Setups & Imports

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
%matplotlib inline

from collections import Counter

import scipy.stats as sts

from datetime import date

from tqdm import tqdm

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
C:\Users\lavre\anaconda3\Lib\site-packages\paramiko\transport.py:219: CryptographyDeprecationWarning: Blowfish has be
en deprecated
  "class": algorithms.Blowfish,
```

EDA

Загрузим датасет прослушиваний музыки с kaggle (ссылка - https://www.kaggle.com/competitions/kkbox-music-recommendation-challenge/data (https://www.kaggle.com/competitions/kkbox-music-recommendation-challenge/data)) и проведем exploratory data analysis для каждой из таблиц.

In [2]:
```
songs_df = pd.read_csv('data/songs.csv')
song_extra_info_df = pd.read_csv('data/song_extra_info.csv')
members_df = pd.read_csv('data/members.csv')
train_df = pd.read_csv('data/train.csv')
test_df = pd.read_csv('data/test.csv')
sample_submission_df = pd.read_csv('data/sample_submission.csv')
```

General information about dataset

songs_df

Таблица содержит информацию о музыкальном треке и имеет следующие поля:

- song_id - id музыкального трека;
- song_length - длина музыкального трека (в мс);
- genre_ids - id жанров, к которому относится музыкальный трек(некоторые имеют несколько жаннров);
- artist_name - имя артиста;
- composer - имя композитора;
- lyricist - автор текста;
- language - язык произведения;

In [3]:
```
songs_df.sample(10)
```

Out[3]:

	song_id	song_length	genre_ids	artist_name	composer	lyricist	language
1753532	5adIWoR3U8LISjrFKxTE3QzhtrIcf5bblf6JTz8uiW4=	266240	465	Elka	ELKA	wit☆	17.0
1024319	Mg5gczbqI9pSGzIVfSJ5TN4vNusNW2gANCY/Cb/t0kQ=	191147	139|125|109	The Blenders	NaN	NaN	52.0
1591547	w1//42G5PhFARMqGR8VW3KRjf3AwPsbMztuXAadxX4k=	222540	1259	MC Yogi	NaN	NaN	52.0
2242455	p9hhn7rY2R37BUkHqReK7GlVoU3rPIcuCiXTz0RdUeM=	344119	2122	KEI SASAKI (佐々木 慧)	Vernon Duke	NaN	17.0
495174	j1l3fvvuE+patqq/izI5oC5bYAcKwnPbnrLF3EqXLrg=	229134	359	Dia Frampton	Dia Frampton| Tim Anderson| Daniel Heath	NaN	52.0
2085618	o/ZV5YCnS8gOr1hmjxJ8L8y5Zx9J7sex19w+p99QpAc=	323686	940	Pianissimo Brothers	NaN	NaN	-1.0
1875146	MhAjo8k858FG+1Yr1+LkmyvVSwEASQuI0hgzE1DBrwI=	114149	1981	The Highwaymen	Rossi	NaN	52.0
1232832	+90ME5mnrp6s23j1yw4wTTSFMqbipxAdhTkuEgvBpeo=	190682	465	Fatman Scoop	NaN	NaN	52.0
2234228	cESbN7rydnt3aIf6NGnfUHKEAtVWnGNWRB9dFmv1phc=	273449	NaN	Statue	NaN	NaN	52.0
1238373	7HpJvyIEsOrfHSi6WVjkS5Hi+/sGPhvM5xX6Cdz3Lx8=	260063	2022	Backtrack Professional Karaoke Band	NaN	NaN	52.0

In [4]:
```
songs_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2296320 entries, 0 to 2296319
Data columns (total 7 columns):
 #   Column       Dtype  
---  ------       -----  
 0   song_id      object 
 1   song_length  int64  
 2   genre_ids    object 
 3   artist_name  object 
 4   composer     object 
 5   lyricist     object 
 6   language     float64
dtypes: float64(1), int64(1), object(5)
memory usage: 122.6+ MB
```

In [5]:
```
print(f'Количество дупликатов в таблице songs_df: {songs_df.duplicated().sum()}')
```

