungültig / Keine Antwort', '1': 'u...	None	None

1.2 Laden der Stichproben Ergebnisse

Die Stichproben Ergebnisse werden in ein pandas DataFrame geladen und die Spaltenüberschriften (Fragetexte) mit ihren jeweiligen Codes ersetzt.

In [5]:
```
df_results_raw = pd.read_csv('./data/pia_master.csv', sep=";")
```

In [6]:
```
print(len(codes))
print(len(df_results_raw.columns))
```

```
53
56
```

In [7]:
```
df_results_raw.head()
```

Out[7]:

	Bogen	ein hohes Einkommen	gute Aufstiegsmöglichkeiten	einen sicheren Arbeitsplatz	viele Kontakte zu anderen Menschen	gute/koll Arbeitsatmos
0	1	4	4	5	4	
1	2	4	5	5	5	
2	3	4	2	5	4	
3	4	5	4	5	4	
4	5	5	4	5	3	

5 rows × 56 columns

In [8]:
```
df_results = df_results_raw.rename(columns={
    col: codes[i] for i, col in enumerate([col for i, col in enumerate(df_re
```

In [9]:
```
df_results.head()
```

Out[9]:

	Bogen	60371597	60371598	60371599	60371600	60371601	60371602	60371
0	1	4	4	5	4	4	5	
1	2	4	5	5	5	5	4	
2	3	4	2	5	4	5	4	
3	4	5	4	5	4	4	4	
4	5	5	4	5	3	5	5	

5 rows × 56 columns

1.3. Utility Functions

Die Utility Funktionen werden verwendet, um die Fragen Codes wieder in Text und die numerische Darstellung der Antwortmöglichkeiten in den zugehörigen Text

umzuwandeln.

In [10]:
```
def get_answer_text(question_code: str, answer_res: Union[float, int, str] =
    if isinstance(answer_res, str):
        _res = answer_res
    else:
        _res = int(answer_res) if answer_res != 0 else '<leer>'
    t_code = df_code[df_code["Variable"] == question_code]
    question_type = t_code["Fragetyp"].values[0]
    assert (answer_res is not None and question_type != 'Offene Frage') or (
    answer_dict = t_code["Werte"].values[0]
    out = answer_dict[str(_res)] if answer_res is not None else answer_dict
    return out
```

In [11]:
```
def get_question_text(question_code: str):
    return df_code[df_code["Variable"] == question_code]["Fragetext"].values
```

1.4 Gliederung der Fragetypen

Die Umfrage enthält 3 Fragetypen.

- Skalafragen - Antwortmöglichkeiten werden mittels kategorischen numerischen Werten auf einer Ordinalskala dargestellt.
- 1 aus n - Antwortmöglichkeiten werden mittels kategorischen numerischen Werten auf einer Nominalskala dargestellt.
- Offene Frage - Antwort besteht aus einem Freitext

In [12]:
```
skala_ids = df_code[df_code["Fragetyp"] == "Skalafrage"]["Variable"].tolist(
one_of_n_ids = df_code[df_code["Fragetyp"] == "1 aus n"]["Variable"].tolist(
numeric_ids = skala_ids + one_of_n_ids
open_q_ids = df_code[df_code["Fragetyp"] == "Offene Frage"]["Variable"].toli
```

Neben den fachlichen Fragen der Umfrage, gibt es auch Fragetypen, welche zur Klassifierung der Umfrage Ergebnisse dienen.

- Bitte kreuze an, welcher Generation du zugehörig bist
- Welchem Geschlecht fühlst du dich zugehörig?
- Bitte wähle deinen höchsten formalen Bildungsabschluss aus:
- Bist du derzeit berufstätig?
- In welcher Beschäftigungsform befindest du dich?
- In welchem Wirtschaftssektor bist du beschäftigt?

In [19]:
```
generation_col = '60371649'
gender_col = '60371650'
academic_col = '60371651'
working_col = '60371652'
work_form_col = '60371653'
economic_sector_col = '60371654'
```

Zur weiteren Einschränkung der Fragetypen, können wir die Überschneidungen zwischen "fachlichen" Fragen und "gruppierenden" Fragen bestimmen und für weitere mögliche Auswertungen zwischen den beiden Typen unterscheiden.

