Bibliotheken laden

Daten-Upload

In [2]:
```
import pandas as pd
```

In [3]:
```
Kundensicht = 'raw_data\Kundensicht.csv'
sep = ';'
encoding="ISO-8859-1"
```

In [4]:
```
# Es werden nur die ersten zwei Zeilen geladen, um die Header-Struktur zu überprüfe
df = pd.read_csv(Kundensicht, sep=sep, nrows=2, encoding=encoding)
cols = df.columns
df.head()
```

Out[4]:

	Unnamed: 0	KundenNr	Ort	GeoId
0	1	8715d06c5a081b4fc09a1e3c543a46c0	Oberfell	fc3f49f4a1e3f928b3e98c606ccb1da2
1	2	972ca3aeccb4df7d6702925cfb516847	Rengsdorf	fc19ae7cf6355599ca723f97daecdf06

In [5]:
```
# Hochladen von Daten ohne die erste Spalte
cols = df.columns
df = pd.read_csv(Kundensicht, sep=sep, usecols=cols[1:], encoding=encoding)
df.head(3)
```

Out[5]:

	KundenNr	Ort	GeoId
0	8715d06c5a081b4fc09a1e3c543a46c0	Oberfell	fc3f49f4a1e3f928b3e98c606ccb1da2
1	972ca3aeccb4df7d6702925cfb516847	Rengsdorf	fc19ae7cf6355599ca723f97daecdf06
2	10105a93b7fb41180b2986c790e79a1b	Oberelbert	5e4ec8e68ae5642f5aefc7c25ef2cf6e

Basisinformationen zum Datensatz

In [6]:
```
# Datendimensionen
df.shape
```

Out[6]:
```
(7661, 3)
```

In [7]:
```
# ob es leere Zellen gibt
df.isna().sum()
```

Out[7]:
```
KundenNr      0
Ort         104
GeoId         0
dtype: int64
```

In [8]:
```
#um die Zeilen anzuzeigen, in denen die Spalte „Ort“ leere Werte enthält:
empty_ort_df = df[df['Ort'].isna()]
display(empty_ort_df.head())
```

	KundenNr	Ort	GeoId
1568	282dbf77001d25ec8a108557ca6b6618	NaN	c7f66da1cae4f223b9bae717f05900f7
1569	7a797fc024702c30d510ff7c41797782	NaN	c7f66da1cae4f223b9bae717f05900f7
1570	efd1988b1bc4044c0f41819cd51a3472	NaN	c7f66da1cae4f223b9bae717f05900f7
1571	560303717a2b2cc75bc1f927f04c4f43	NaN	c7f66da1cae4f223b9bae717f05900f7
1572	8448a3a317b3b4be965078f286789d5f	NaN	c7f66da1cae4f223b9bae717f05900f7

In [9]:
```
# Was sind die einzigartigen GeoId-Spaltenwerte für diese Teilmenge?
empty_ort_df['GeoId'].unique()
```

Out[9]:
```
array(['c7f66da1cae4f223b9bae717f05900f7'], dtype=object)
```

In [10]:
```
# Finden einzigartiger GeoId-Werte für leere Orte
unique_geo_ids_for_empty_ort = empty_ort_df['GeoId'].unique()

# DataFrame-Filterung für Fälle, in denen Ort nicht leer ist und GeoId zu den einzi
non_empty_ort_with_geo_ids = df[df['GeoId'].isin(unique_geo_ids_for_empty_ort) & df

non_empty_ort_with_geo_ids
```

Out[10]:

KundenNr	Ort	GeoId

In [11]:
```
# leere Werte durch einen temporären, nicht realen Namen ersetzen
df['Ort'].fillna('Gotham City', inplace=True)
```

In [12]:
```
# Überprüfung des Datentyps
df.dtypes
```

Out[12]:
```
KundenNr    object
Ort         object
GeoId       object
dtype: object
```

In [13]:
```
df.describe()
```

Out[13]:

	KundenNr	Ort	GeoId
count	7661	7661	7661
unique	7610	814	7305
top	79143db83c8e10f0cb6af3e37d2f0e67	Koblenz	c7f66da1cae4f223b9bae717f05900f7
freq	4	1736	104

In [14]:
```
# Überprüfung der Standorte für den „beliebtesten“ Kunden – ob sie gleich oder unte
temp_df = df[df['KundenNr']=='79143db83c8e10f0cb6af3e37d2f0e67']
temp_df
```

Out[14]:

	KundenNr	Ort	GeoId
2254	79143db83c8e10f0cb6af3e37d2f0e67	Stahlhofen	1a756b82c4d6213e7feade71cdeafaa6
2255	79143db83c8e10f0cb6af3e37d2f0e67	Stahlhofen	1a756b82c4d6213e7feade71cdeafaa6
2256	79143db83c8e10f0cb6af3e37d2f0e67	Stahlhofen	1a756b82c4d6213e7feade71cdeafaa6
2257	79143db83c8e10f0cb6af3e37d2f0e67	Stahlhofen	1a756b82c4d6213e7feade71cdeafaa6

In [15]:
```
# Verdacht auf doppelte Werte
duplicates = df.duplicated()
duplicates.any()
```

Out[15]:
```
True
```

In [16]:
```
# Duplikate entfernen
df = df.drop_duplicates()
```

In [17]:
```
df.describe()
```

Out[17]:

	KundenNr	Ort	GeoId
count	7610	7610	7610
unique	7610	814	7305
top	8715d06c5a081b4fc09a1e3c543a46c0	Koblenz	c7f66da1cae4f223b9bae717f05900f7
freq	1	1724	103

Speichern des aktuellen DataFrame in einer CSV-Datei

In [18]:
```
df.to_csv('clean_raw_data\Kundensicht.csv', index=False)
```

In [ ]:
```

```