```
Количество дупликатов в таблице songs_df: 0
```

Построим датафрэйм, в котором отразим количество пропусков и уникальных значений, как в абсолютном, так и в процентном соотношении.

In [6]:
```
songs_nan_df = pd.DataFrame(dict(songs_df.isna().sum()).items(), columns=['column', 'misses_cnt'])
songs_nan_df['misses_percent'] = round((songs_nan_df['misses_cnt'] / songs_df.shape[0]) * 100, 2)
songs_nan_df['unique_cnt'] = dict(songs_df.nunique()).values()
songs_nan_df['unique_percent'] = round((songs_nan_df['unique_cnt'] / songs_df.shape[0]) * 100, 2)
songs_nan_df
```

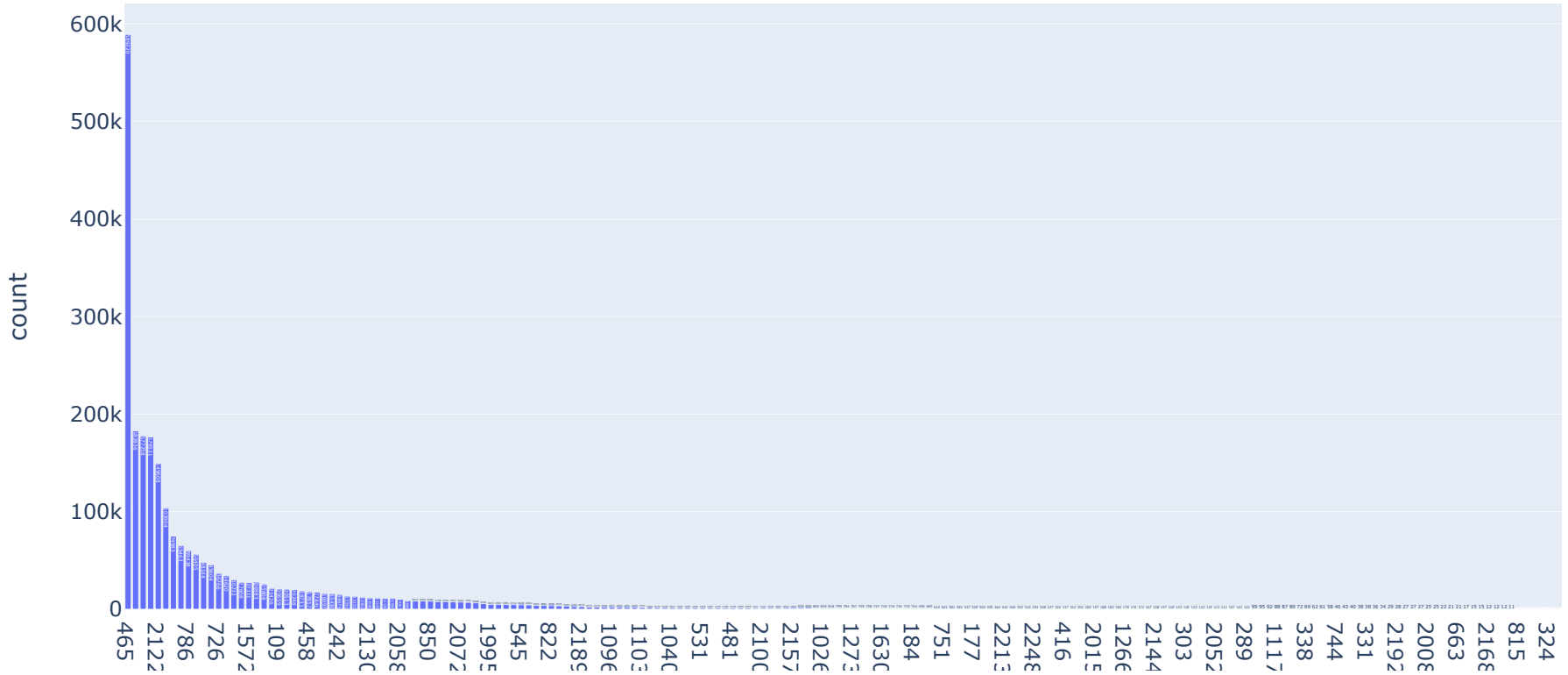
Out[6]:

	column	misses_cnt	misses_percent	unique_cnt	unique_percent
0	song_id	0	0.00	2296320	100.00
1	song_length	0	0.00	146534	6.38
2	genre_ids	94116	4.10	1045	0.05
3	artist_name	0	0.00	222363	9.68
4	composer	1071358	46.66	329822	14.36
5	lyricist	1945306	84.71	110924	4.83
6	language	1	0.00	10	0.00

Рассмотрим количество уникальных жанров в таблице.

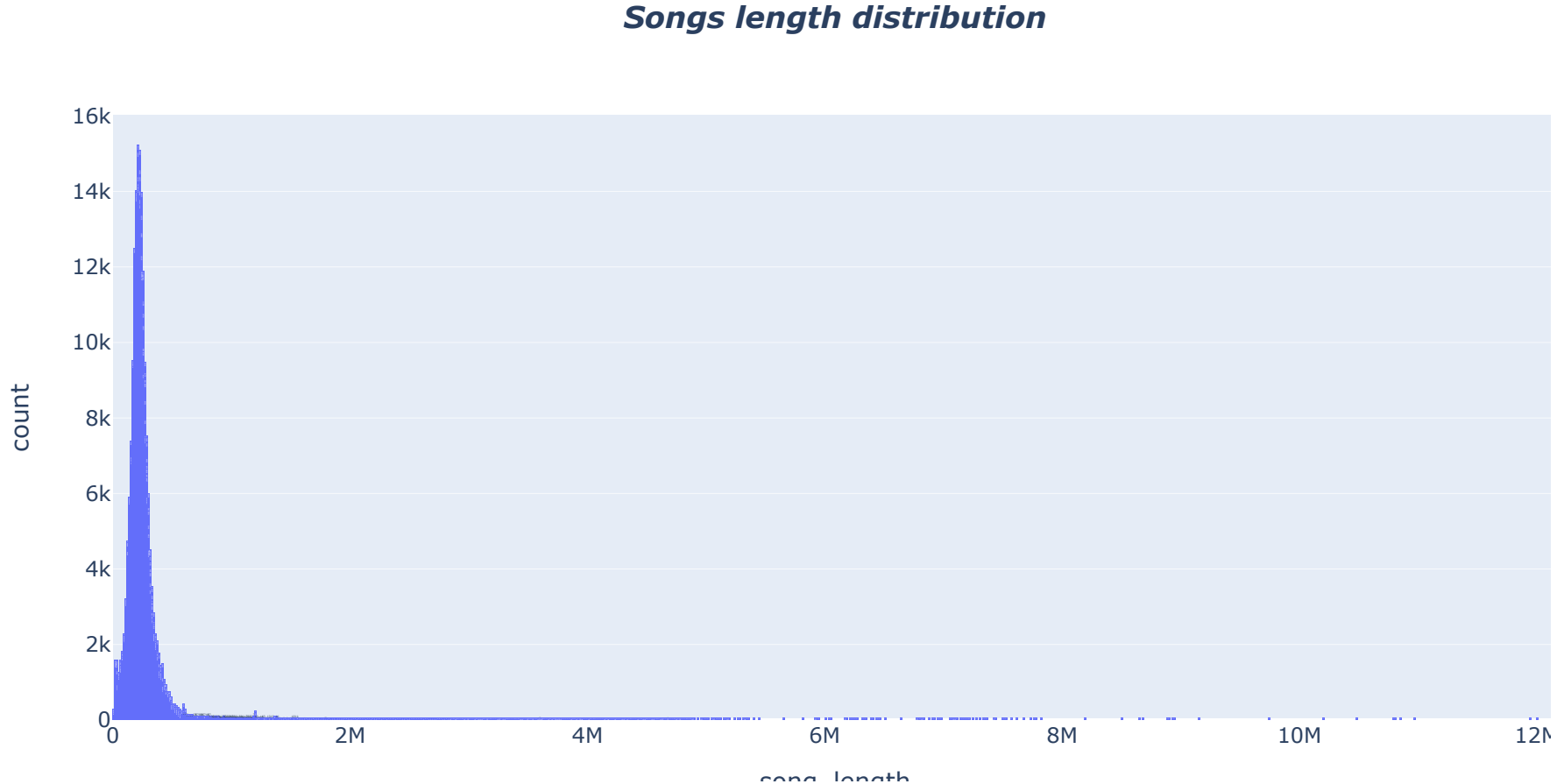
In [7]:

