

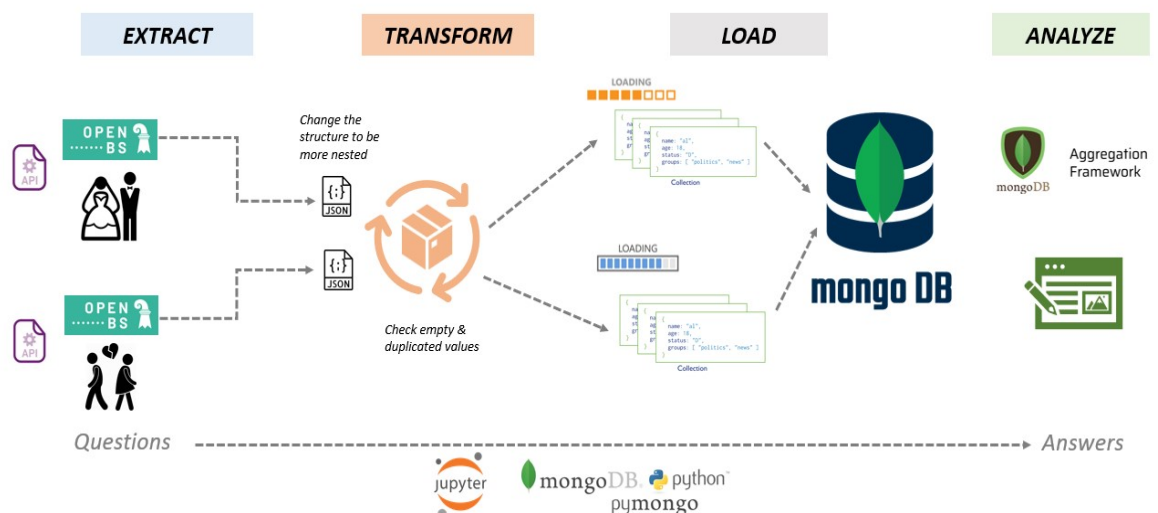
Marriage: happily ever after? An investigation into family status in Switzerland

Recently, the two Hollywood movie stars, Johny Depp and Amber Heard, have gotten public attention over several media channels on the controversial defamation trial after a long-time divorce. However, not all couples end up in nightmares after a happy wedding. Marriage symbolizes a big step and a new chapter in building a family in one's life. Indeed, marriage is a sweet dream of many people who find their Mr. or Mrs. Right and are ready to commit and embark on a lifelong relationship. Marriage, also known as wedlock or matrimony, refers to a legitimate and socially perceived joint between spouses. This relationship status institutes their rights and responsibilities, including with each other, their in-law families, and children. In contrast, divorce or dissolution of marriage is a legal procedure to terminate the marriage. Without further delay, let's dive into the topic.

1 Summary and overview

This project attempts to investigate the marriage and divorce trend in Switzerland. The next goal is to analyze in-depth concerning several factors such as year, age, gender, and nationality.

1.1 Methodology



The methodology of this paper follows an Extract Transform Load (ETL) process, as summarized in the figure above. The first step is to find open data sources with public API and fetch data. Next, I transformed the JSON data from a flat to a nested structure to make more complex queries later. This step helps to learn to build various aggregation pipelines with documents and sub-documents.

Afterward, the data is loaded to MongoDB. Lastly, several questions were answered based on numerous queries and visualizations. The questions are shown in the analysis chapter.

1.2 Data source & description

The data come from the Statistical Office of canton Basel-Stadt and Statistics of the natural population movement (BEVNAT) of the Federal Statistical Office (FSO). There are two links to collect data for this paper as follows:

- Marriage JSON data: <https://data.bs.ch/explore/dataset/100156/download/?format=json&timezone=Europe/Berlin&lang=en/>
- Divorce JSON data: <https://data.bs.ch/api/v2/catalog/datasets/100139/exports/json/>

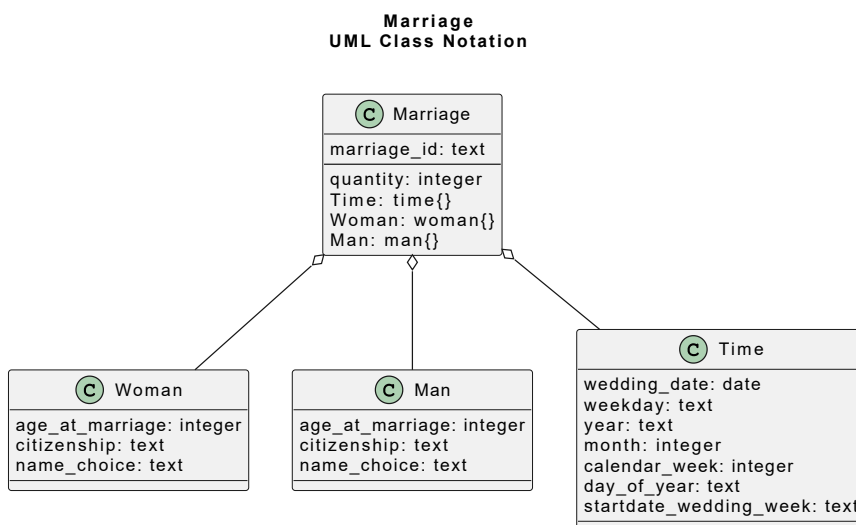
Both datasets provide observations on some aspects of marriage and divorce from 1979 to 2020 in Basel city. Although Basel-Stadt is just one of 26 cantons in Switzerland, the data still brings a snapshot of the overall trend. To be more concrete, the variable descriptions of two data collections are described below:

Data	Name	Description
Marriage	trauungsdat	wedding date
Marriage	jahr	year
Marriage	monat	month
Marriage	woche_in_jahr	calendar week
Marriage	datum_wochenstart_trauungsdatum	date of week start
Marriage	wochentag	weekday
Marriage	persalter_frau_vollendet	at of woman at marriage
Marriage	persalter_mann_vollendet	at of man at marriage
Marriage	namenswahl_frau	woman's choice for name
Marriage	namenswahl_mann	man's choice for name
Marriage	nationalitat_frau	woman's nationality
Marriage	nationalitat_mann	man's nationality
Marriage	anzahl	number of cases
Divorce	scheidungsdatum	divorce date
Divorce	jahr	year
Divorce	monat	month
Divorce	woche_in_jahr	calendar week
Divorce	datum_wochenstart_scheidungsdatum	date of week start
Divorce	tag_in_jahr	day of year
Divorce	persalter_frau_vollendet	age of woman at divorce
Divorce	persalter_mann_vollendet	age of man at divorce
Divorce	nationalitaet_frau	woman's nationality
Divorce	nationalitaet_mann	man's nationality
Divorce	anzahl	number of cases
Divorce	jahreehedauer_vollendet	marriage duration

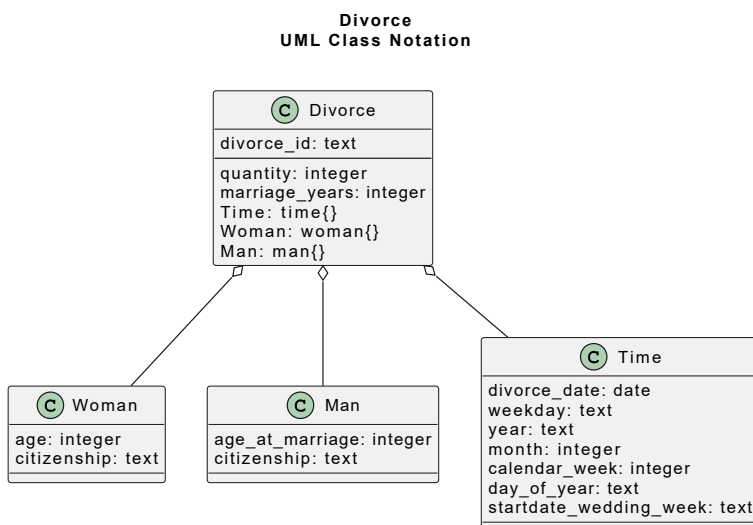
For the ease of querying, all variable names were translated from German to English before loading into the database.

1.3 Data modeling

The image below illustrates the static structure of each document through Class diagram in Unified Modeling Language.



Marriage and divorce are the two collections in the database `'dbresident'`. The attribute such as quantity (number of cases) is directly under the big class without nesting. Since a couple has one wife and one husband, each spouse will be one nested sub-document. There are attributes such as *age*, *citizenship*, or *choices of name* under their class. Time is another class containing the date of the event, month, and calendar week.



2 Requirements & Configuration

```
In [88]: ! pip3 list | findstr "pymongo dnspython pandas"
```

```
dnspython          2.2.1
pandas             1.4.2
pymongo            4.1.0
```

```
In [10]: import pymongo
from pprint import pprint
import pandas as pd
import requests
import urllib.json
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
import pathpy as pp
import networkx as nx
```

```
In [11]: # pandas configuration
pd.set_option('display.precision', 2)
pd.set_option('display.max_rows', 30)
pd.set_option('display.max_colwidth', 70)
# pd.describe_option('max_rows')
# pd.describe_option('precision')
# pd.describe_option('max_colwidth')
```

```
In [12]: # API and Database details
# ## population data
API_URL = "https://data.bs.ch/api/v2/catalog/datasets/100128/exports/json"
## marriage data
API_URL1 = "https://data.bs.ch/explore/dataset/100156/download/?format=json&timezone=Europe/Berlin"
## divorce data
API_URL2 = "https://data.bs.ch/api/v2/catalog/datasets/100139/exports/json"

CNX_STR = "localhost:27017"
# CNX_STR = "mongodb+srv://dbUser:dbUser@cluster0.uppfo.mongodb.net/test"

DB_NAME = "db_resident"
COLL_NAME = "population"
COLL_NAME1 = "marriage"
COLL_NAME2 = "divorce"
```

```
In [14]: # connection to MongoDB
client = pymongo.MongoClient(CNX_STR)
db = client[DB_NAME]
population = db[COLL_NAME]
marriage = db[COLL_NAME1]
divorce = db[COLL_NAME2]
```

3 ETL

3.1 Remove all existing documents

```
In [15]: # Reset collection
population.drop()
population.count_documents({})
marriage.drop()
marriage.count_documents({})
divorce.drop()
divorce.count_documents({})
```

```
Out[15]: 0
```

3.2 Fetch data

```
In [16]: # fetch JSON from API_URL
r = requests.get(API_URL)
data = json.loads(r.text)
r1 = requests.get(API_URL1)
data1 = json.loads(r1.text)
r2 = requests.get(API_URL2)
data2 = json.loads(r2.text)
```

```
In [17]: # A quick glance at data
# print("Population: " + 60*" - ")
# print(r.text[0:500])
print("Marriage: " + 60*" - ")
print(r1.text[0:500])
print("Divorce: " + 60*" - ")
print(r2.text[0:500])
```

```

Marriage: -----
[{"datasetid": "100156", "recordid": "918c7918307a63b7d55fe11497e86d6fb86e4469", "fields":
{"trauungsdat": "1983-05-13", "namenswahl_mann": "unbekannt", "monat": 5, "nationalitat_man
n": "CH", "persalter_mann_vollendet": 23, "tag_in_jahr": 133, "woche_in_jahr": 19, "anzahl":
1, "datum_wochenstart_trauungsdatum": "1983-05-09", "wochentag": "Fr", "namenswahl_frau": "un
bekannt", "persalter_frau_vollendet": 21, "jahr": "1983", "nationalitat_frau": "CH"}, "record
_timestamp": "2022-05-26T03:15:44.928+0
Divorce: -----
[{"scheidungsdatum": "1999-01-19", "jahr": "1999", "monat": 1, "woche_in_jahr": 3, "datum_woc
henstart_scheidungsdatum": "1999-01-18", "tag_in_jahr": 19, "wochentag": "Di", "persalter_fra
u_vollendet": 39, "persalter_mann_vollendet": 46, "nationalitaet_frau": "Ausl", "nationalitae
t_mann": "Ausl", "jahreehedauer_vollendet": 18, "anzahl": 1},{ "scheidungsdatum": "1999-01-1
9", "jahr": "1999", "monat": 1, "woche_in_jahr": 3, "datum_wochenstart_scheidungsdatum": "199
9-01-18", "tag_in_jahr": 19, "wochent

```

3.3 Change data structure

As shown above, the datasets have a flat structure. Also, the data are all in German. Hence, let's make a new nested structure for these JSON files in English:

```

In [18]: # Change structure of population data
data_transformed = []
for i in data:
    new_structure = {
        'district_id':i.get('wohnviertel_id'),
        'residential_area':{'municipality':i.get('gemeinde'),
                             'district':i.get('wohnviertel')},
        'demographic':{'age':i.get('person_alter'),
                        'citizenship':i.get('staatsangehoerigkeit'),
                        'gender':i.get('geschlecht')},
        'quantity':i.get('anzahl'),
        'time': {'date': i.get('datum'),
                  'year': i.get('jahr')}
    }
    data_transformed.append(new_structure)

print(data_transformed[0:2])

[{'district_id': '01', 'residential_area': {'municipality': 'Basel', 'district': 'Altstadt Gr
ossbasel'}, 'demographic': {'age': 0, 'citizenship': 'CH', 'gender': 'M'}, 'quantity': 5, 'ti
me': {'date': '2020-12-31', 'year': '2020'}}, {'district_id': '01', 'residential_area': {'mun
icipality': 'Basel', 'district': 'Altstadt Grossbasel'}, 'demographic': {'age': 1, 'citizens
ip': 'A', 'gender': 'M'}, 'quantity': 1, 'time': {'date': '2020-12-31', 'year': '2020'}}]

```

Because the URL for the marriage data is slightly different from the other datasets, only variables belonging to the documents *'field'* are relevant in this case. Thus, let's extract the *'field'* first and then create the new structure:

```

In [19]: # For marriage data, get the elements inside "fields" first
data1_fields = []

for i in range(len(data1)):
    new = data1[i]["fields"]
    data1_fields.append(new)

# Change structure of marriage data
data_transformed1 = []
for i in data1_fields:
    new_structure1 = {
        'woman':{'age_at_marriage':i.get('persalter_frau_vollendet'),
                  'citizenship':i.get('nationalitat_frau'),
                  'name_choice':i.get('namenswahl_frau')},
        'man':{'age_at_marriage':i.get('persalter_mann_vollendet'),
                'citizenship':i.get('nationalitat_mann'),
                'name_choice':i.get('namenswahl_mann')},
        'time': {'wedding_date': i.get('trauungsdat'),
                  'weekday': i.get('wochentag'),
                  'year': i.get('jahr'),