In [13]:
```
skala_fach_ids = [x for x in skala_ids if int(x) <= 60371648]
one_of_n_fach_ids = [x for x in one_of_n_ids if int(x) <= 60371648]
num_fach_cols = skala_fach_ids + one_of_n_fach_ids

print(skala_fach_ids)
print(one_of_n_fach_ids)
```

```
['60371597', '60371598', '60371599', '60371600', '60371601', '60371602', '603
71603', '60371604', '60371605', '60371606', '60371607', '60371608', '6037160
9', '60371615', '60371616', '60371617', '60371618', '60371619', '60371620',
'60371624', '60371625', '60371626', '60371627', '60371628', '60371629', '6037
1630', '60371631', '60371632', '60371633', '60371634', '60371635', '6037163
6', '60371637', '60371638', '60371639', '60371640', '60371641', '60371642',
'60371643', '60371644', '60371645', '60371646', '60371647', '60371648']
['60371613', '60371614']
```

In [14]:
```
for col in num_fach_cols:
    print(f"{col}: {get_question_text(col)}")
```

```
60371597: 2.1 ein hohes Einkommen
60371598: 2.2 gute Aufstiegsmöglichkeiten
60371599: 2.3 einen sicheren Arbeitsplatz
60371600: 2.4 viele Kontakte zu anderen Menschen
60371601: 2.5 eine gute/kollegiale Arbeitsatmosphäre
60371602: 2.6 das Gefühl, etwas zu leisten
60371603: 2.7 das Gefühl, anerkannt zu werden
60371604: 2.8 Möglichkeiten, sich um andere Menschen zu kümmern
60371605: 2.9 Möglichkeiten, eigene Ideen einzubringen
60371606: 2.10 Möglichkeiten, etwas Nützliches für die Gesellschaft zu tun
60371607: 2.11 die Möglichkeit, etwas zu tun, das ich sinnvoll finde
60371608: 2.12 genügend Freizeit neben der Berufstätigkeit
60371609: 2.13 Flexible Arbeitszeiten (z.B. Home-Office oder Gleitzeit)
60371615: 4.1 Größe des Unternehmens
60371616: 4.2 Ruf des Unternehmens
60371617: 4.3 Standort des Unternehmens
60371618: 4.4 Bereitstellung neuster Technologien am Arbeitsplatz
60371619: 4.5 Flache Hierarchien (auf Augenhöhe miteinander arbeiten)
60371620: 4.6 Nachhaltigkeit  Corporate Social Responsibility *
60371624: 5.1 Gesetz und Ordnung respektieren
60371625: 5.2 einen hohen Lebensstandard haben
60371626: 5.3 Macht und Einfluss haben
60371627: 5.4 seine eigene Phantasie und Kreativität entwickeln
60371628: 5.5 nach Sicherheit streben
60371629: 5.6 sozial Benachteiligten und gesellschaftlichen Randgruppen helfe
n
60371630: 5.7 sich und seine Bedürfnisse gegen andere durchsetzen
60371631: 5.8 fleißig und ehrgeizig sein
60371632: 5.9 auch solche Meinungen tolerieren, denen man eigentlich nicht zu
stimmen kann
60371633: 5.10 sich politisch engagieren
60371634: 5.11 das Leben in vollen Zügen genießen
60371635: 5.12 eigenverantwortlich leben und handeln
60371636: 5.13 das tun, was die anderen auch tun
60371637: 5.14 am Althergebrachten festhalten
60371638: 5.15 ein gutes Familienleben führen
60371639: 5.16 stolz sein auf die deutsche Geschichte
60371640: 5.17 einen Partner haben, dem man vertrauen kann
60371641: 5.18 gute Freunde haben, die einen anerkennen und akzeptieren
60371642: 5.19 viele Kontakte zu anderen Menschen haben
60371643: 5.20 gesundheitsbewusst leben
60371644: 5.21 sich bei seinen Entscheidungen auch von seinen Gefühlen leiten
lassen
60371645: 5.22 von anderen Menschen unabhängig sein
60371646: 5.23 sich unter allen Umständen umweltbewusst verhalten
60371647: 5.24 an Gott glauben
60371648: 5.25 die Vielfalt der Menschen anerkennen und respektieren
60371613: 3.1 Welches der Kriterien ist für dich am wichtigsten?
60371614: 3.2 Welches der Kriterien ist für dich am wenigsten wichtig?
```