```
unique_genres = []

for x in songs_df.genre_ids.dropna():
    if isinstance(x, str):
        unique_genres.extend(x.split('|'))
    else:
        unique_genres.append(x)

fig = px.bar(
    pd.DataFrame(Counter(unique_genres).items(), columns=['genre_ids', 'count']),
    x='genre_ids',
    y='count',
    text_auto='outside',
    title='<i><b>Genres count</b></i>',
)

fig.update_xaxes(categoryorder='total descending')

fig.update_layout(title_x=0.5)

fig.show()
```



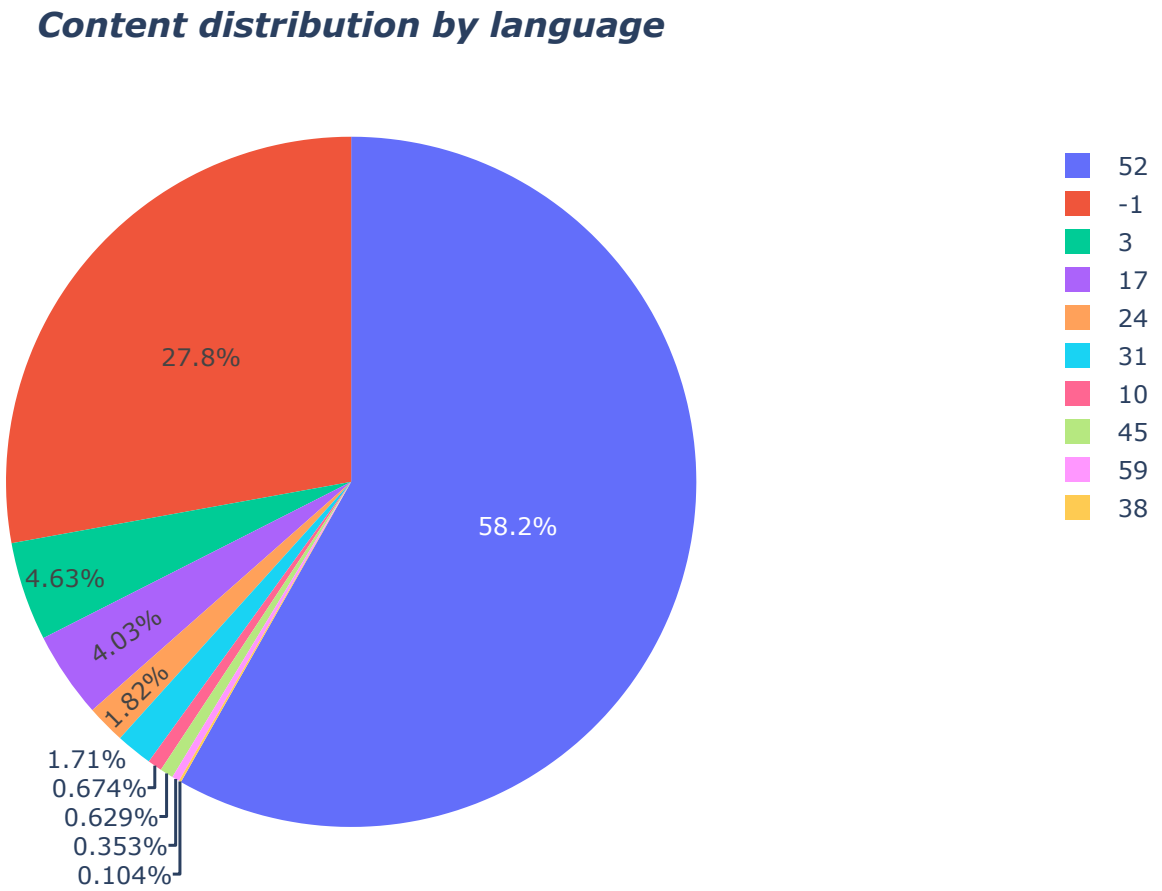
Наблюдается яркий перекос в сторону 10 самых популярных с убывающем трендом по остальным жанрам.

In [8]:
```
fig = px.histogram(
    songs_df.song_length,
    x='song_length',
    text_auto='outside',
    title='<i><b>Songs length distribution</b></i>',
)

fig.update_xaxes(categoryorder='total descending')

fig.update_layout(title_x=0.5)