```

```

        'month': i.get('monat'),
        'calendar_week': i.get('woche_in_jahr'),
        'day_of_year': i.get('tag_in_jahr'),
        'startdate_wedding_week': i.get('datum_wochenstart_trauungsdatum')),
    'quantity': i.get('anzahl'),
}
data_transformed1.append(new_structure1)

# Check
print(data_transformed1[0:2])

```

```

[{'woman': {'age_at_marriage': 21, 'citizenship': 'CH', 'name_choice': 'unbekannt'}, 'man': {'age_at_marriage': 23, 'citizenship': 'CH', 'name_choice': 'unbekannt'}, 'time': {'wedding_date': '1983-05-13', 'weekday': 'Fr', 'year': '1983', 'month': 5, 'calendar_week': 19, 'day_of_year': 133, 'startdate_wedding_week': '1983-05-09'}, 'quantity': 1}, {'woman': {'age_at_marriage': 34, 'citizenship': 'CH', 'name_choice': 'unbekannt'}, 'man': {'age_at_marriage': 34, 'citizenship': 'CH', 'name_choice': 'unbekannt'}, 'time': {'wedding_date': '1983-05-13', 'weekday': 'Fr', 'year': '1983', 'month': 5, 'calendar_week': 19, 'day_of_year': 133, 'startdate_wedding_week': '1983-05-09'}, 'quantity': 1}]

```

```

In [20]: # Change structure of divorce data
data_transformed2 = []
for i in data2:
    new_structure2 = {
        'woman': {'age': i.get('persalter_frau_vollendet'),
                    'citizenship': i.get('nationalitaet_frau')},
        'man': {'age': i.get('persalter_mann_vollendet'),
                 'citizenship': i.get('nationalitaet_mann')},
        'marriage_years': i.get('jahreehedauer_vollendet'),
        'time': {'divorce_date': i.get('scheidungsdatum'),
                  'weekday': i.get('wochentag'),
                  'year': i.get('jahr'),
                  'month': i.get('monat'),
                  'calendar_week': i.get('woche_in_jahr'),
                  'day_of_year': i.get('tag_in_jahr'),
                  'startdate_wedding_week': i.get('datum_wochenstart_scheidungsdatum')},
        'quantity': i.get('anzahl'),
    }
    data_transformed2.append(new_structure2)

# Check
print(data_transformed2[0:2])

```

```

[{'woman': {'age': 39, 'citizenship': 'Ausl'}, 'man': {'age': 46, 'citizenship': 'Ausl'}, 'marriage_years': 18, 'time': {'divorce_date': '1999-01-19', 'weekday': 'Di', 'year': '1999', 'month': 1, 'calendar_week': 3, 'day_of_year': 19, 'startdate_wedding_week': '1999-01-18'}, 'quantity': 1}, {'woman': {'age': 40, 'citizenship': 'CH'}, 'man': {'age': 51, 'citizenship': 'CH'}, 'marriage_years': 17, 'time': {'divorce_date': '1999-01-19', 'weekday': 'Di', 'year': '1999', 'month': 1, 'calendar_week': 3, 'day_of_year': 19, 'startdate_wedding_week': '1999-01-18'}, 'quantity': 1}]

```

3.4 Load into MongoDB

```

In [21]: # insert the list of population into MongoDB collection "population"
population.insert_many(data_transformed);

# insert the list of marriage into MongoDB collection "marriage"
marriage.insert_many(data_transformed1);

# insert the list of divorce into MongoDB collection "divorce"
divorce.insert_many(data_transformed2);

```

```

In [22]: # count number of documents inserted
coll = []

for collection in db.list_collection_names():
    count = db[collection].count_documents({})
    coll.append({'coll_name': collection, 'number_docs': count})

```

```
coll_count = pd.DataFrame.from_records(coll)
coll_count
```

```
Out[22]:
```

	coll_name	number_docs
0	divorce	17350
1	population	319301
2	marriage	44002

After inserting data into the MongoDB collection, we can check the collection names and the number of corresponding documents under each collection as above. There are 44,002 and 17,350 documents in 'marriage' and 'divorce' respectively.

```
In [23]: # A quick look at one document
pprint(marriage.find_one())

{'_id': ObjectId('628f84460cba82c58e34d1e4'),
 'man': {'age_at_marriage': 23,
        'citizenship': 'CH',
        'name_choice': 'unbekannt'},
 'quantity': 1,
 'time': {'calendar_week': 19,
        'day_of_year': 133,
        'month': 5,
        'startdate_wedding_week': '1983-05-09',
        'wedding_date': '1983-05-13',
        'weekday': 'Fr',
        'year': '1983'},
 'woman': {'age_at_marriage': 21,
        'citizenship': 'CH',
        'name_choice': 'unbekannt'}}
```

```
In [24]: # A quick look in dataframe format
c = marriage.aggregate([
    {"$limit": 5},
])

pd.DataFrame(c)
```

```
Out[24]:
```

	_id	woman	man	time	quantity
0	628f84460cba82c58e34d1e4	{'age_at_marriage': 21, 'citizenship': 'CH', 'name_choice': 'unbek...	{'age_at_marriage': 23, 'citizenship': 'CH', 'name_choice': 'unbek...	{'wedding_date': '1983-05-13', 'weekday': 'Fr', 'year': '1983', 'm...	1
1	628f84460cba82c58e34d1e5	{'age_at_marriage': 34, 'citizenship': 'CH', 'name_choice': 'unbek...	{'age_at_marriage': 34, 'citizenship': 'CH', 'name_choice': 'unbek...	{'wedding_date': '1983-05-13', 'weekday': 'Fr', 'year': '1983', 'm...	1
2	628f84460cba82c58e34d1e6	{'age_at_marriage': 27, 'citizenship': 'CH', 'name_choice': 'unbek...	{'age_at_marriage': 27, 'citizenship': 'CH', 'name_choice': 'unbek...	{'wedding_date': '1983-05-14', 'weekday': 'Sa', 'year': '1983', 'm...	1
3	628f84460cba82c58e34d1e7	{'age_at_marriage': 28, 'citizenship': 'Aus', 'name_choice': 'unb...	{'age_at_marriage': 26, 'citizenship': 'CH', 'name_choice': 'unbek...	{'wedding_date': '1983-05-16', 'weekday': 'Mo', 'year': '1983', 'm...	1
4	628f84460cba82c58e34d1e8	{'age_at_marriage': 30, 'citizenship': 'CH', 'name_choice': 'unbek...	{'age_at_marriage': 38, 'citizenship': 'CH', 'name_choice': 'unbek...	{'wedding_date': '1983-05-20', 'weekday': 'Fr', 'year': '1983', 'm...	1

Each column or sub-document shows a nested structure with several fields as planned to serve the purpose of this project.

3.5 Transform

3.5.1 Missing value

```
In [25]: # Check for empty value
print("1.", "~"*30)
# find document where field value is None
for d in marriage.find({"quantity": 0}):
    pprint(d)

print("2.", "~"*30)
# find document where field value is type 10 == None
for d in marriage.fields.find({"man": {"$type": 10}}):
    pprint(d)

print("3.", "~"*30)
# find document where filed name is missing
for d in marriage.find({"time": {"$exists": False}}):
    pprint(d)

1. ~~~~~
2. ~~~~~
3. ~~~~~
```

3.5.2 Duplicate value

```
In [26]: # Delete any duplicate value
pipeline = [
    { "$group": {
        "_id": {"key": "$key"},
        "duplicates": { "$addToSet": "$_id" },
        "count": { "$sum" : 1 }
    }
},
    { "$match": {
        "count": { "$gt": 1 }
    }
}
]

for d in marriage.aggregate(pipeline):
    to_be_deleted = d["duplicates"][1:] # all except the first item should be deleted
    db.inventory.delete_many({
        "_id": {"$in": to_be_deleted}
    })
```

```
In [27]: print(marriage.count_documents({}))

44002
```

3.5.3 Flatten

As shown in part 3.4, the data was changed to the nested structure by purpose. Both *marriage* and *divorce* collections include data from 1979 to 2020. Hence, I flatten the form of both groups to quickly compare them.

```
In [28]: # flatten the needed fields in collection marriage
flatten = {"$addFields": {
    "year": "$time.year",
    "wedding_date": "$time.wedding_date",
    "woman_age": "$woman.age_at_marriage",
    "man_age": "$man.age_at_marriage",
    "woman_citizenship": "$woman.citizenship",
    "man_citizenship": "$man.citizenship",
    "woman_name_choice": "$woman.name_choice",
    "man_name_choice": "$man.name_choice",
}}

project = {"$project": {"_id": 0, "woman": 0,
```



```

        "man": 0, "time":0}} # remove nested

sort = {"$sort": {'year': 1}}
limit = {"$limit": 10}
pipeline = [flatten, project, sort, limit]

cursor = client[DB_NAME].marriage.aggregate(pipeline)

# change to df
df_marriage_flatten = pd.DataFrame(cursor)
df_marriage_flatten

```

Out[28]:

	quantity	year	wedding_date	woman_age	man_age	woman_citizenship	man_citizenship	woman_name_ch
0	1	1979	1979-01-05	24	26	CH	CH	unbek
1	1	1979	1979-12-27	27	46	Ausl	CH	unbek
2	1	1979	1979-01-25	17	19	Ausl	Ausl	unbek
3	1	1979	1979-01-05	38	52	CH	CH	unbek
4	1	1979	1979-12-20	27	30	CH	CH	unbek
5	1	1979	1979-01-12	41	47	CH	CH	unbek
6	1	1979	1979-01-26	20	22	CH	CH	unbek
7	1	1979	1979-01-25	23	36	CH	CH	unbek
8	1	1979	1979-12-20	31	31	CH	CH	unbek
9	1	1979	1979-12-14	52	56	CH	CH	unbek

In [29]:

```

# flatten the needed fields in collection divorces
flatten = {"$addFields": {
    "year": "$time.year",
    "divorce_date": "$time.divorce_date",
    "woman_age": "$woman.age",
    "man_age": "$man.age",
    "woman_citizenship": "$woman.citizenship",
    "man_citizenship": "$man.citizenship"
}}

project = {"$project": {"_id": 0, "woman": 0,
                        "man": 0, "time":0}} # remove nested

sort = {"$sort": {'year': 1}}
limit = {"$limit": 10}
pipeline = [flatten, project, sort, limit]

cursor = client[DB_NAME].divorce.aggregate(pipeline)

# change to df
df_divorce_flatten = pd.DataFrame(cursor)
df_divorce_flatten

```

Out[29]:

	marriage_years	quantity	year	divorce_date	woman_age	man_age	woman_citizenship	man_citizenship
0	0	455	1979	1979-01-01	0	0	unbekannt	unbekannt
1	0	426	1980	1980-01-01	0	0	unbekannt	unbekannt
2	0	432	1981	1981-01-01	0	0	unbekannt	unbekannt
3	0	442	1982	1982-01-01	0	0	unbekannt	unbekannt
4	0	446	1983	1983-01-01	0	0	unbekannt	unbekannt
5	11	1	1984	1984-01-17	36	39	CH	CH
6	8	1	1984	1984-01-17	38	34	CH	CH
7	2	1	1984	1984-01-17	24	32	CH	CH
8	30	1	1984	1984-01-17	51	50	CH	CH
9	27	1	1984	1984-01-12	53	50	CH	CH

The divorce data was aggregated from 1979 to 1983, while the marriage data differs in these corresponding years. Therefore, we should only compare marriage and divorce in terms of age and citizenship from 1984 forwards.

4 Data analysis

This chapter analyzes in-depth, diverse factors of marriage and divorce. Each section's name indicates one question. The corresponding queries with the aggregation pipeline generate a table to answer that question. It is challenging to recognize patterns only with tabular format (Pandas dataframe) in some parts. Hence, data visualizations help spot insights, trends, and outliers. This section flow starts with a general overview and then zoom-in on more detailed information. Also, there are mixtures of marriage and divorce to enable readers easier to compare.

4.1 How does the number of marriages and divorces change over the years?

First of all, let's check the movement of marriage in comparison to divorce.

```
In [30]: # Find total marriages per year
c = marriage.aggregate([
    {"$group": { "_id": "$time.year",
                  "number_marriages": {"$sum": "$quantity"} }},
    {"$project": {"year": "$_id",
                  "_id": 0,
                  "number_marriages": "$number_marriages"}},
    {"$sort": {"year": 1}},
    {}
])

df_marriage_year = pd.DataFrame(c)

# Find total divorce per year
c = divorce.aggregate([
    {"$match": {"quantity": {"$exists": True}}},
    {"$group": { "_id": "$time.year",
                  "number_divorce": {"$sum": "$quantity"} }},
    {"$project": {"year": "$_id",
                  "_id": 0,
                  "number_divorce": "$number_divorce"}},
    {"$sort": {"year": 1}},
    {}
])

df_divorce_year = pd.DataFrame(c)
```

```
# dataframe with both cases of marriage & divorce by year
df_m_d_year = pd.merge(df_marriage_year, df_divorce_year, on='year')
df_m_d_year['marriage_subtract_divorce'] = df_m_d_year['number_marriages'] \
    - df_m_d_year['number_divorce']
df_m_d_year.head()
```