Darüberhinaus gliedert sich die Umfrage in 2 Fragen-Cluster:

- Fragen rund um die Erwartungen der Probanten an einen Arbeitsplatz sowie Arbeitgebermerkmalen
- Fragen an persönliche Wertevorstellungen der Probanten

In [15]:
```
erwartungen_ids = [x for x in codes if 60371597 <= int(x) and int(x) <= 6037
print(erwartungen_ids)
```

```
['60371597', '60371598', '60371599', '60371600', '60371601', '60371602', '603
71603', '60371604', '60371605', '60371606', '60371607', '60371608', '6037160
9', '60371615', '60371616', '60371617', '60371618', '60371619', '60371620']
```

In [16]:
```
werte_ids = [x for x in codes if 60371624 <= int(x) and int(x) <= 60371648]
print(werte_ids)
```

```
['60371624', '60371625', '60371626', '60371627', '60371628', '60371629', '603
71630', '60371631', '60371632', '60371633', '60371634', '60371635', '6037163
6', '60371637', '60371638', '60371639', '60371640', '60371641', '60371642',
'60371643', '60371644', '60371645', '60371646', '60371647', '60371648']
```

2. Pre Processing

Bevor wir mit der Analyse der Umfrage Ergebnisse starten können, müssen einige Pre Processing Schritte vollzogen werden, um die Daten fachlich aufzubereiten.

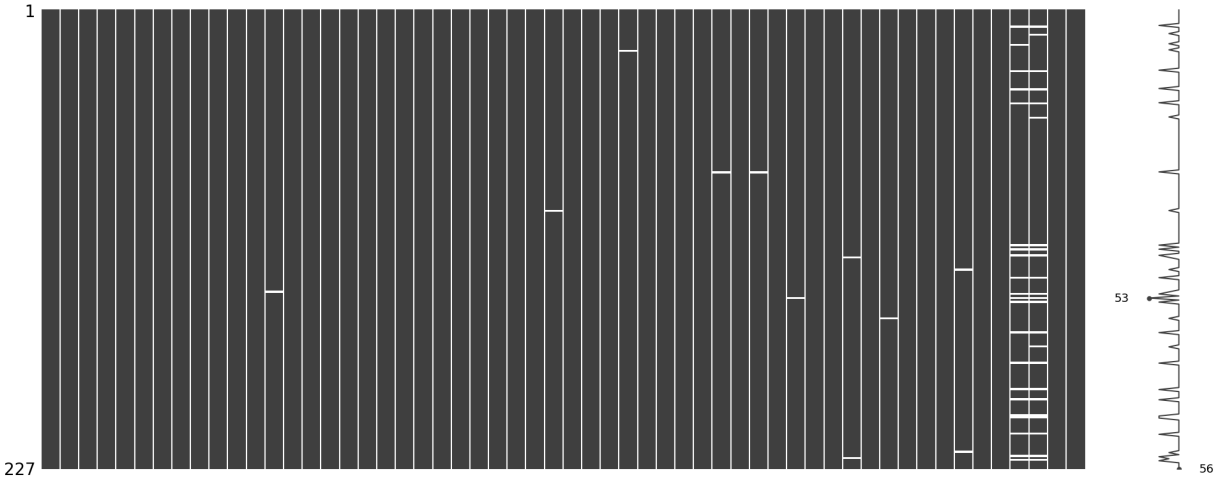
Punkte die dabei beachtet werden müssen sind:

- Behandlung von fehlenden/falschen Werten
- Standardisierung der Antworten (Werte liegen in integer und float Darstellung vor)
- etc.

Im ersten Schritte werden ungültige Werte ausgewertet. Unter ungültigen Werten verstehen wir im Folgenden fehlende Werte.

In [17]:
```
msno.matrix(df_results)
```

Out[17]:
```
<Axes: >
```



In [20]:
```
pd.crosstab(index=df_results[generation_col], columns=[academic_col])
```

Out[20]:

col_0	60371651
60371649	
1	109
2	96
3	22

Im weiteren Verlauf werden nur noch die Generationen Y und Z verglichen. Die restlichen Generationen werden aussortiert. In unserem Fall entfallen so 22 Beobachtungen.