fig.show()
```



Построенная гистограмма длинны музыкальных треков имеет нормальное распределение, что вполне соотносится с бытовой логикой.
Также нужно выяснить наиболее популярный язык треков для этого нанесем на pie plot сгруппированные по id язык.

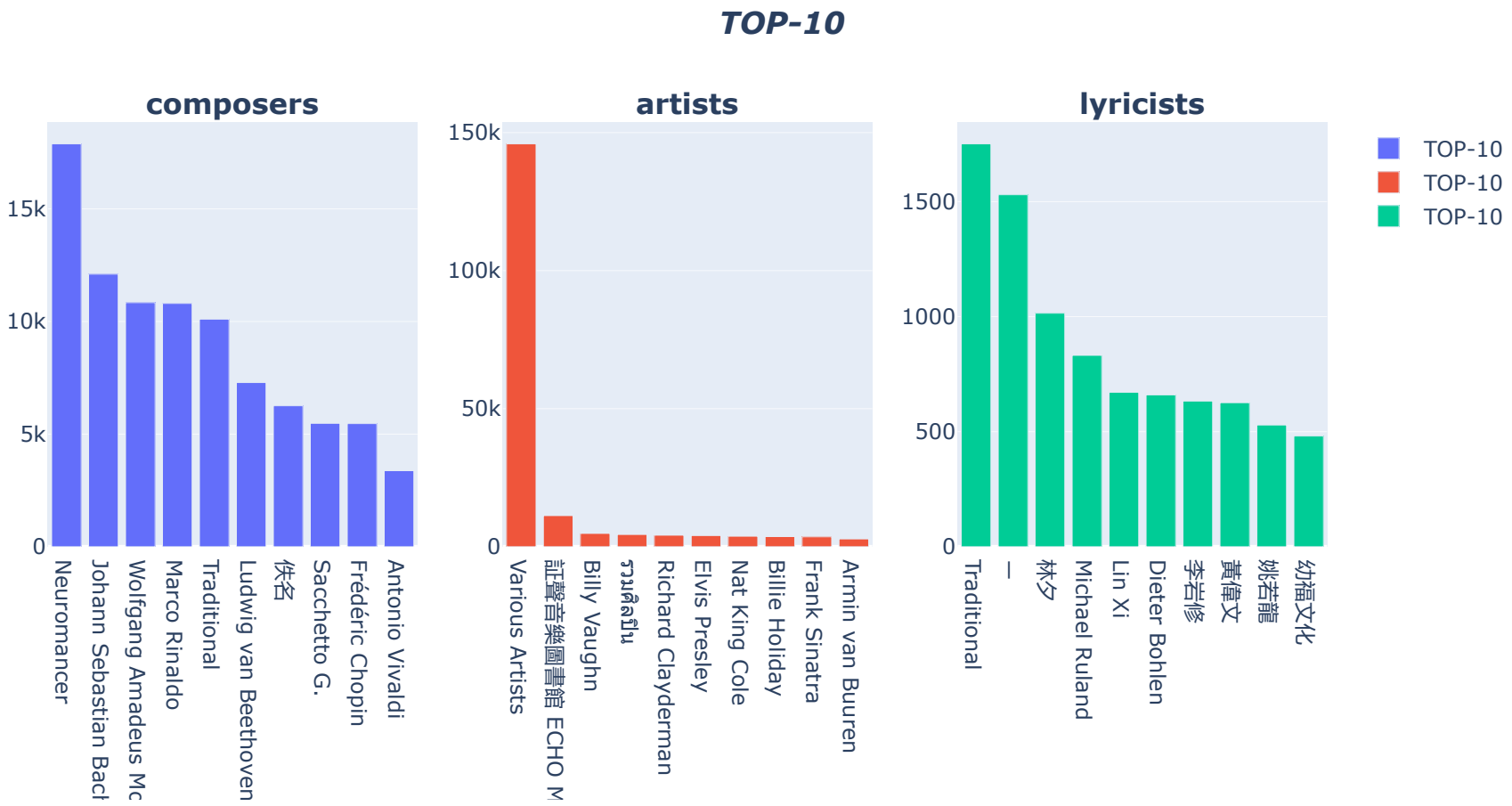
In [9]:
```
fig = px.pie(
    pd.DataFrame(dict(songs_df.groupby('language')['song_id'].count()).items(), columns=['language', 'count']),
    values='count',
    names='language',
    title='<i><b>Content distribution by language</b></i>'
)

fig.update_layout(title_x=0.5)

fig.show()
```



Из pie plot можно сделать вывод, что наиболее популярными являются 2 - 52 и -1. И исходя из данных датасета - это английский (52) и язык трека неизвестен (-1)

In [10]:

```
fig = make_subplots(rows=1, cols=3,
                    subplot_titles=['<b>composers</b>', '<b>artists</b>', '<b>lyricists</b>'])

fig.add_trace(
    go.Bar(
        x=pd.DataFrame(dict(songs_df.groupby('composer')['song_id'].count()).items(), columns=['composer', 'count']).s
        y=pd.DataFrame(dict(songs_df.groupby('composer')['song_id'].count()).items(), columns=['composer', 'count']).s
        name='TOP-10 composers'
    ),
    1, 1
)

fig.add_trace(
    go.Bar(
        x=pd.DataFrame(dict(songs_df.groupby('artist_name')['song_id'].count()).items(), columns=['artist_name', 'coun
        y=pd.DataFrame(dict(songs_df.groupby('artist_name')['song_id'].count()).items(), columns=['artist_name', 'coun
        name='TOP-10 artists'
    ),
    1, 2
)

fig.add_trace(
    go.Bar(
        x=pd.DataFrame(dict(songs_df.groupby('lyricist')['song_id'].count()).items(), columns=['lyricist', 'count']).s
        y=pd.DataFrame(dict(songs_df.groupby('lyricist')['song_id'].count()).items(), columns=['lyricist', 'count']).s
        name='TOP-10 lyricists'
    ),
    1, 3
)