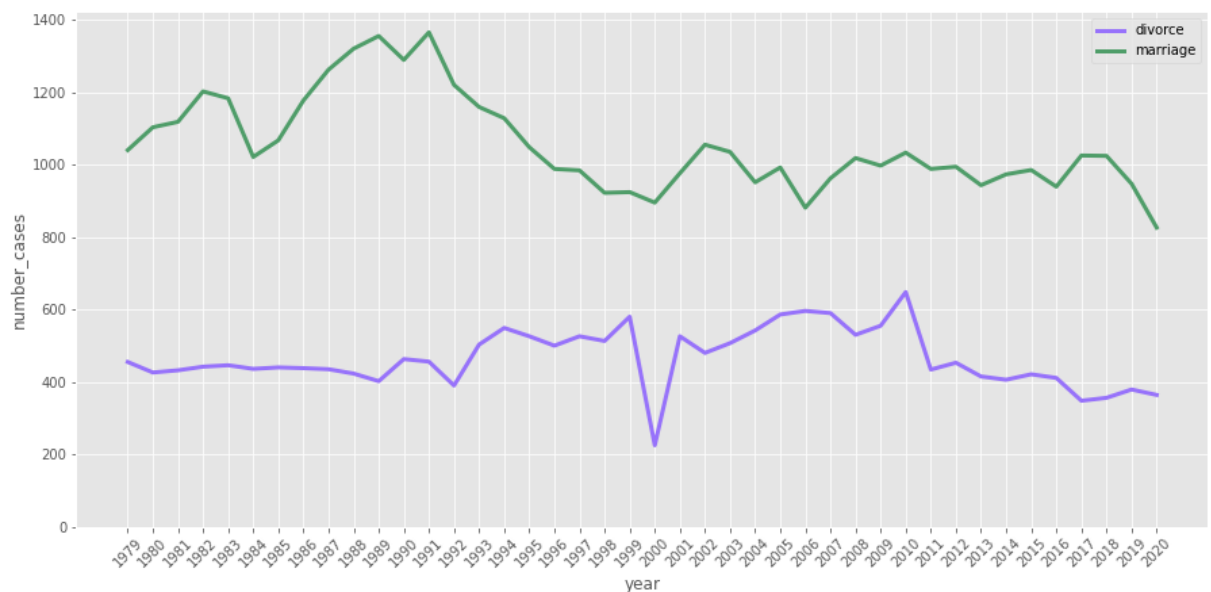
```
Out[30]:
```

	year	number_marriages	number_divorce	marriage_subtract_divorce
0	1979	1040	455	585
1	1980	1103	426	677
2	1981	1118	432	686
3	1982	1202	442	760
4	1983	1183	446	737

```
In [31]: # Plot the marriages & divorces by year
plt.style.use('ggplot')
plt.figure(figsize = (15,7))
ax1 = sns.lineplot(data=df_m_d_year,
    x="year", y="number_divorce",
    label='divorce', color="#9772FB",
    linewidth = 3)
ax1.set_ylabel("number_cases")

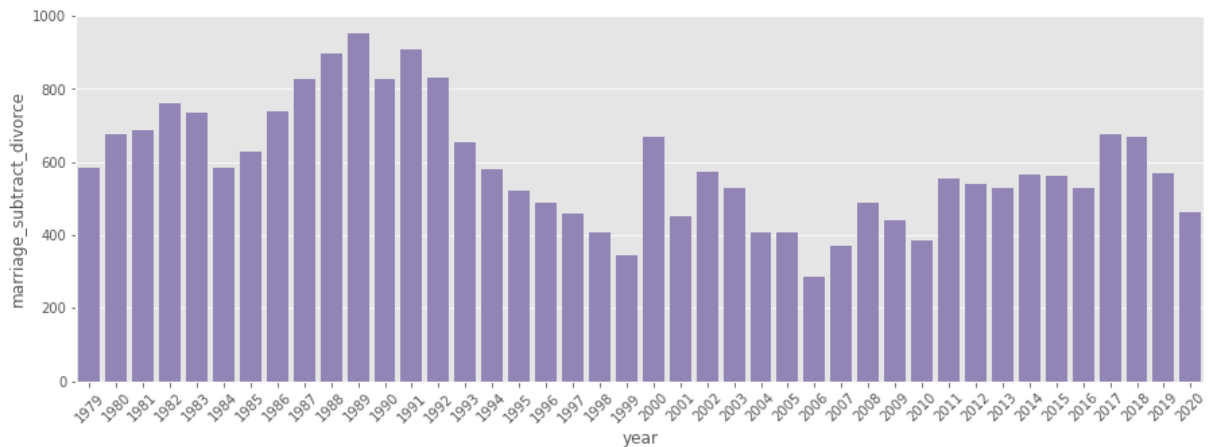
ax2 = sns.lineplot(data=df_m_d_year,
    x="year", y="number_marriages",
    label="marriage", color="#4F9D69",
    linewidth = 3)
ax1.set_ylim(ymin=0)
ax2.set_ylim(ymin=0)

locs, labels = plt.xticks()
plt.setp(labels, rotation=45)
plt.show()
```



The plot above displays a fluctuating trend in marriage over the years. Interestingly, marriage cases surged after 1984, peaked in 1991, and declined sharply till 2000. Also, fewer couples are getting married overall in the 2000s compared to the past. The tendency to drop the number of marriages persists as the matrimony loses its appeal to couples. One main reason could be the massive income tax penalty added together for tax calculation after marriage. There had been initiatives for ending unequal taxation of marital couples, yet, they were not passed by public votes (according to Swiss news). On the other hand, the divorce trendline appears to be more stable, with approximately 400 to 600 cases per year. Nevertheless, there was a dip in divorce occurrences in 2000 and a peak ten years later.

```
In [32]: # Plot difference cases of marriage & divorce over years
plt.style.use('ggplot')
plt.figure(figsize = (15,5))
ax = sns.barplot(x="year",
                 y="marriage_subtract_divorce",
                 data=df_m_d_year, color="#8E7DBE")
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)
plt.show()
```



The bar chart above presents the difference between each year's weddings and marital breakups. At the end of the 80s and beginning of the 90s, many marriages happened more than the split cases. In contrast, in 2006, with a shorter bar, many couples dissolved their relationship, and not so many united in the legal wedlock.

4.2 Which months do people tend to marry and divorce?

```
In [33]: # Find total marriages per month
c = marriage.aggregate([
    {"$match": {"time.month": {"$gt": 0}}},
    {"$group": { "_id": "$time.month",
                  "number_marriage": {"$sum": "$quantity"} }},
    {"$project": {"month": "$_id", "_id": 0,
                  "number_marriage": "$number_marriage"}},
    {"$sort": {"month": 1}}
])

df_marriage_month = pd.DataFrame(c)

# Find total divorces per month
c = divorce.aggregate([
    {"$match": {"time.month": {"$gt": 0}}},
    {"$group": { "_id": "$time.month",
                  "number_divorce": {"$sum": "$quantity"} }},
    {"$project": {"month": "$_id", "_id": 0,
                  "number_divorce": "$number_divorce"}},
    {"$sort": {"month": 1}}
])

df_divorce_month = pd.DataFrame(c)

# dataframe with marriage & divorce by month
df_m_d_month = pd.merge(df_marriage_month, df_divorce_month, on='month')
df_m_d_month
```

```
Out[33]:
```

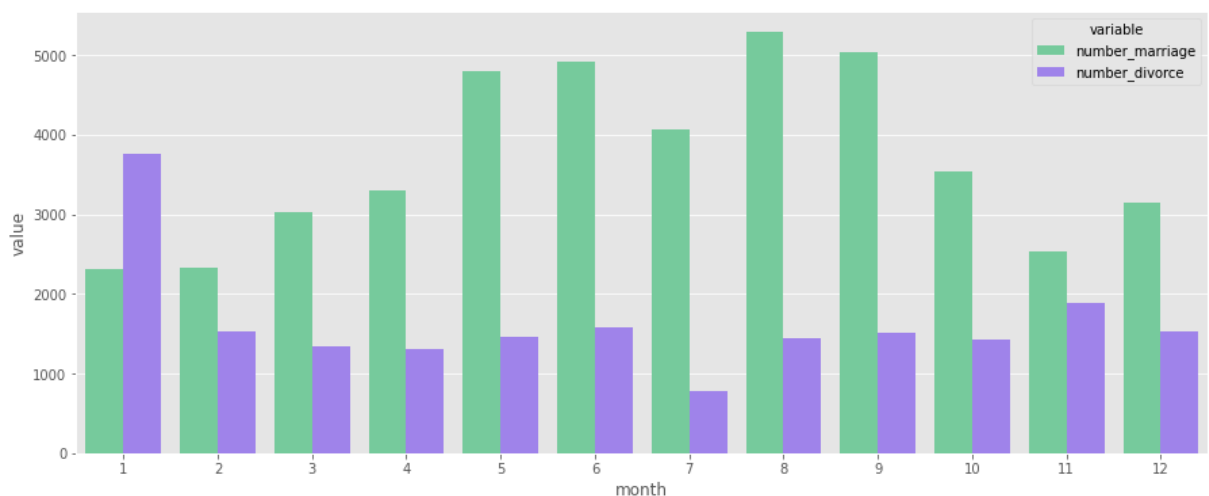
	month	number_marriage	number_divorce
0	1	2311	3760
1	2	2327	1527
2	3	3033	1347
3	4	3305	1313
4	5	4804	1462
5	6	4915	1587
6	7	4063	778
7	8	5289	1444
8	9	5039	1509
9	10	3542	1419
10	11	2534	1880
11	12	3152	1527

```
In [34]: # Melt data to fit with seaborn
df_m_d_month_melt = pd.melt(df_m_d_month, id_vars=["month"])

# color
color = ['#68D89B', '#9772FB']

# Plot marriages in different months
plt.style.use('ggplot')
plt.figure(figsize = (15,6))
ax = sns.barplot(x="month",
                 y="value",
                 hue="variable",
                 data=df_m_d_month_melt,
                 palette=color)

plt.show()
```



Months with better weather, such as May, June, August, July and September, tend to be favored by couples to organize their wedding. On the other hand, more spouses split up in January, whereas July has fewer divorce circumstances.

4.3 Which weekdays do people tend to marry and divorce?

```
In [37]: # Find total marriages per weekday
c = marriage.aggregate([
    {"$group": { "_id": "$time.weekday",
                  "number_marriage": {"$sum": "$quantity"} }},
```

```

    {"$project": {"weekday": "$_id", "_id": 0,
                  "number_marriage": "$number_marriage"}},
    {"$sort": {"number_marriage": -1}}
  ])

df_marriage_weekday = pd.DataFrame(c)

# Find total divorces per weekday
c = divorce.aggregate([
    {"$group": { "_id": "$time.weekday",
                  "number_divorce": {"$sum": "$quantity"} }},
    {"$project": {"weekday": "$_id", "_id": 0,
                  "number_divorce": "$number_divorce"}},
    {"$sort": {"number_divorce": -1}}
  ])

df_divorce_weekday = pd.DataFrame(c)

# dataframe
df_m_d_weekday = pd.merge(df_marriage_weekday, df_divorce_weekday, on='weekday')
df_m_d_weekday

```

Out[37]:

	weekday	number_marriage	number_divorce
0	Fr	20940	2839
1	Do	8737	2294
2	Mo	6372	2608
3	Di	4308	8564
4	Sa	2745	860
5	Mi	1019	2353
6	So	193	35

In [36]:

```

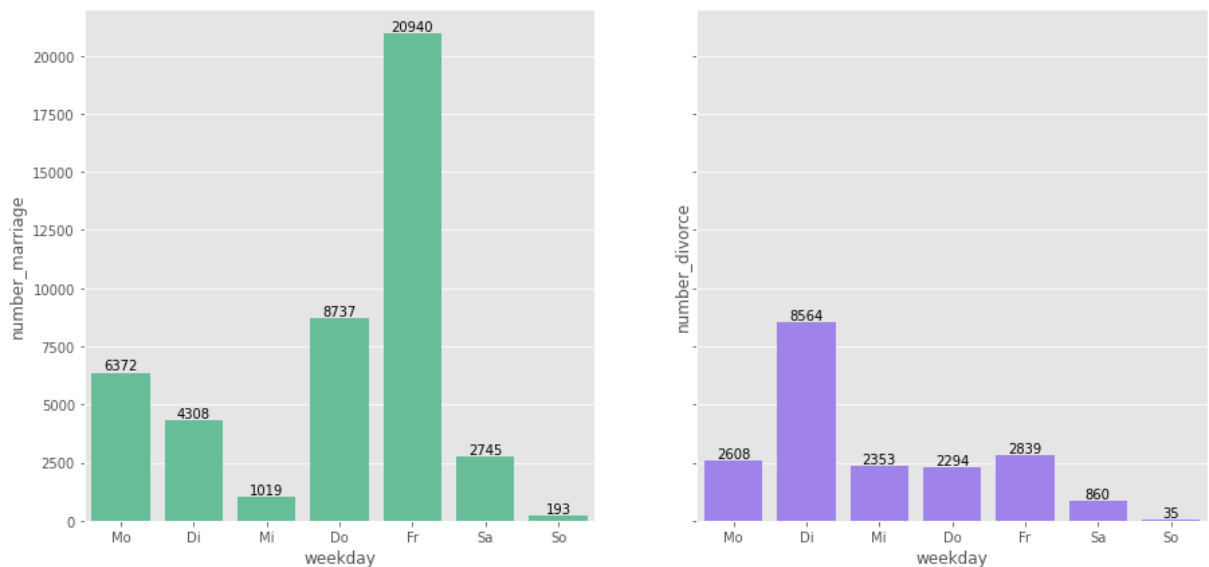
# create subplot side by side, sharey to have same y axis
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 7), sharey=True)

# plot a bar chart with given order of bars as days in a week
ax1 = sns.barplot(
    x="weekday", y="number_marriage",
    data=df_marriage_weekday,
    order=["Mo", "Di", "Mi", "Do", "Fr", "Sa", "So"],
    color='#57cc99', ax=ax1);
ax1.bar_label(ax1.containers[0])

ax2 = sns.barplot(
    x="weekday", y="number_divorce",
    data=df_divorce_weekday,
    order=["Mo", "Di", "Mi", "Do", "Fr", "Sa", "So"],
    color='#9772FB', ax=ax2);
ax2.bar_label(ax2.containers[0])

plt.show()

```



Friday is far more popular than other days to celebrate a wedding. Besides, Thursday is still the second most common day but not Saturday. Surprisingly, more couples married on Monday than Wednesday and Saturday. There is no further information from the data source. Still, the main reason might be that these marriage dates are registered dates at the civil office. That's why people tend to have it on the weekdays than the weekend. So many divorces happened on Tuesday as it recurred more than on other days.