In [21]:
```
df_yz = df_results[df_results['60371649'] != 3]
```

In [22]:
```
df_yz.info()
```

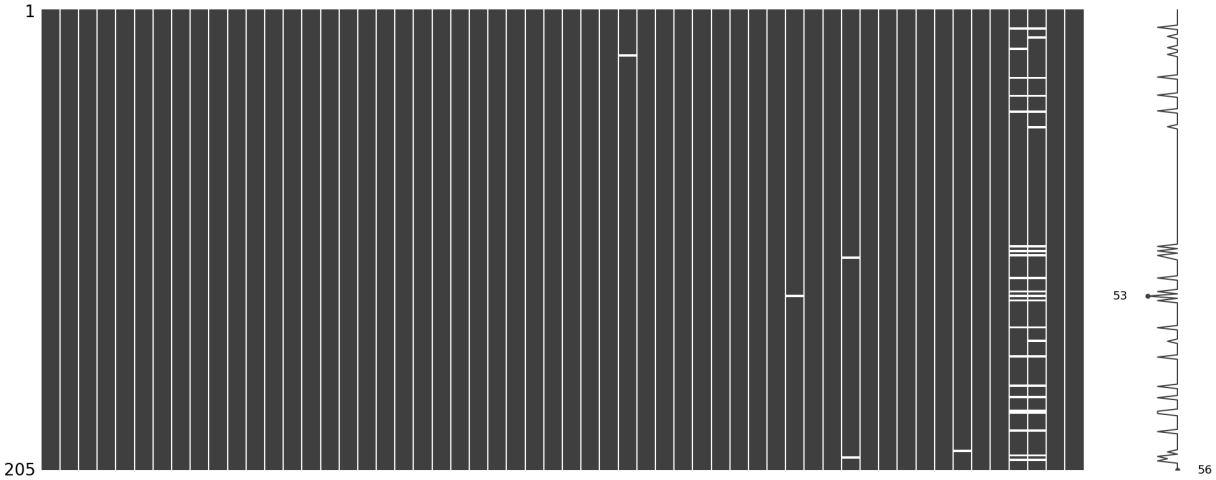
```
<class 'pandas.core.frame.DataFrame'>
Index: 205 entries, 0 to 226
Data columns (total 56 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Bogen               205 non-null    int64
 1   60371597            205 non-null    int64
 2   60371598            205 non-null    int64
 3   60371599            205 non-null    int64
 4   60371600            205 non-null    int64
 5   60371601            205 non-null    int64
 6   60371602            205 non-null    int64
 7   60371603            205 non-null    int64
 8   60371604            205 non-null    int64
 9   60371605            205 non-null    int64
 10  60371606            205 non-null    int64
 11  60371607            205 non-null    int64
 12  60371608            205 non-null    float64
 13  60371609            205 non-null    int64
 14  60371612            205 non-null    object
 15  60371613            205 non-null    int64
 16  60371614            205 non-null    int64
 17  60371615            205 non-null    int64
 18  60371616            205 non-null    int64
 19  60371617            205 non-null    int64
 20  60371618            205 non-null    int64
 21  60371619            205 non-null    int64
 22  60371620            205 non-null    int64
 23  60371624            205 non-null    int64
 24  60371625            205 non-null    int64
 25  60371626            205 non-null    int64
 26  60371627            205 non-null    int64
 27  60371628            205 non-null    float64
 28  60371629            205 non-null    int64
 29  60371630            205 non-null    int64
 30  60371631            205 non-null    int64
 31  60371632            204 non-null    float64
 32  60371633            205 non-null    int64
 33  60371634            205 non-null    int64
 34  60371635            205 non-null    int64
 35  60371636            205 non-null    int64
 36  60371637            205 non-null    float64
 37  60371638            205 non-null    int64
 38  60371639            205 non-null    float64
 39  60371640            205 non-null    int64
 40  60371641            204 non-null    float64
 41  60371642            205 non-null    int64
 42  60371643            205 non-null    int64
 43  60371644            203 non-null    float64
 44  60371645            205 non-null    int64
 45  60371646            205 non-null    float64
 46  60371647            205 non-null    int64
 47  60371648            205 non-null    int64
 48  60371649            205 non-null    int64
 49  60371650            204 non-null    float64
 50  60371651            205 non-null    int64
```

```
 51  60371652            205 non-null    int64
 52  60371653            184 non-null    float64
 53  60371654            182 non-null    float64
 54  Zeitstempel         205 non-null    object
 55  Datensatz-Ursprung  205 non-null    object
dtypes: float64(11), int64(42), object(3)
memory usage: 91.3+ KB
```

Wir betrachten die fehlenden Werte genauer, um zu entscheiden, wie wir mit diesen Werten umgehen möchten.