fig.update_layout(title_text='<i><b>TOP-10</b></i>', title_x=0.5)

fig.show()
```



Из данных bar plot ясно, что наиболее популярны классические композиторы, "попсовые" артисты.

song_extra_info_df

Таблица содержит дополнительную информацию о музыкальном треке и имеет следующие поля:

- song_id - id музыкального трека;

- song name - название песни;
- isrc - международный стандартный код записи для уникальной идентификации звукозаписей и музыкальных видеозаписей;

In [11]:
```
song_extra_info_df.sample(10)
```

Out[11]:

	song_id	name	isrc
1408578	MPpTYIl8tHeGZDCIYWZ5xPQPBCE97MvdEbjVfKjLIv0=	八年的爱	FR10S1669716
1999296	mL9zKEXmDMvyAn/q8coUTIcsueiLR28JF0K6er8shkM=	夏の思ひ出（カニ）	JPZ921312797
989842	gre4LTv5AH1GPatqO7P1EjPyoIdk+vsBY659bPrq3Qg=	Streams Flow	GBPS81524023
293026	8gkFvY8dDLy9JJwWcOz9XBFvklvX5OorztGN0cEhhQ0=	Nothing's Gonna Stop Us Now(Albert Hammond| Di...	NaN
745237	St2hvl0LOzUlmh7TgheZzB4KGKCpXwOX0tQWlepTIUs=	You Are the Sun	CA9T91500012
106903	SLBfm8Qfs69mnrsTxO6Cx4bNU+/8Pe6fItXAB73bNCM=	最好的安排 (Best Plan)	CAN111600329
445389	RSEXOcCyvJ+uca0Lf2rJHY87outoGfetx9lChxnScqw=	Ave Maria	NLA309700126
1628277	mEifKmS6ih8AlkqKJOS/yetNgcsCHgrQg8JbMszHBTA=	The Party's over Now	QMFMG1464696
750420	X9h+6X5AYeEHHUa3Fs16mWEIL8CqgM8/PhqIlvv2yrY=	Da Funk (Dubstep Remix)	FR0W69913618
1868429	nr/D0Cawu7bNsbkfIqs71ZMLa2zZ39BHKx3pmcjNhOM=	Skype Call	BGA471303570

In [12]:
```
song_extra_info_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2295971 entries, 0 to 2295970
Data columns (total 3 columns):
 #   Column   Dtype 
---  ------   ----- 
 0   song_id  object
 1   name     object
 2   isrc     object
dtypes: object(3)
memory usage: 52.6+ MB
```

In [13]:
```
print(f'Количество дупликатов в таблице song_extra_info_df: {song_extra_info_df.duplicated().sum()}')
```

```
Количество дупликатов в таблице song_extra_info_df: 0
```

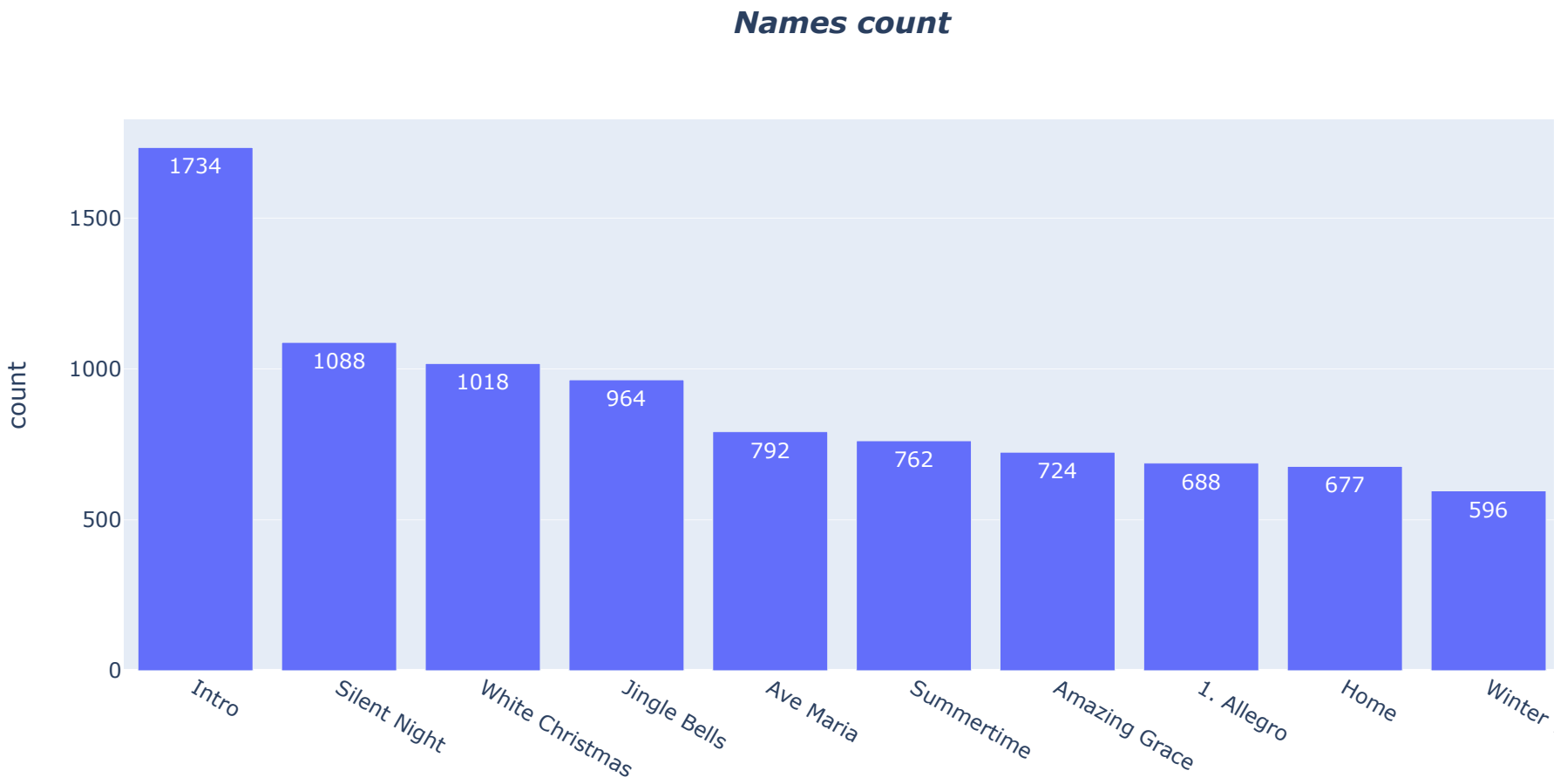
In [14]:
```
song_info_nan_df = pd.DataFrame(dict(song_extra_info_df.isna().sum()).items(), columns=['column', 'misses_cnt'])
song_info_nan_df['misses_percent'] = round((song_info_nan_df['misses_cnt'] / song_extra_info_df.shape[0]) * 100, 2)
song_info_nan_df['unique_cnt'] = dict(song_extra_info_df.nunique()).values()
song_info_nan_df['unique_percent'] = round((song_info_nan_df['unique_cnt'] / song_extra_info_df.shape[0]) * 100, 2)
song_info_nan_df
```

Out[14]:

	column	misses_cnt	misses_percent	unique_cnt	unique_percent
0	song_id	0	0.00	2295971	100.00
1	name	3	0.00	1168978	50.91
2	isrc	136548	5.95	1806825	78.70

In [15]:

```
fig = px.bar(
    pd.DataFrame(dict(song_extra_info_df.groupby('name')['song_id'].count()).items(), columns=['name', 'count']).sort_
    x='name',
    y='count',
    text_auto='outside',
    title='<i><b>Names count</b></i>',
)

fig.update_xaxes(categoryorder='total descending')

fig.update_layout(title_x=0.5)

fig.show()
```



members_df

Таблица содержит информацию о пользователе и имеет следующие поля:

- msno - id пользователя;
- city - город;
- bd - возраст;

- gender - пол;
- registered_via - метод регистрации;
- registration_init_time - дата регистрации в формате: %Y%m%d;
- expiration_date - дата окончания срока действия регистрации в формате: format %Y%m%d;

In [16]:
```
members_df.sample(10)
```

Out[16]:

	msno	city	bd	gender	registered_via	registration_init_time	expiration_date
4346	xlQpnVInh7lFcx3o3EJUcdNIRH6vQ6ycLKlkp8g9wSw=	1	0	NaN	4	20161217	20161220
14643	17JlEr1MDqIAoQHg48Iw5Vp7kPnI8PbESvb/m4A7+Ac=	6	23	female	3	20130828	20171008
27552	NLHnZSK4pOPsNSGoWwG7uQQBtEsMmGlFOkL8wV5CzOk=	1	0	NaN	4	20161119	20161126
997	SS/a9A+PdNaLGf6Mdrfvg5hJ4c5G+3MbMB3fevFARFM=	1	0	NaN	4	20161218	20161221
2203	nWPPpL02/T6wzbtg6a2yDNVB9WYf/94SduoQBD2TsoM=	1	0	NaN	7	20161017	20170916
7946	xuv8pFi3jYf8uOECA7uCz0vLh4xB/Z8o2IGGkMye4Ko=	1	0	NaN	4	20161231	20170103
24457	3ggfD7BT/hfeMMlRhit7e2UW0KE46crVJ2Tbyu+iMOo=	7	22	female	3	20121206	20170924
1318	ROSWh0VFV3aO4e4fkGaWw+9rC3zusWZ3TOsZ9OrQqzE=	1	0	NaN	7	20160209	20160309
10293	ZqPg5C9cGdBWHsjU35z1j7CcAlUEwvYiR6xLXyKTHbI=	1	0	NaN	4	20170101	20170709
29047	+jQKqtJRJqSFRQLsDKPGpoW+ya/NQ1o4BmnGZ+bkNZY=	21	0	NaN	3	20140215	20170608

In [17]:
```
members_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34403 entries, 0 to 34402
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   msno                    34403 non-null  object
 1   city                    34403 non-null  int64
 2   bd                      34403 non-null  int64
 3   gender                  14501 non-null  object
 4   registered_via          34403 non-null  int64
 5   registration_init_time  34403 non-null  int64
 6   expiration_date         34403 non-null  int64
dtypes: int64(5), object(2)
memory usage: 1.8+ MB
```

In [18]:
```
print(f'Количество дупликатов в таблице members_df: {members_df.duplicated().sum()}')
```