4.4 How do couples change their names after the wedding?

Many countries and cultures have different regulations and traditions on changing names after marriage. Even though this custom might seem normal in Switzerland, people in other cultures do not ever change their last name due to weddings. The tradition started in the ninth century when authorities considered family laws that a wife and a husband should be "one." Hence, the women took the husband's surname as the concept of coverture (meaning cover by). Whether to keep a maiden name or change to a spouse's name can be substantial for many people regarding identity, administration, and value. This section explores the choices of names for couples. First, let's check unique values in the document:

```
In [38]: # Unique value for name choices
print(marriage.distinct("man.name_choice"))
print(marriage.distinct("woman.name_choice"))
```

```
['andere namensführung (ausländisches Recht)', 'führt ausschliesslich seinen bisherigen Name n', 'nimmt den Namen der Frau an', 'stellt dem Namen der Frau seinen bisherigen voran (bis 2012)', 'unbekannt']
['andere Namensführung (ausländisches Recht)', 'führt ausschliesslich ihren bisherigen Name n', 'nimmt den Namen des Mannes an', 'stellt dem Namen des Mannes ihren bisherigen voran (bis 2012)', 'unbekannt']
```

Translation for the choices of name are as follows:

- andere Namensführung (ausländisches Recht): other name (foreign law)
- führt ausschliesslich ihren bisherigen Namen: use previous name
- nimmt den Namen des Mannes/der Frau an: take the husband's/wife's name
- stellt dem Namen des Mannes/der Frau ihren bisherigen voran (bis 2012): puts the husband's/wife's name in front of her previous ones (until 2012)
- unbekannt: unknown

```
In [39]: # Caculate the total cases in marriage
c = marriage.aggregate([
    "$group": {"_id": "null", "total": {"$sum": "$quantity"}}
])
total = pd.DataFrame(c).total.iloc[0]
```

```
# Convert total variable from np int64 to python int
total = np.int64(total).item()
total
```

Out[39]: 44314

```
In [40]: # Find total marriages per women's choice of name
group = {"$group": {
    "_id": "$woman.name_choice",
    "number_marriages": {"$sum": "$quantity"},
}}

project = {"$project": {
    "woman_name": "$_id",
    "_id": 0,
    "number_marriages": "$number_marriages",
    "percent": {
        "$multiply": [{"$divide": ["$number_marriages", total]], 100}}
    }
}

sort = {"$sort": {"number_marriages": 1}}
pipeline = [group, project, sort]
c = marriage.aggregate(pipeline)

df_woman_name = pd.DataFrame(c)
df_woman_name
```

Out[40]:

	woman_name	number_marriages	percent
0	andere Namensführung (ausländisches Recht)	685	1.55
1	stellt dem Namen des Mannes ihren bisherigen voran (bis 2012)	3784	8.54
2	führt ausschliesslich ihren bisherigen Namen	4596	10.37
3	nimmt den Namen des Mannes an	12933	29.18
4	unbekannt	22316	50.36

```
In [41]: # Find total marriages per men's choice of name
group = {"$group": {
    "_id": "$man.name_choice",
    "number_marriages": {"$sum": "$quantity"},
}}

project = {"$project": {
    "man_name": "$_id",
    "_id": 0,
    "number_marriages": "$number_marriages",
    "percent": {
        "$multiply": [{"$divide": ["$number_marriages", total]], 100}}
    }
}

sort = {"$sort": {"number_marriages": 1}}
pipeline = [group, project, sort]
c = marriage.aggregate(pipeline)

df_man_name = pd.DataFrame(c)
df_man_name
```


Out[41]:

	man_name	number_marriages	percent
0	andere namensführung (ausländisches Recht)	125	0.28
1	nimmt den Namen der Frau an	402	0.91
2	stellt dem Namen der Frau seinen bisherigen voran (bis 2012)	432	0.97
3	führt ausschliesslich seinen bisherigen Namen	21040	47.48
4	unbekannt	22315	50.36

In [42]:

```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(20,14))

# Pie chart woman marriage name choice:
labels1 = df_woman_name.woman_name
sizes1 = df_woman_name.percent

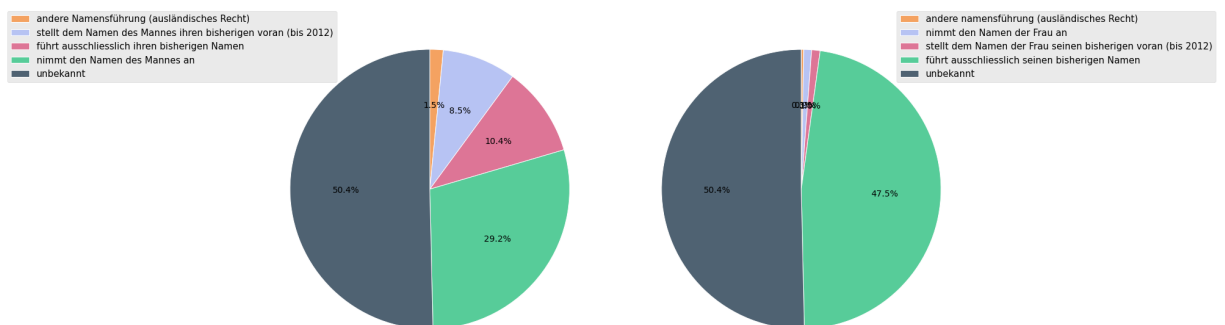
# Create a set of colors
colors = ['#F4A261', '#B7C3F3', '#DD7596', '#57cc99', '#4F6272']

pie = ax1.pie(sizes1, labels=None, autopct='%1.1f%%',
              startangle=90, colors=colors,
              counter-clock=False,
              wedgeprops={'edgecolor': 'white', 'linewidth': 1},
              textprops={'fontsize': 14}, center = (-2,0))
ax1.axis('equal') # Equal aspect ratio ensures pie is drawn as a circle.
ax1.legend(pie[0],labels1, bbox_to_anchor=(0.2,0.8),
           loc=1, fontsize=15,
           bbox_transform=plt.gcf().transFigure)

# Pie chart man marriage name choice:
labels2 = df_man_name.man_name
sizes2 = df_man_name.percent

pie2 = ax2.pie(sizes2, labels=None, autopct='%1.1f%%',
               startangle=90, colors=colors,
               counter-clock=False,
               wedgeprops={'edgecolor': 'white', 'linewidth': 1},
               textprops={'fontsize': 14}, center = (2,0))
ax2.axis('equal')
ax2.legend(pie2[0],labels2, bbox_to_anchor=(1.2,0.8),
           loc=1, fontsize=15,
           bbox_transform=plt.gcf().transFigure)

plt.show()
```



We do not know 50.4% of the cases how names were changed. Until 2012, people could add the husband's or wife's name besides their family name. Almost 30% of women took their husbands' names, while 10% decided to maintain their names. On the other hand, nearly half of the data indicates that men preferred to keep their names. Interestingly, 432 husbands, or around 1%, changed their names after their wives.

4.5 How has the name choice of married women changed over the years?

In recent years, did the rise of the feminist movement affect the name choice of married women? Even though we do not have more variables to answer this question, it's still essential to examine the trend from 1979 till today. Hence, let's look at the data through the years to answer this question:

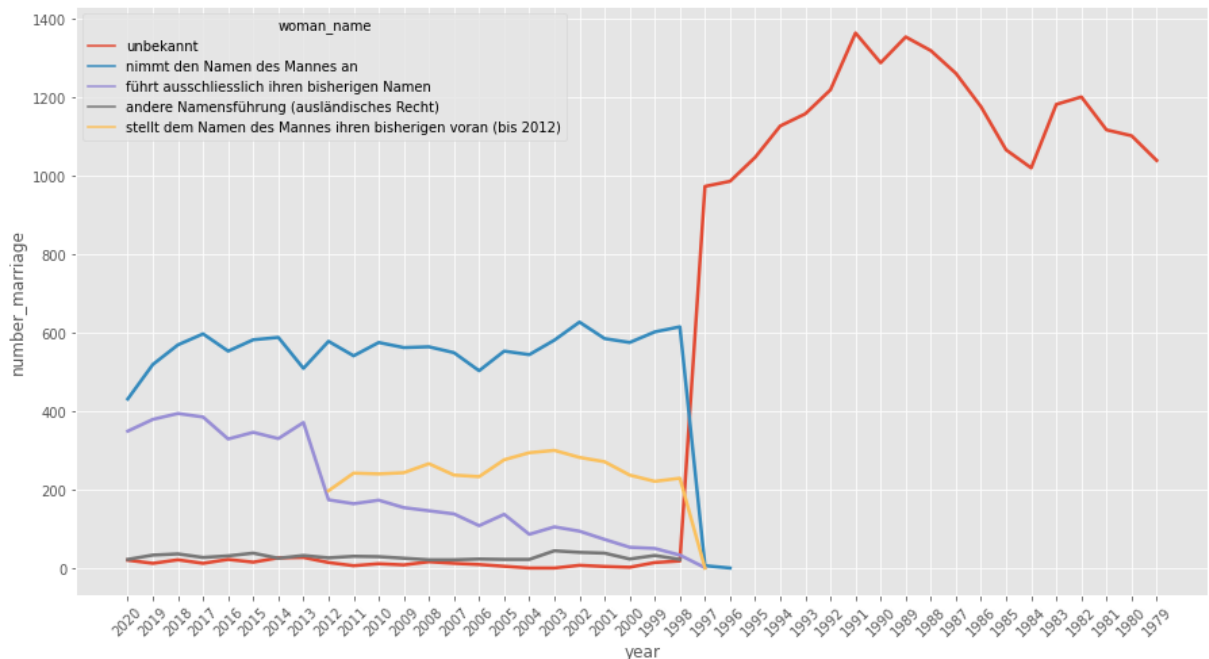
```
In [43]: # A further look into women's name choice over years
c = marriage.aggregate([
    {"$group" :
     {"_id": {
         "woman_name": "$woman.name_choice",
         "year": "$time.year"},
        "number_marriage": {"$sum": "$quantity"}}},
    {"$project": {
        "_id": 0,
        "year": "$_id.year",
        "woman_name": "$_id.woman_name",
        "number_marriage": "$number_marriage"
    }},
    {"$sort": {"year": -1}}
])

df_wname_year = pd.DataFrame(c)
df_wname_year.head(n=8)
```

```
Out[43]:
```

	year	woman_name	number_marriage
0	2020	unbekannt	21
1	2020	nimmt den Namen des Mannes an	432
2	2020	führt ausschliesslich ihren bisherigen Namen	350
3	2020	andere Namensführung (ausländisches Recht)	23
4	2019	unbekannt	13
5	2019	andere Namensführung (ausländisches Recht)	34
6	2019	nimmt den Namen des Mannes an	520
7	2019	führt ausschliesslich ihren bisherigen Namen	380

```
In [44]: # Plot female name over years
plt.figure(figsize = (15,8))
sns.lineplot(data=df_wname_year, x="year", y="number_marriage",
             hue="woman_name", linewidth = 2.5)
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)
plt.show()
```



From 1979 to 1996, there was no registration for different name choices. Consequently, all the cases at this time were unknown. Afterward, the plot shows a stable trend of circumstances where wives took their husbands' names until 2017, but this custom dropped from 2017 to 2020. In contrast, the number of women keeping their names gradually increased.

4.6 At which age do people tend to get married?

Cultures also play a vital role in the age at which people build a family. In some countries, people tend to establish a marital relationship much earlier. This section focuses on their ages.

```
In [45]: # Total cases by ages & gender
## multiple aggregation pipelines within a single stage on the same set of input documents
facet = {"$facet": {
    "woman": [
        {"$group": {"_id": "$woman.age_at_marriage",
            "number_marriage": {"$sum": "$quantity"}},
        {"$project": {"gender": "female",
            "number_marriage": "$number_marriage"
        }
    ],
    "man": [
        {"$group": {"_id": "$man.age_at_marriage",
            "number_marriage": {"$sum": "$quantity"}},
        {"$project": {"gender": "male",
            "number_marriage": "$number_marriage",
        }
    ]
}]

project1 = {"$project": {"both": {"$concatArrays": ["$woman", "$man"]}}}
unwind = {"$unwind": "$both"}

## replaces all existing fields in the input document, including the _id field
replaceroot = {"$replaceRoot": {"newRoot": "$both"}}
project2 = {"$project": {"age": "$_id",
    "_id": 0,
    "gender": "$gender",
    "number_marriage": "$number_marriage"}}
sort = {"$sort": {"number_marriage": -1}}

pipeline = [facet, project1, unwind, replaceroot, project2, sort]
```

```
c = marriage.aggregate(pipeline)

df_marriage_age_gender = pd.DataFrame(c)
df_marriage_age_gender.head(n=8)
```

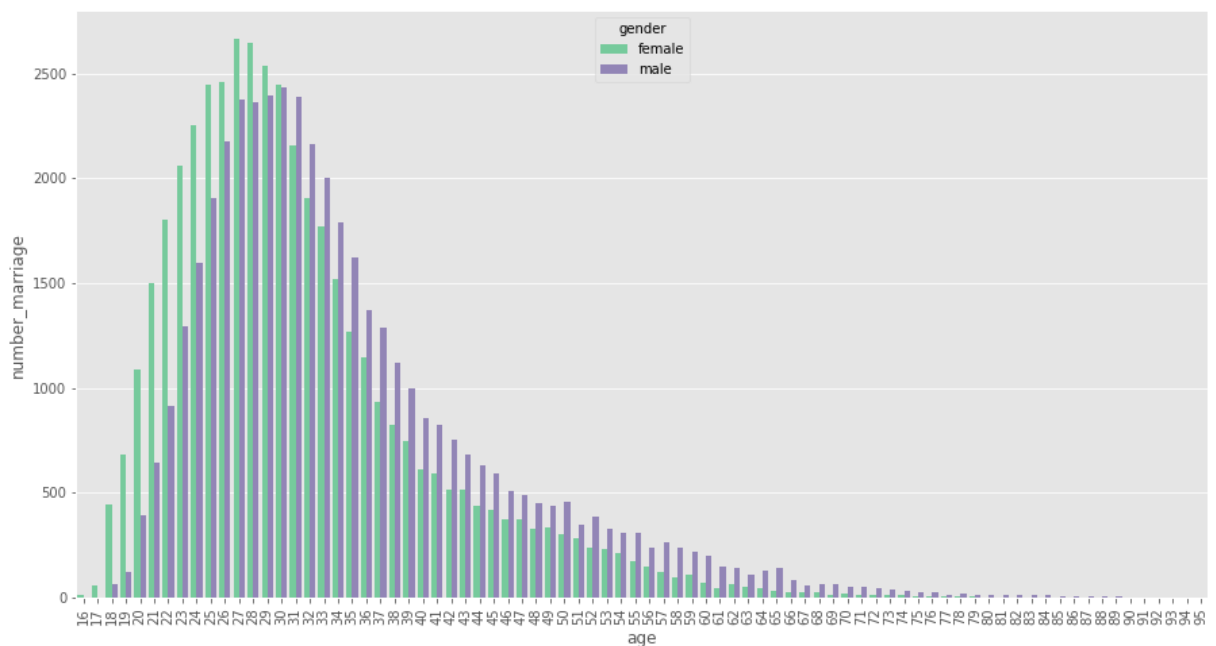
```
Out[45]:
```

	age	gender	number_marriage
0	27	female	2665
1	28	female	2643
2	29	female	2537
3	26	female	2460
4	25	female	2447
5	30	female	2446
6	30	male	2433
7	29	male	2395

From the table above, the most prevalent age to get married for women is 27 years old, while 30 years old for males.

```
In [46]: # Marriage bar chart by ages & genders
color = ['#68D89B', '#8E7DBE']

plt.figure(figsize = (15,8))
ax = sns.barplot(x="age", y="number_marriage",
                 hue="gender", palette=color,
                 data=df_marriage_age_gender)
locs, labels = plt.xticks()
plt.setp(labels, rotation=90)
plt.show()
```



The graph above gives a comprehensive view of both gender age. Female and male distributions have similar patterns, but the men distribution is located slightly more to the right. So far, women are likely to get married earlier than men as the highest number of cases are 24 to 32 years old. Men start having a family more commonly from 26 to 35 years old. Yet, remarkably, we have a few cases where women become wives at very young ages (16 to 18 years old). After investigating the overall age trend, we will examine whether the ages of marriage changed in different decades.