In [23]:
```
msno.matrix(df_yz)
```

Out[23]: <Axes: >



Wir können feststellen, dass die meisten fehlenden Werte in den Fragen 60371653 und 60371653 vorkommen. Ansonsten existieren so gut wie keine fehlenden Werte. Da es sich bei den Fragen 60371653 und 60371653 um Fragen zur Einordnung der Befragten handelt, welche in den weiteren Auswertungen keinen großen EInfluss mehr haben, füllen wir diese Werte mit 0 auf. Außerdem bereinigen wir unseren Datensatz, indem wir die restlichen Stichproben mit fehlenden Daten entfernen.

In einem letzten Schritt wandeln wir die numerische Darstellung der Beobachtungen für die numerischen Skalafragen in ein einheitliches Format um. Hier wählen wir die Darstellung als Integer (Antwortmöglichkeiten zwischen 1 und 5).

In [25]:
```
df_yz = df_yz.dropna(subset=['60371632', '60371641', '60371644', '60371650']
df_yz.fillna(0, inplace=True)
df_yz[skala_ids] = df_yz[skala_ids].applymap(np.int64)
```

Übrig bleiben somit 200 Beobachtungen, welche im folgenden als Grundlage für unsere Analyse dienen.

In [26]:
```
df_yz.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 200 entries, 0 to 226
Data columns (total 56 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Bogen               200 non-null    int64
 1   60371597            200 non-null    int64
 2   60371598            200 non-null    int64
 3   60371599            200 non-null    int64
 4   60371600            200 non-null    int64
 5   60371601            200 non-null    int64
 6   60371602            200 non-null    int64
 7   60371603            200 non-null    int64
 8   60371604            200 non-null    int64
 9   60371605            200 non-null    int64
 10  60371606            200 non-null    int64
 11  60371607            200 non-null    int64
 12  60371608            200 non-null    int64
 13  60371609            200 non-null    int64
 14  60371612            200 non-null    object
 15  60371613            200 non-null    int64
 16  60371614            200 non-null    int64
 17  60371615            200 non-null    int64
 18  60371616            200 non-null    int64
 19  60371617            200 non-null    int64
 20  60371618            200 non-null    int64
 21  60371619            200 non-null    int64
 22  60371620            200 non-null    int64
 23  60371624            200 non-null    int64
 24  60371625            200 non-null    int64
 25  60371626            200 non-null    int64
 26  60371627            200 non-null    int64
 27  60371628            200 non-null    int64
 28  60371629            200 non-null    int64
 29  60371630            200 non-null    int64
 30  60371631            200 non-null    int64
 31  60371632            200 non-null    int64
 32  60371633            200 non-null    int64
 33  60371634            200 non-null    int64
 34  60371635            200 non-null    int64
 35  60371636            200 non-null    int64
 36  60371637            200 non-null    int64
 37  60371638            200 non-null    int64
 38  60371639            200 non-null    int64
 39  60371640            200 non-null    int64
 40  60371641            200 non-null    int64
 41  60371642            200 non-null    int64
 42  60371643            200 non-null    int64
 43  60371644            200 non-null    int64
 44  60371645            200 non-null    int64
 45  60371646            200 non-null    int64
 46  60371647            200 non-null    int64
 47  60371648            200 non-null    int64
 48  60371649            200 non-null    int64
 49  60371650            200 non-null    float64
 50  60371651            200 non-null    int64
```

```
 51  60371652            200 non-null    int64
 52  60371653            200 non-null    float64
 53  60371654            200 non-null    float64
 54  Zeitstempel         200 non-null    object
 55  Datensatz-Ursprung  200 non-null    object
dtypes: float64(3), int64(50), object(3)
memory usage: 89.1+ KB
```

Das einfache Pre Processing der Daten ist damit abgeschlossen. Die aufbereiteten speichern wir in einem parquet File ab. Zur einfacherern Weiterverarbeitung kann dieser Datensatz geladen werden

In [26]:
```
table_yz = pa.Table.from_pandas(df_yz)
pq.write_table(table_yz, 'results_preprocessed.parquet')
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```