```
Количество дупликатов в таблице members_df: 0
```

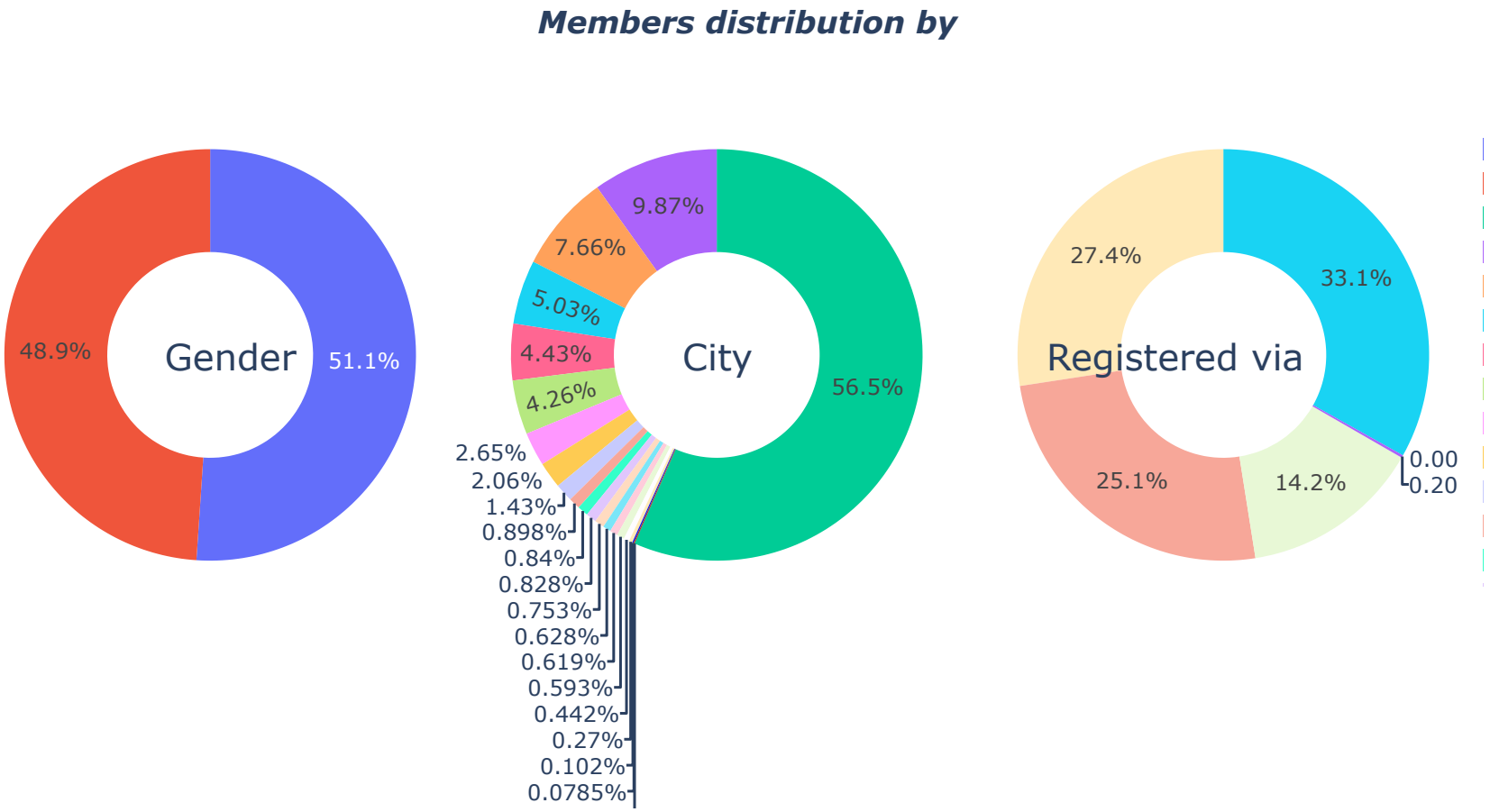
In [19]:
```
members_nan_df = pd.DataFrame(dict(members_df.isna().sum()).items(), columns=['column', 'misses_cnt'])
members_nan_df['misses_percent'] = round((members_nan_df['misses_cnt'] / members_df.shape[0]) * 100, 2)
members_nan_df['unique_cnt'] = dict(members_df.nunique()).values()
members_nan_df['unique_percent'] = round((members_nan_df['unique_cnt'] / members_df.shape[0]) * 100, 2)
members_nan_df
```

Out[19]:

	column	misses_cnt	misses_percent	unique_cnt	unique_percent
0	msno	0	0.00	34403	100.00
1	city	0	0.00	21	0.06
2	bd	0	0.00	95	0.28
3	gender	19902	57.85	2	0.01
4	registered_via	0	0.00	6	0.02
5	registration_init_time	0	0.00	3862	11.23
6	expiration_date	0	0.00	1484	4.31

In [20]:

```
fig = make_subplots(rows=1, cols=3, specs=[[{'type':'domain'}, {'type':'domain'}, {'type':'domain'}]])

fig.add_trace(
    go.Pie(
        labels=pd.DataFrame(dict(members_df.groupby('gender')['msno'].count()).items(), columns=['gender', 'count'])['
        values=pd.DataFrame(dict(members_df.groupby('gender')['msno'].count()).items(), columns=['gender', 'count'])['
        name="Gender"
    ), 1, 1
)

fig.add_trace(
    go.Pie(
        labels=pd.DataFrame(dict(members_df.groupby('city')['msno'].count()).items(), columns=['city', 'count'])['city
        values=pd.DataFrame(dict(members_df.groupby('city')['msno'].count()).items(), columns=['city', 'count'])['coun
        name="City"
    ), 1, 2
)

fig.add_trace(
    go.Pie(
        labels=pd.DataFrame(dict(members_df.groupby('registered_via')['msno'].count()).items(), columns=['registered_v
        values=pd.DataFrame(dict(members_df.groupby('registered_via')['msno'].count()).items(), columns=['registered_v
        name="Registered via"
    ), 1, 3
)

fig.update_traces(hole=.5, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="<i><b>Members distribution by</b></i>",
    title_x=0.5,
    annotations=[dict(text='Gender', x=0.11, y=0.5, font_size=20, showarrow=False),
                 dict(text='City', x=0.5, y=0.5, font_size=20, showarrow=False),
                 dict(text='Registered via', x=0.915, y=0.5, font_size=20, showarrow=False)])