4.7 How has the marriage age changed over time?

First, let's examine gender distribution for age and married cases over four decades.

```
In [47]: # Marriage cases by age, gender and years
## multiple aggregation pipelines within a single stage on the same set of input documents
facet = {"$facet": {
    "woman": [
        {"$group": {"_id": {"age": "$woman.age_at_marriage", "year": "$time.year"},
            "number_marriage": {"$sum": "$quantity"}}},
        {"$project": {"gender": "female",
            "year": "$time.year",
            "number_marriage": "$number_marriage"
        }}
    ],
    "man": [
        {"$group": {"_id": {"age": "$man.age_at_marriage", "year": "$time.year"},
            "number_marriage": {"$sum": "$quantity"}}},
        {"$project": {"gender": "male",
            "year": "$time.year",
            "number_marriage": "$number_marriage"
        }}
    ]
}}

project1 = {"$project": {"both": {"$concatArrays": ["$woman", "$man"]}}}
unwind = {"$unwind": "$both"}

## replaces all existing fields in the input document, including the _id field
replaceroot = {"$replaceRoot": {"newRoot": "$both"}}
project2 = {"$project": {"age": "$_id.age",
    "year": "$_id.year",
    "_id": 0,
    "gender": "$gender",
    "number_marriage": "$number_marriage"
}}

sort = {"$sort": {"number_marriage": -1}}

pipeline = [facet, project1, unwind, replaceroot, project2, sort]
c = marriage.aggregate(pipeline)

df_marriage_age_gender_year = pd.DataFrame(c)
df_marriage_age_gender_year.head(n=8)
```

Out[47]:

	age	year	gender	number_marriage
0	25	1990	female	109
1	27	1989	female	106
2	23	1991	female	101
3	25	1989	female	101
4	24	1981	female	101
5	24	1982	female	99
6	28	1990	male	99
7	23	1980	female	97

```
In [48]: # Marriage cases by gender, age in different decades
fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(20,6))
# 1980s
sns.lineplot(data=df_marriage_age_gender_year[
    (df_marriage_age_gender_year["year"] >= "1980")
    & (df_marriage_age_gender_year["year"] <= "1989")],
    x="age", y="number_marriage", hue="gender", ax=ax[0])

# 1990s
```

```

sns.lineplot(data=df_marriage_age_gender_year[
    (df_marriage_age_gender_year["year"] >= "1990")
    & (df_marriage_age_gender_year["year"] <= "1999")],
    x="age", y="number_marriage", hue="gender", ax=ax[1])

# 2000s
sns.lineplot(data=df_marriage_age_gender_year[
    (df_marriage_age_gender_year["year"] >= "2000")
    & (df_marriage_age_gender_year["year"] <= "2009")],
    x="age", y="number_marriage", hue="gender", ax=ax[2])

# 2010s & 2020
sns.lineplot(data=df_marriage_age_gender_year[
    (df_marriage_age_gender_year["year"] >= "2010")
    & (df_marriage_age_gender_year["year"] <= "2020")],
    x="age", y="number_marriage", hue="gender", ax=ax[3])

ax[0].set_title("1980s")
ax[1].set_title("1990s")
ax[2].set_title("2000s")
ax[3].set_title("2010s & 2020")
locs, labels = plt.xticks()
plt.tight_layout()

```



The plot above exhibits a pretty similar pattern between decades. Nonetheless, there is an age shift in the four-period windows. To illustrate this point better, let's do another plot where we check the change of married cases with age over the years.

```

In [49]: # Marriage cases by age & years
## multiple aggregation pipelines within a single stage on the same set of input documents
facet = {"$facet": {
    "woman": [
        {"$group": {"_id": {"age": "$woman.age_at_marriage", "year": "$time.year"},
            "number_marriage": {"$sum": "$quantity"}}},
        {"$project": {"year": "$time.year",
            "number_marriage": "$number_marriage"
        }}
    ],
    "man": [
        {"$group": {"_id": {"age": "$man.age_at_marriage", "year": "$time.year"},
            "number_marriage": {"$sum": "$quantity"}}},
        {"$project": {"year": "$time.year",
            "number_marriage": "$number_marriage",
        }}
    ]
}}

project1 = {"$project": {"both": {"$concatArrays": ["$woman", "$man"]}}}
unwind = {"$unwind": "$both"}

## replaces all existing fields in the input document, including the _id field
replaceroot = {"$replaceRoot": {"newRoot": "$both"}}
project2 = {"$project": {"age": "$_id.age",
    "year": "$_id.year",
    "_id": 0,

```

```

        "number_marriage": "$number_marriage"
    }}
sort = {"$sort": {"number_marriage": -1}}

pipeline = [facet, project1, unwind, replaceroot, project2, sort]
c = marriage.aggregate(pipeline)

df_marriage_age_year = pd.DataFrame(c)
df_marriage_age_year.head()

```

Out[49]:

	age	year	number_marriage
0	25	1990	109
1	27	1989	106
2	25	1989	101
3	24	1981	101
4	23	1991	101

```

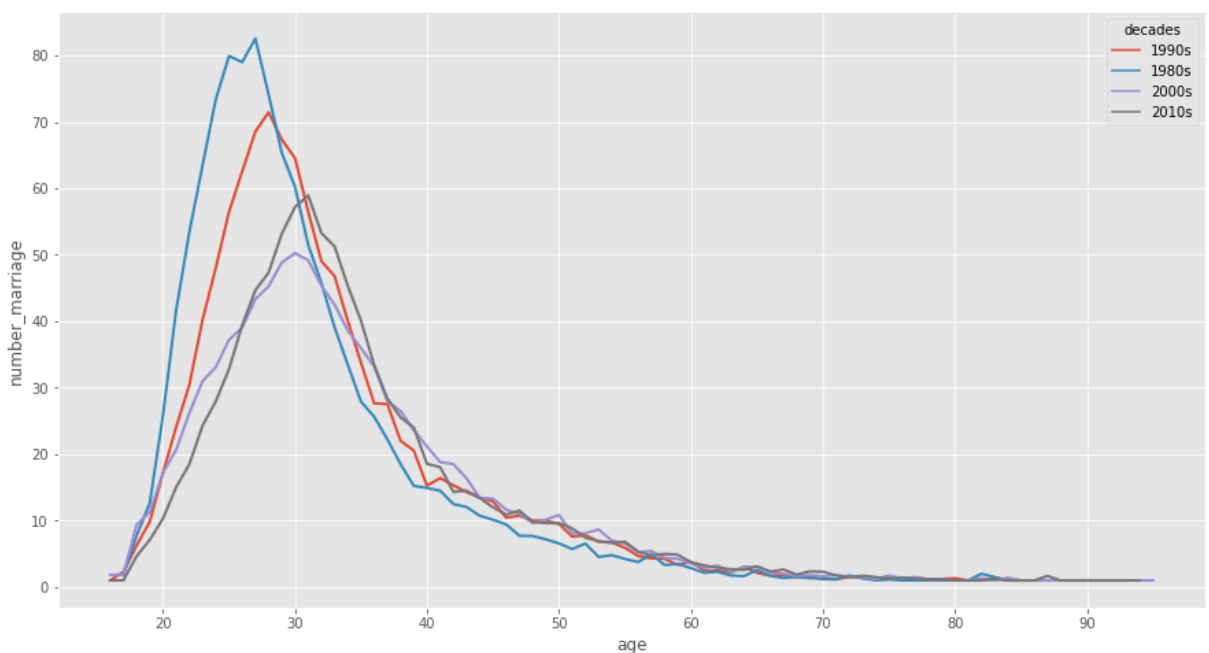
In [50]: # Line plot marriage by age & decades
# create a list of our conditions
conditions = [
    (df_marriage_age_year['year'] >= "1979") & \
    (df_marriage_age_year['year'] <= "1989"),
    (df_marriage_age_year['year'] >= "1990") & \
    (df_marriage_age_year['year'] <= "1999"),
    (df_marriage_age_year['year'] >= "2000") & \
    (df_marriage_age_year['year'] <= "2009"),
    (df_marriage_age_year['year'] >= "2010"),
]

# create a list of the values we want to assign for each condition
values = ['1980s', '1990s', '2000s', '2010s']

# create a new column & use np.select to assign values using lists as arguments
df_marriage_age_year['decades'] = np.select(conditions, values)

# Plot
plt.figure(figsize = (15,8))
sns.lineplot(data=df_marriage_age_year,
             x="age", y="number_marriage",
             hue="decades", ci=None,
             linewidth = 2)
plt.show()

```



Noticeably, this plot captures differences in age of marriage more prominent than in the previous graph. First, more younger people joined a legal relationship in the 1980s. With the time, people have a wedding at older ages as these lines move to the right. We witness a sharp behavior transformation over different decades. For example, the grey line with the 2000-2010 period indicates that couples marry mostly at their 30 years old. For the same decade, there are more cases of marriages from 30 to 50 years old compared to the same age range in other decades.

4.8 How is the age distribution of divorced people?

After scrutinizing the ages of married people, the next step is to compare with splitted couples.

```
In [51]: # Total divorce cases by ages & gender
## multiple aggregation pipelines within a single stage on the same set of input documents
facet = {"$facet": {
    "woman": [
        {"$match": {"woman.age": {"$gt": 0}}},
        {"$group": {"_id": "$woman.age",
                    "number_divorce": {"$sum": "$quantity"}}},
        {"$project": {"gender": "female",
                    "number_divorce": "$number_divorce"}
        }
    ],
    "man": [
        {"$match": {"man.age": {"$gt": 0}}},
        {"$group": {"_id": "$man.age",
                    "number_divorce": {"$sum": "$quantity"}}},
        {"$project": {"gender": "male",
                    "number_divorce": "$number_divorce",
                    }
        }
    ]
}}

project1 = {"$project": {"both": {"$concatArrays": ["$woman", "$man"]}}}
unwind = {"$unwind": "$both"}

## replaces all existing fields in the input document, including the _id field
replaceroor = {"$replaceRoot": {"newRoot": "$both"}}
project2 = {"$project": {"age": "$_id",
                        "_id": 0,
                        "gender": "$gender",
                        "number_divorce": "$number_divorce"}}
sort = {"$sort": {"number_divorce": -1}}

pipeline = [facet, project1, unwind, replaceroor, project2, sort]
c = divorce.aggregate(pipeline)

df_divorce_age_gender = pd.DataFrame(c)
df_divorce_age_gender.head()
```

```
Out[51]:
```

	age	gender	number_divorce
0	37	female	652
1	35	female	644
2	36	male	638
3	34	female	624
4	36	female	624

According to the data frame above, female broke up mostly from 34 to 37 years old.

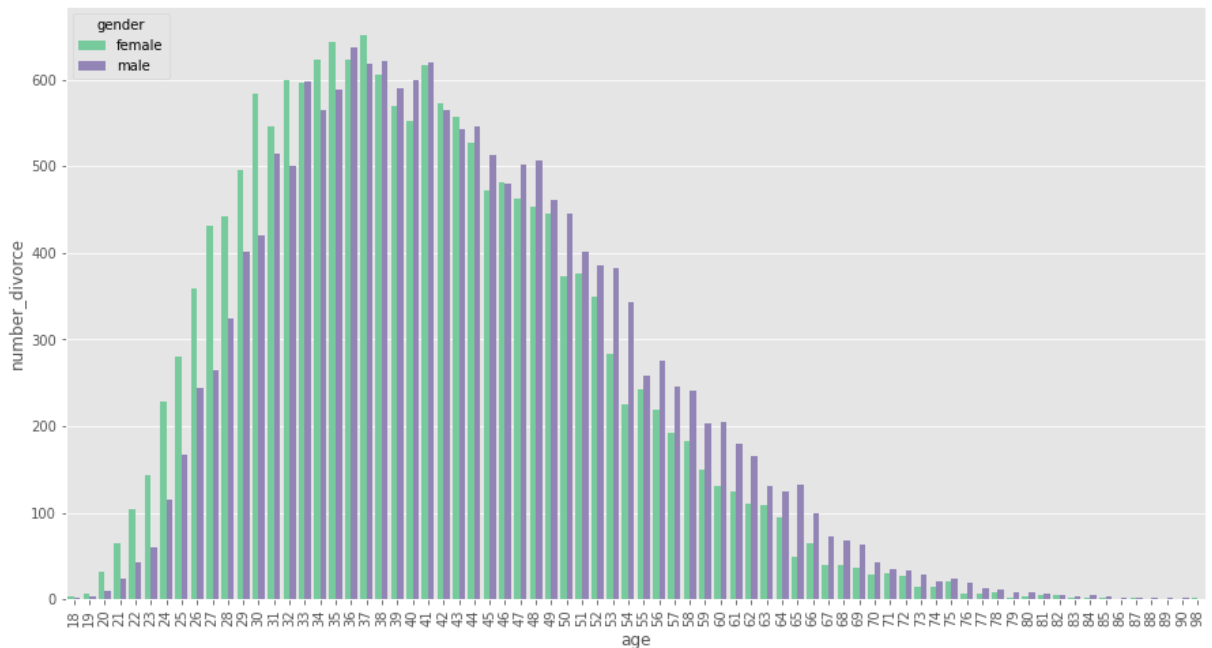
```
In [52]: # Marriage bar chart by ages & genders
plt.figure(figsize = (15,8))
```



```

color = ['#68D89B', "#8E7DBE"]
ax = sns.barplot(x="age", y="number_divorce",
                 hue="gender", palette=color,
                 data=df_divorce_age_gender)
locs, labels = plt.xticks()
plt.setp(labels, rotation=90)
plt.show()

```



Recall that the marriage age distribution has a thin form, a long tail and most of observations locate in the beginning. This divorce age distribution looks wider and somewhat different. Compared to the peak ages in marriage, the divorce distribution concentration focused in the older age range, especially for men.

4.9 How are number of occurrences concerning the nationality combinations?

This part concentrates on the citizenship of couples and scrutinizes multiple other factors in the interaction with citizenship. In the **marriage** collection, the two distinct values of nationality are *foreign* ('Ausl') and *Swiss* ('CH'). It would be even more compelling if each person's original homeland were given in this data for analysis.

```

In [53]: print(marriage.distinct("woman.citizenship"))
         print(marriage.distinct("man.citizenship"))

```

```

['Ausl', 'CH']
['Ausl', 'CH']

```

```

In [54]: # Marriage cases by citizenship in general
c = marriage.aggregate([
    {"$group" : {
        "_id": {"woman_citizenship": "$woman.citizenship",
               "man_citizenship": "$man.citizenship"},
        "number_marriage": {"$sum": "$quantity"}}
    },
    {"$project": {
        "_id": 0,
        "woman_citizenship": "$_id.woman_citizenship",
        "man_citizenship": "$_id.man_citizenship",
        "wife_and_husband":
            {"$concat": ["$_id.woman_citizenship", "&", "$_id.man_citizenship"]},
        "number_marriage": "$number_marriage",
        "percent": {"$multiply": [{"$divide": ["$number_marriage", total]}, 100]}
    }},

```

```

    {"$sort": {"number_marriage": -1}}
])

df_citizenship = pd.DataFrame(c)
df_citizenship

```

Out[54]:

	woman_citizenship	man_citizenship	wife_&_husband	number_marriage	percent
0	CH	CH	CH_&_CH	20957	47.29
1	Ausl	CH	Ausl_&_CH	10203	23.02
2	Ausl	Ausl	Ausl_&_Ausl	6772	15.28
3	CH	Ausl	CH_&_Ausl	6382	14.40

Out of 44,314 occurrences, nearly **half** are marriages between **Swiss spouses**. The **runner-up** is a union of **a foreign wife** with a **Swiss husband**. The wedding occasions between two foreigners or a Swiss wife with a foreign husband are almost the same at around 15%. Markedly, the situations in which a Swiss groom marries an immigrant are more common than a Swiss bride with an outlander. Next, it's vital to compare with the divorces:

In [55]:

```

# Divorce cases by citizenship in general
c = divorce.aggregate([
    {"$match": {"time.year": {"$gt": "1983"}}},
    {"$group" : {
        "_id": {"woman_citizenship": "$woman.citizenship",
                "man_citizenship": "$man.citizenship"},
        "number_divorce": {"$sum": "$quantity"}}
    },
    {"$project": {
        "_id": 0,
        "woman_citizenship": "$_id.woman_citizenship",
        "man_citizenship": "$_id.man_citizenship",
        "wife_&_husband":
            {"$concat": ["$_id.woman_citizenship", "_&", "$_id.man_citizenship"]},
        "number_divorce": "$number_divorce"
    }},
    {"$sort": {"number_divorce": -1}}
])

df_divorce_citizenship = pd.DataFrame(c)
df_divorce_citizenship['percent'] = (df_divorce_citizenship['number_divorce'] /
                                     df_divorce_citizenship['number_divorce'].sum()) * 100
df_divorce_citizenship

```

Out[55]:

	woman_citizenship	man_citizenship	wife_&_husband	number_divorce	percent
0	CH	CH	CH_&_CH	9740	56.13
1	CH	Ausl	CH_&_Ausl	3526	20.32
2	Ausl	Ausl	Ausl_&_Ausl	2561	14.76
3	Ausl	CH	Ausl_&_CH	1525	8.79

For divorces, the **Swiss couples** made up **56% of the occurrences**. Next, the pairs of Swiss spouses and expatriate partners are the second-highest segment with 20%. The lowest ratio of divorces is the combination between Swiss husband and foreign wife.

4.10 How have the citizenship combinations of married couples changed over time?

In [56]:

```

# Marriage cases by citizenship & year
c = marriage.aggregate([
    {"$group" : {

```

```

        "_id": {"woman_citizenship": "$woman.citizenship",
                "man_citizenship": "$man.citizenship",
                "year": "$time.year"},
        "number_marriage": {"$sum": "$quantity"}}
    },
    {"$project": {
        "_id": 0,
        "year": "$_id.year",
        "wife_&_husband":
            {"$concat": ["$_id.woman_citizenship", "_&", "$_id.man_citizenship"]},
        "number_marriage": "$number_marriage"
    }},
    {"$sort": {"year": 1}}
])

df_citizenship_year = pd.DataFrame(c)
df_citizenship_year.head()

```

Out[56]:

	year	wife_&_husband	number_marriage
0	1979	CH_&_CH	717
1	1979	CH_&_Ausl	73
2	1979	Ausl_&_Ausl	85
3	1979	Ausl_&_CH	165
4	1980	Ausl_&_CH	187

In [57]:

```

# Stacked area plot for marriage cases in terms of citizenship & year
# pivot the table to have the wide format for area chart
df_citizenship_year_pivot = pd.pivot_table(df_citizenship_year,
                                            values='number_marriage',
                                            index=["year"],
                                            columns=["wife_&_husband"],
                                            aggfunc=np.sum)

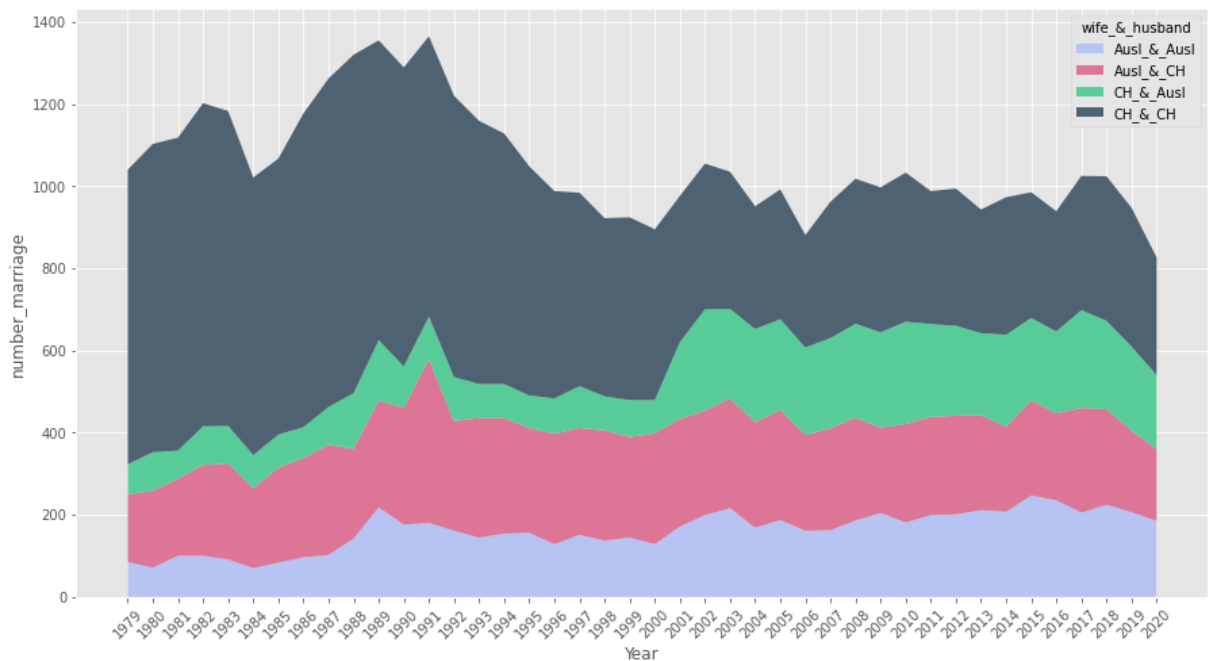
# color palette
colors = ['#B7C3F3', '#DD7596', '#57cc99', '#4F6272']

plt.figure(figsize = (15,8))
# create area chart
plt.stackplot(df_citizenship_year_pivot.index,
              df_citizenship_year_pivot["Ausl_&_Ausl"],
              df_citizenship_year_pivot["Ausl_&_CH"],
              df_citizenship_year_pivot["CH_&_Ausl"],
              df_citizenship_year_pivot["CH_&_CH"],
              labels=['Ausl_&_Ausl', 'Ausl_&_CH', 'CH_&_Ausl', 'CH_&_CH'],
              colors=colors)

#add Legend
plt.legend(loc='upper right', title = "wife_&_husband")
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)

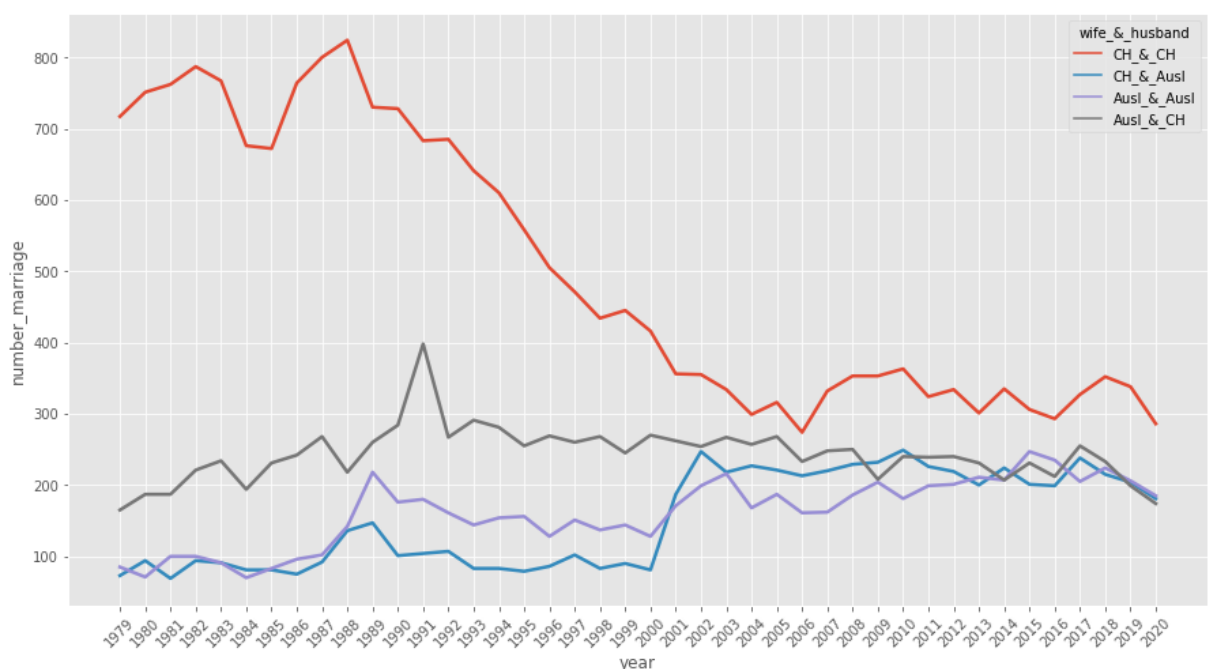
#add axis labels
plt.xlabel('Year')
plt.ylabel('number_marriage')
plt.show()

```



It's essential to check the total value and understand the breakdown of that total by groups. Hence, we will look at the stacked area above. Comparing the heights of each category enables us to get how each of them contributes to the total cases. Note that each stacked area is presented with different colors, plotted once at a time with the height of the most recently-plotted group performing as a moving baseline. Thus, each area does not start from 0. The area of "CH_&_CH" become smaller, even though it is the most extensive area of all. The green space got larger after 2000, meaning that Swiss women espoused increasingly with outlanders.

```
In [58]: # Line chart for marriage cases in terms of citizenship & year
plt.figure(figsize = (15,8))
sns.lineplot(data=df_citizenship_year, x="year",
             y="number_marriage",
             hue="wife_&_husband", ci=None,
             linewidth = 2.5)
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)
plt.show()
```



The line chart gives an overview of couples' pairing nationality over the time. In general, the number of both spouses are Swiss dropped dramatically after 1988. The combinations of Swiss wife and foreign

husband or two expatriates increased slightly. In 1991, many Swiss men married with immigrants. Overall, the flow of Swiss husband with foreign wife remained quite stable. The next part analyzes further citizenship and ages of married couples.

4.11 How have the citizenship combinations of divorced pairs changed over time?

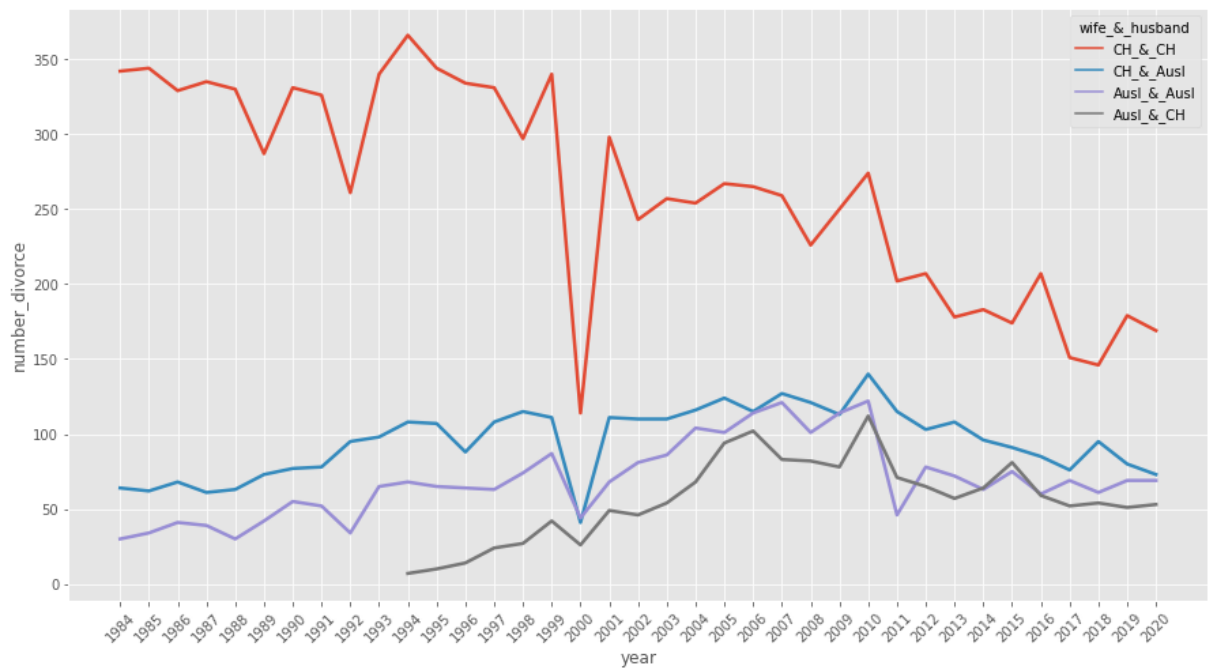
```
In [59]: # Divorce cases by citizenship & year
c = divorce.aggregate([
    {"$match": {"woman.citizenship": {"$ne": "unbekannt"}}},
    {"$group" : {
        "_id": {"woman_citizenship": "$woman.citizenship",
            "man_citizenship": "$man.citizenship",
            "year": "$time.year"},
        "number_divorce": {"$sum": "$quantity"}}
    },
    {"$project": {
        "_id": 0,
        "year": "$_id.year",
        "wife_&_husband":
            {"$concat": ["$_id.woman_citizenship", "_&_", "$_id.man_citizenship"]},
        "number_divorce": "$number_divorce"
    }},
    {"$sort": {"year": 1}}
])

df_divorce_citizenship_year = pd.DataFrame(c)
df_divorce_citizenship_year.head()
```

```
Out[59]:
```

	year	wife_&_husband	number_divorce
0	1984	CH_&_CH	342
1	1984	CH_&_Ausl	64
2	1984	Ausl_&_Ausl	30
3	1985	Ausl_&_Ausl	34
4	1985	CH_&_CH	344

```
In [60]: # Line chart for marriage cases in terms of citizenship & year
plt.figure(figsize = (15,8))
sns.lineplot(data=df_divorce_citizenship_year, x="year",
             y="number_divorce",
             hue="wife_&_husband", ci=None,
             linewidth = 2.5)
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)
plt.show()
```



In 2000, all the divorced occurrences dropped.

4.12 How are the age differences from pair of spouses?

```
In [61]: # Marriage cases by citizenship & differences in age
c = marriage.aggregate([
    {"$group": {
        "_id": {"woman_citizenship": "$woman.citizenship",
            "man_citizenship": "$man.citizenship",
            "woman_age": "$woman.age_at_marriage",
            "man_age": "$man.age_at_marriage"},
        "number_marriage": {"$sum": "$quantity"}}
    },
    {"$project": {
        "_id": 0,
        "man_citizenship": "$_id.man_citizenship",
        "man_age": "$_id.man_age",
        "woman_citizenship": "$_id.woman_citizenship",
        "woman_age": "$_id.woman_age",
        "man_subtract_woman_age":
            {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
        "number_marriage": "$number_marriage",
        "husband_&_wife":
            {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
    }},
    {"$sort": {"man_subtract_woman_age": 1}}
])

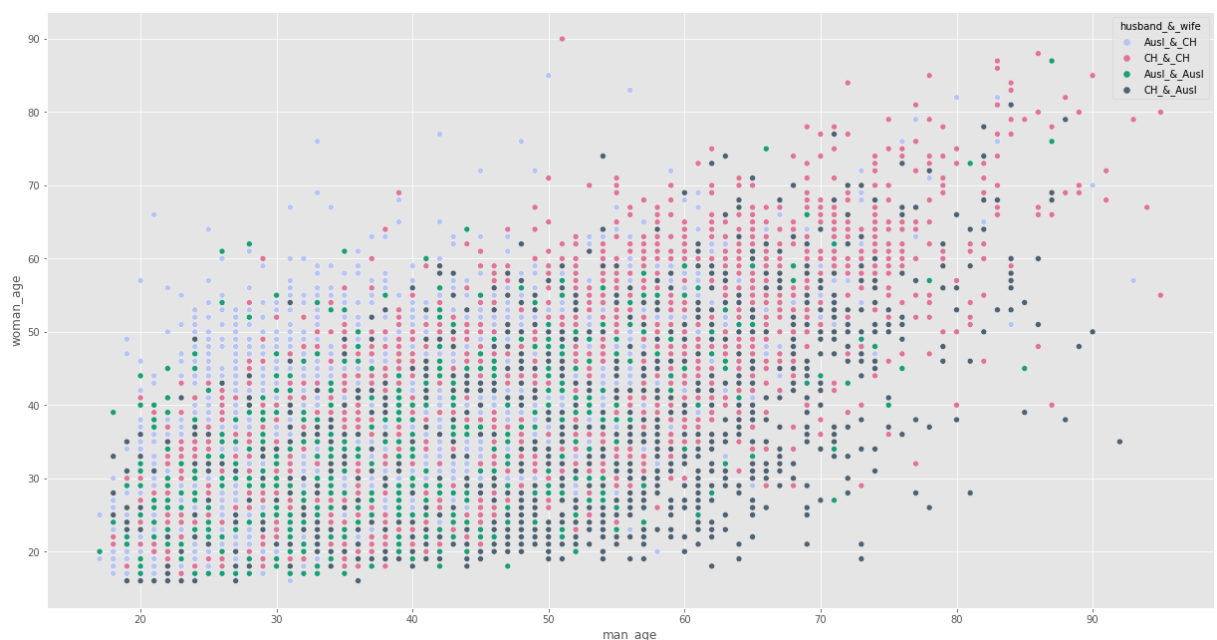
df_citizenship_age_diff = pd.DataFrame(c)
df_citizenship_age_diff.head(n=8)
```

```
Out[61]:
```

	man_citizenship	man_age	woman_citizenship	woman_age	man_subtract_woman_age	number_marriage	hu
0	Ausl	21	CH	66	-45	1	
1	Ausl	33	CH	76	-43	1	
2	Ausl	25	CH	64	-39	1	
3	CH	51	CH	90	-39	1	
4	Ausl	20	CH	57	-37	1	
5	Ausl	31	CH	67	-36	1	
6	Ausl	33	CH	69	-36	1	
7	Ausl	42	CH	77	-35	1	

To have a more clear picture, I created a new variable called "man_subtract_woman_age". The table above shows many interesting cases where there are huge differences between ages of wife and husband. Let's visualize to detect interesting patterns.

```
In [64]: # Scatter plot marriage cases by ages
plt.figure(figsize = (21,11))
# color palette
colors = ['#B7C3F3', '#DD7596', '#1b9e77', '#4F6272']
sns.scatterplot(data=df_citizenship_age_diff,
                x="man_age",
                y="woman_age",
                hue="husband_&_wife",
                palette=colors)
plt.show()
```



The scatter plot displays a generally increasing trend. The older the husbands are, the older his wife also is. Moreover, some noticeable patterns can be identified in this graph. First, looking at the light purple points on the middle-left, many Swiss females were married at their older ages to younger expatriates.

In contrast, many dark blue points stay on the bottom middle. These are the cases when Swiss husbands aged 50-70 years old were with foreign wives from 19 to 30 years old. More green points concentrate on the left, where immigrants marry at their younger ages, from 20 to 55. Meanwhile, some uncommon cases are outstanding such as more aged than 90 years old couples. In general, Swiss pairs commonly build a family at higher age according to many pink points on the right upper side. Notably, one pink point on the top middle shows a Swiss couple, a 90-year-old wife, with her 51-year-old husband.

4.13 Which nationality and age combinations of spouses are popular?

In this section, I took the query from the previous question and drew a direct network with the combination of nationality and age for each spouse. Notice that only instances that happened more than 50 times are taken into this network to eliminate too crowded presentation. In total, there are 23 nodes as actors and 114 links as their connections in the network.

```
In [65]: # Create a new dataframe to draw network
df_new = df_citizenship_age_diff
df_new["man"] = df_citizenship_age_diff["man_citizenship"] + \
"_" + df_citizenship_age_diff["man_age"].astype(str)
df_new["woman"] = df_citizenship_age_diff["woman_citizenship"] + \
"_" + df_citizenship_age_diff["woman_age"].astype(str)
df_new = df_new[["man", "woman", "number_marriage"]]

# Filter the observations with > 50 times
df_new = df_new[df_new["number_marriage"] > 50]
df_new
```

```
Out[65]:
```

	man	woman	number_marriage
1583	CH_30	CH_32	55
1584	CH_28	CH_30	73
1612	CH_29	CH_31	64
1622	CH_26	CH_28	54
1623	CH_25	CH_27	55
...
3069	CH_34	CH_28	74
3081	CH_28	CH_22	59
3084	CH_30	CH_24	68
3210	CH_32	CH_25	55
3250	CH_30	CH_23	54

114 rows × 3 columns

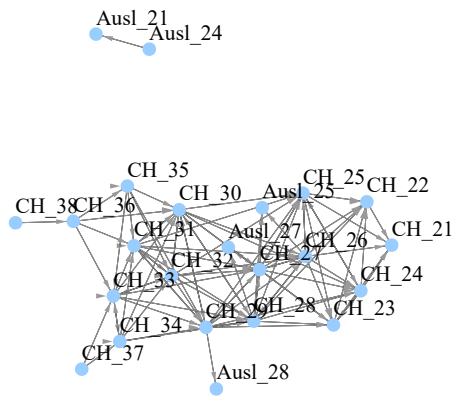
```
In [66]: # Direct network with pathpy
t = pp.Network(directed=True)
df_new.apply(lambda x: t.add_edge(x["man"],
                                   x["woman"],
                                   weight=x["number_marriage"]), axis=1)

print(t)
```

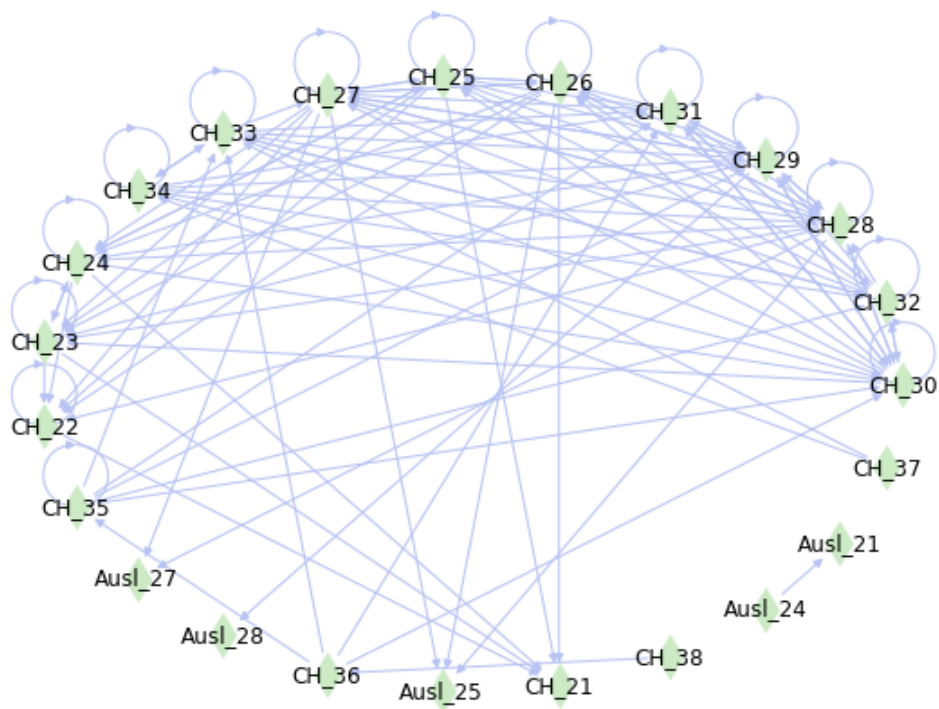
```
Directed network
Nodes:          23
Links:         114
```

The picture below shows a direct network from popular cases. There are only 4 times we detect foreigner nodes here. Let's visualize this network in a circle to read better.

[save svg]



```
In [67]: # Draw circular network of pairs with > 50 occurrences
G = nx.Graph()
G = nx.from_pandas_edgelist(df_new, 'man', 'woman',
                           edge_attr='number_marriage',
                           create_using=nx.DiGraph)
plt.figure(figsize = (8,6))
nx.draw_circular(G, with_labels=True, node_color="#cceb5",
                 edge_color= "#B7C3F3", node_shape='d')
```



The graph above displays a circular network of pairs for all the cases occurring more than 50 times. In many situations, the nodes connecting to themselves present the same age and nationality duos, especially Swiss people from 22 to 34 years. All of the most typical situations are pairs of small age gaps. In other words, their age differences are mostly less than 10 years old. We also see that some ages receive more links, like 28 to 31 years old.

4.14 What are the biggest age gaps in marriage for each nationality pairing?

To answer this question, it's crucial to find cases with the largest and smallest age difference. The age gap

is calculated as the male age subtracting the female age within a pair. Hence, the most negative age gaps depict occasions when a wife is much older than her husband. Consequently, the age gap is examined in both genders.

```
In [68]: # Top biggest and smallest difference in age for each
c = marriage.aggregate([
    {"$group": {
        "_id": {"woman_citizenship": "$woman.citizenship",
            "man_citizenship": "$man.citizenship",
            "woman_age": "$woman.age_at_marriage",
            "man_age": "$man.age_at_marriage"},
        "number_marriage": {"$sum": "$quantity"}}
    },
    {"$project": {
        "_id": 0,
        "man_citizenship": "$_id.man_citizenship",
        "man_age": "$_id.man_age",
        "woman_citizenship": "$_id.woman_citizenship",
        "woman_age": "$_id.woman_age",
        "man_subtract_woman_age":
            {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
        "number_marriage": "$number_marriage",
        "husband_&_wife":
            {"$concat": ["$_id.man_citizenship", "_&", "$_id.woman_citizenship"]}
    }},
    {"$group": {
        "_id": "$husband_&_wife",
        "biggest_diff": {"$max": "$man_subtract_woman_age"},
        "smallest_diff": {"$min": "$man_subtract_woman_age"},
    }},
    {"$project": {
        "_id": 0,
        "husband_&_wife": "$_id",
        "biggest_diff": "$biggest_diff",
        "smallest_diff": "$smallest_diff",
        "man_age": "$man_age"
    }}
])

df_citizenship_age_top = pd.DataFrame(c)
df_citizenship_age_top
```

```
Out[68]:
```

	husband_&_wife	biggest_diff	smallest_diff
0	AusI_&_AusI	44	-35
1	CH_&_CH	47	-39
2	CH_&_AusI	57	-25
3	AusI_&_CH	41	-45

The table above portrays only extreme records regarding four types of nationality combo. The `'_biggestdiff'` stands for husbands much older than his wife, and the `'_smallestdiff'` column refers to brides much more vintage than her companion. The most significant age difference is 57 years old for all the observations. In this case, the Swiss husband is 57 years older than his immigrant lover. On the other hand, minus 45, the most negative number, shows the foreign husband is 45 years younger than his Swiss wife.

4.15 Who are the top five age gaps in marriage and divorce?

```
In [69]: # Top 5 age gaps of husband_wife
c = marriage.aggregate([
    {"$group": {
        "_id": {"woman_citizenship": "$woman.citizenship",
            "man_citizenship": "$man.citizenship",
```

```

        "woman_age": "$woman.age_at_marriage",
        "man_age": "$man.age_at_marriage"},
    "number_marriage":{"$sum":"$quantity"},
  }},
  {"$project": {
    "_id": 0,
    "man_subtract_woman_age":
    {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
    "man_age": "$_id.man_age",
    "woman_age": "$_id.woman_age",
    "number_marriage": "$number_marriage",
    "husband_&_wife":
    {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
  }},
  {"$sort": {"man_subtract_woman_age": -1}},
  {"$limit": 5},
])

df_citizenship_age_top1 = pd.DataFrame(c)
df_citizenship_age_top1

```

Out[69]:

	man_subtract_woman_age	man_age	woman_age	number_marriage	husband_&_wife
0	57	92	35	1	CH_&_Ausl
1	54	73	19	1	CH_&_Ausl
2	53	81	28	1	CH_&_Ausl
3	52	78	26	1	CH_&_Ausl
4	52	73	21	1	CH_&_Ausl

The top five most prominent age disparities belong to the pair group with Swiss husbands and foreign wives. While these men are from 73 to 92, their immigrant women are pretty young, with the youngest of 19 years old. Next, let's look at the top five highest age gap in divorce:

In [70]:

```

# Top 5 age gaps of divorced pairs
c = divorce.aggregate([
  {"$group": {
    "_id": {"woman_citizenship":"$woman.citizenship",
            "man_citizenship":"$man.citizenship",
            "woman_age": "$woman.age",
            "man_age": "$man.age"},
    "number_divorce":{"$sum":"$quantity"},
  }},
  {"$project": {
    "_id": 0,
    "man_subtract_woman_age":
    {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
    "man_age": "$_id.man_age",
    "woman_age": "$_id.woman_age",
    "number_divorce": "$number_divorce",
    "husband_&_wife":
    {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
  }},
  {"$sort": {"man_subtract_woman_age": -1}},
  {"$limit": 5},
])

df_divorce_citizenship_age_top5 = pd.DataFrame(c)
df_divorce_citizenship_age_top5

```

Out[70]:

	man_subtract_woman_age	man_age	woman_age	number_divorce	husband_&_wife
0	54	78	24	1	CH_&Ausl
1	48	90	42	1	CH_&CH
2	45	79	34	1	CH_&CH
3	44	69	25	1	CH_&CH
4	42	75	33	1	CH_&CH

Comparing the top five age gaps between marriage and divorce, the males seem to be younger in the divorce data. While married couples are all Swiss husbands and foreign wives, four pairs in divorce are Swiss people.

4.16 Who are the top five oldest husbands and wives in marriage cases?

```
In [71]: # Top 5 oldest husbands
c = marriage.aggregate([
    {"$group" : {
        "_id": {"woman_citizenship": "$woman.citizenship",
                "man_citizenship": "$man.citizenship",
                "woman_age": "$woman.age_at_marriage",
                "man_age": "$man.age_at_marriage",
                "number_marriage": {"$sum": "$quantity"}},
    }},
    {"$project": {
        "_id": 0,
        "man_age": "$_id.man_age",
        "woman_age": "$_id.woman_age",
        "number_marriage": "$number_marriage",
        "man_subtract_woman_age":
            {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
        "husband_&_wife":
            {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
    }},
    {"$sort": {"man_age": -1}},
    {"$limit": 5},
])

df_citizenship_age_oldest_m = pd.DataFrame(c)
df_citizenship_age_oldest_m
```

Out[71]:

	man_age	woman_age	number_marriage	man_subtract_woman_age	husband_&_wife
0	95	55	1	40	CH_&CH
1	95	80	1	15	CH_&CH
2	94	67	1	27	CH_&CH
3	93	79	1	14	CH_&CH
4	93	57	1	36	Ausl_&CH

The top five oldest male spouse entered to marital relationship at later than 90 years old and they mostly come from Switzerland. Let's check top five oldest wives below:

```
In [72]: # Top 5 oldest wives
c = marriage.aggregate([
    {"$group" : {
        "_id": {"woman_citizenship": "$woman.citizenship",
                "man_citizenship": "$man.citizenship",
                "woman_age": "$woman.age_at_marriage",
                "man_age": "$man.age_at_marriage",
                "number_marriage": {"$sum": "$quantity"}},
    }},
    {"$project": {
        "_id": 0,
        "man_age": "$_id.man_age",
        "woman_age": "$_id.woman_age",
        "number_marriage": "$number_marriage",
        "man_subtract_woman_age":
            {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
        "husband_&_wife":
            {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
    }},
    {"$sort": {"woman_age": -1}},
    {"$limit": 5},
])

df_citizenship_age_oldest_w = pd.DataFrame(c)
df_citizenship_age_oldest_w
```

```

    }},
    {"$project": {
      "_id": 0,
      "woman_age": "$_id.woman_age",
      "man_age": "$_id.man_age",
      "number_marriage": "$number_marriage",
      "man_subtract_woman_age":
        {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
      "husband_&_wife":
        {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
    }},
    {"$sort": {"woman_age": -1}},
    {"$limit": 5},
  ])

df_citizenship_age_oldest_w = pd.DataFrame(c)
df_citizenship_age_oldest_w

```

Out[72]:

	woman_age	man_age	number_marriage	man_subtract_woman_age	husband_&_wife
0	90	51	1	-39	CH_&_CH
1	88	86	1	-2	CH_&_CH
2	87	87	1	0	Ausl_&_Ausl
3	87	83	1	-4	CH_&_CH
4	86	83	1	-3	CH_&_CH

4.17 Who are the top five youngest husbands and wives in marriage cases?

In [73]:

```

# Top 5 youngest husbands
c = marriage.aggregate([
  {"$group": {
    "_id": {"woman_citizenship": "$woman.citizenship",
            "man_citizenship": "$man.citizenship",
            "woman_age": "$woman.age_at_marriage",
            "man_age": "$man.age_at_marriage"},
    "number_marriage": {"$sum": "$quantity"},
  }},
  {"$project": {
    "_id": 0,
    "man_age": "$_id.man_age",
    "woman_age": "$_id.woman_age",
    "number_marriage": "$number_marriage",
    "man_subtract_woman_age":
      {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
    "husband_&_wife":
      {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
  }},
  {"$sort": {"man_age": 1}},
  {"$limit": 5},
])

df_citizenship_age_youngest_m = pd.DataFrame(c)
df_citizenship_age_youngest_m

```

Out[73]:

	man_age	woman_age	number_marriage	man_subtract_woman_age	husband_&_wife
0	17	25	1	-8	Ausl_&_CH
1	17	20	1	-3	Ausl_&_Ausl
2	17	19	1	-2	Ausl_&_CH
3	18	20	1	-2	CH_&_Ausl
4	18	21	2	-3	Ausl_&_CH

```
In [ ]: # Top 5 youngest wives
c = marriage.aggregate([
    {"$group": {
        "_id": {"woman_citizenship": "$woman.citizenship",
            "man_citizenship": "$man.citizenship",
            "woman_age": "$woman.age_at_marriage",
            "man_age": "$man.age_at_marriage"},
        "number_marriage": {"$sum": "$quantity"},
    }},
    {"$project": {
        "_id": 0,
        "woman_age": "$_id.woman_age",
        "man_age": "$_id.man_age",
        "number_marriage": "$number_marriage",
        "man_subtract_woman_age":
            {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
        "husband_&_wife":
            {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
    }},
    {"$sort": {"woman_age": 1}},
    {"$limit": 5},
])

df_citizenship_age_youngest_w = pd.DataFrame(c)
df_citizenship_age_youngest_w
```

4.18 How do the most common divorce situations look, regarding nationality and age combinations?

```
In [75]: # Divorce cases by citizenship & differences in age
c = divorce.aggregate([
    {"$match": {"woman.age": {"$ne": 0}}},
    {"$group": {
        "_id": {"woman_citizenship": "$woman.citizenship",
            "man_citizenship": "$man.citizenship",
            "woman_age": "$woman.age",
            "man_age": "$man.age"},
        "number_divorce": {"$sum": "$quantity"}
    }},
    {"$project": {
        "_id": 0,
        "number_divorce": "$number_divorce",
        # "man_citizenship": "$_id.man_citizenship",
        "man_age": "$_id.man_age",
        # "woman_citizenship": "$_id.woman_citizenship",
        "woman_age": "$_id.woman_age",
        "man_subtract_woman_age":
            {"$subtract": ["$_id.man_age", "$_id.woman_age"]},
        "husband_&_wife":
            {"$concat": ["$_id.man_citizenship", "&", "$_id.woman_citizenship"]}
    }},
    {"$sort": {"number_divorce": -1}}
])

df_divorce_citizenship_age_diff = pd.DataFrame(c)
df_divorce_citizenship_age_diff.head(n=10)
```

Out[75]:

	number_divorce	man_age	woman_age	man_subtract_woman_age	husband_&_wife
0	47	36	35	1	CH_&_CH
1	47	37	36	1	CH_&_CH
2	45	41	39	2	CH_&_CH
3	42	41	41	0	CH_&_CH
4	39	41	40	1	CH_&_CH
5	39	38	38	0	CH_&_CH
6	38	46	43	3	CH_&_CH
7	38	40	37	3	CH_&_CH
8	37	37	37	0	CH_&_CH
9	37	35	32	3	CH_&_CH

The table above exhibits the most separated situations by sorting the number of divorces and grouping age and citizenship. All of the top 10 are Swiss people in their late 30s to their mid-40s.

5 Conclusions

The research fetched JSON data from canton Basel-Stadt, transformed the flat data structure to be nested, then loaded it into MongoDB. Once the ETL process had finished, the data was queried through many aggregation pipelines to answer numerous questions pivoting on marriage and divorce. Several data visualization were created to illustrate and explain better. Although there are no demographic or characteristic data about spouses to further analyze, the project serves as a good starting point for future research.

The project discovers myriad fascinating insights from data about marriage and divorce in Switzerland. To sum up, people tend to get married at older ages from decades to decades. Also, the number of marriage circumstances had reduced while divorces remained relatively stable over the years. Warm and hot weather months are trendy for organizing a wedding, and in contrast, January witnesses many split-ups. Moreover, Friday is a favorite marital day, whereas most Tuesday is likely chosen by broken-up couples. Additionally, the number of brides keeping their maiden names after a marriage has increased. The most common marriage age is from 25 to 30, while the most frequent divorce age is 34 to 37. There are slightly different between males and females in terms of age, such as men marrying at an elder age. The scatter plot of married couples' generation with the color of their nationality combination display some exciting patterns from different pair groups.

The top prominent age gaps in marriage belong to Swiss men with immigrant wives or Swiss women with foreign husbands. The five youngest bridges and grooms are just 16 to 18 years old. In contrast, the five oldest grooms are from 93 to 95. The most enormous age gap for marriage and divorce is 57 and 55 years. Overall, half of marriages and divorce occurrences are occupied by Swiss spouses. Remarkably, the combination of a Swiss wife with an immigrant husband slightly increases over time, generating the lowest marriage ratio but filling the second-highest proportion in divorce. Reversely, the pairs of a Swiss man with a foreign wife are the second most popular in marriage and the least common case in total divorces.

6 Remarks and learnings

All in all, doing the individual project enables me to practice the ETL process. I had never worked so much with Python before (only two projects) because most of my previous courses were in R (due to the requirements of these subjects). Hence, I could learn about combining Python and MongoDB from this

project. Likewise, I had a chance to practice writing Python codes and visualizing data with Python for the first time. Again, it facilitates me to compare data visualization in ggplot, Tableau with seaborn, and matplotlib. One thing that surprised me was the challenge of exporting a PDF file from jupyter notebook. I did some R-markdown projects before, and the PDF document looks nice. The lecturer suggested three different ways to achieve a PDF version, and each of them has advantages and disadvantages. I did two different ways to compare:

- Print screen from the actual Jupyter notebook: Heading numbers are included, but sometimes there is space under the graphs & tables
- File -> Save as HTML file -> print screen from the browser of that HTML: There is no space under graph and table but also no automatic heading numbers

Also, as I'm not familiar with API and JSON format, it took pretty long to search for open data sources. So I went through all the websites suggested by the lecturer and checked on multiple JSON files and finally could pick up exciting JSON datasets. Then, I could design the data structure and then visualize the document structure via a class diagram. The most exciting but challenging part is probably the aggregation pipelines. So I could try several query operators and aggregation pipeline stages such as match, projection, sorting, limit, sort, and aggregation pipeline operators. Many jupyter and Pycharm examples present many different types of queries. Even though there were several difficulties along the way, and time-consuming for me, I loved it when I could query the desired tables to answer the questions.

The module offers various valuable materials like the summary of the DataCamp course, data modeling, class diagram, entity relationship diagram, and many more. I really appreciate this module because I acquired relational databases, non-relational databases, and the comparisons between different types of databases regarding their characteristics and use cases. For example, the main differences between NoSQL compared to SQL are dynamic schema, document, key-value store, unstructured data, and horizontally scalable. There are also different categories of NoSQL databases, such as graph databases, key-value databases, and wide-column databases, besides document databases like MongoDB. Furthermore, the figure of the CAP theorem with three elements of consistency, availability, and partition tolerance gives an overview of existing databases on the market. I also love the first two lectures when we could revise SQL before jumping into NoSQL. The lecturer's website wiki.revolytics is super well structured and easy to follow the course. Installing new tools and working with Mongo Cloud Atlas or a local machine was also well explained.

This class is definitely the key to opening the door to the NoSQL wonderland.