fig.show()
```



Датасет сбалансирован с гендерной точки зрения и методов регистрации, явный перекос возникает в сторону города 1.

In [21]:
```
def convert_to_date(x: int) -> date:
    year = int(str(x)[: 4])
    month = int(str(x)[4: 6])
    day = int(str(x)[6: ])
    return date(year, month, day)
```

In [22]:
```
members_df['registration_init_time'] = members_df['registration_init_time'].apply(convert_to_date)
members_df['expiration_date'] = members_df['expiration_date'].apply(convert_to_date)
members_df.head()
```

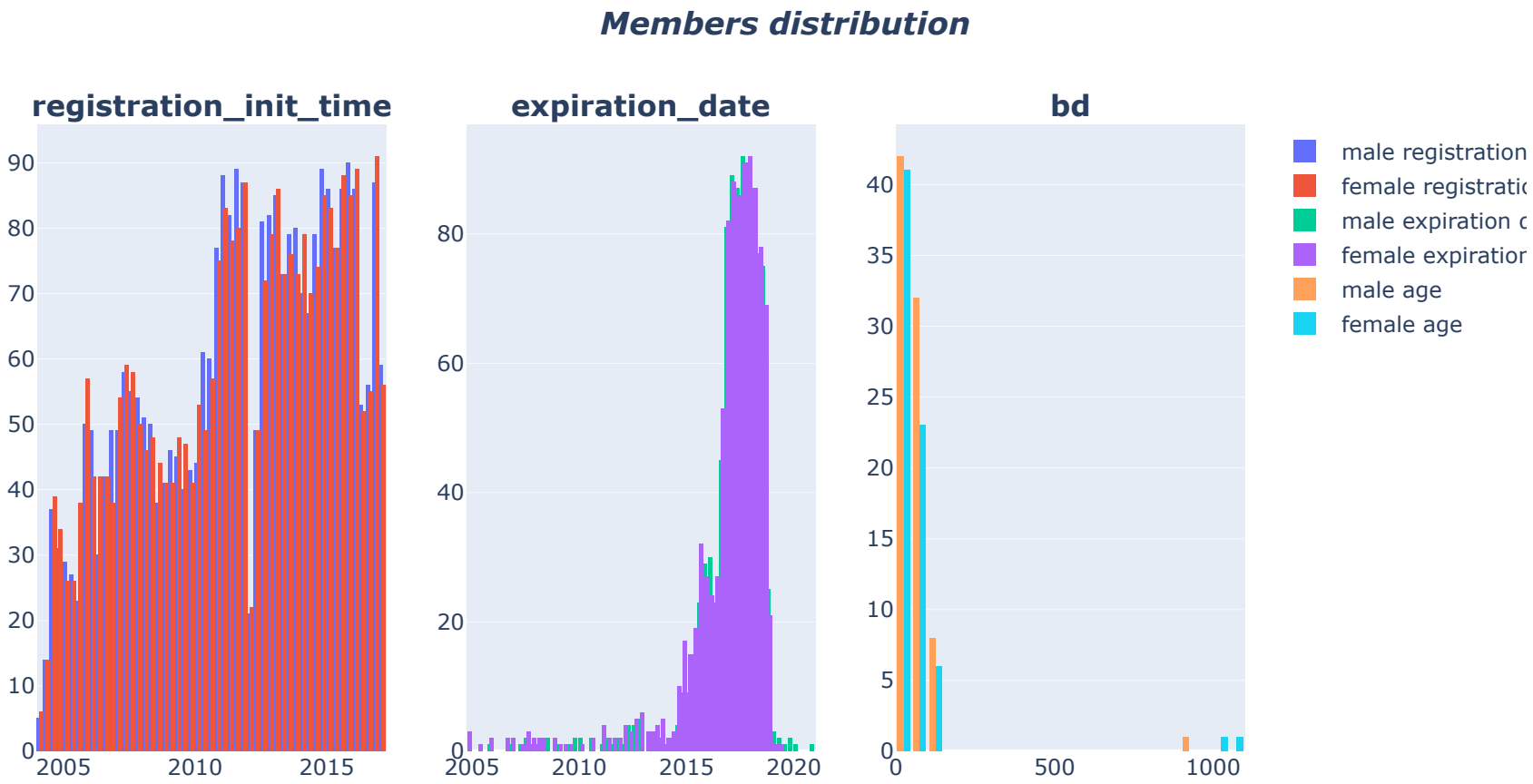
Out[22]:

	msno	city	bd	gender	registered_via	registration_init_time	expiration_date
0	XQxgAYj3klVKjR3oxPPXYYFp4soD4TuBghkhMTD4oTw=	1	0	NaN	7	2011-08-20	2017-09-20
1	UizsfmJb9mV54qE9hCYyU07Va97c0lCRLEQX3ae+ztM=	1	0	NaN	7	2015-06-28	2017-06-22
2	D8nEhsIOBSoE6VthTaqDX8U6lqjJ7dLdr72mOyLya2A=	1	0	NaN	4	2016-04-11	2017-07-12
3	mCuD+tZ1hERA/o5GPqk38e041J8ZsBaLcu7nGoIIvhI=	1	0	NaN	9	2015-09-06	2015-09-07
4	q4HRBfVSssAFS9iRfxWrohxuk9kCYMKjHOEagUMV6rQ=	1	0	NaN	4	2017-01-26	2017-06-13

In [23]:
```
male_registration_df = pd.DataFrame(dict(members_df[members_df.gender == 'male'].groupby('registration_init_time')['ms
male_registration_df['registration_init_time'] = male_registration_df['registration_init_time']

female_registration_df = pd.DataFrame(dict(members_df[members_df.gender == 'female'].groupby('registration_init_time')
female_registration_df['registration_init_time'] = female_registration_df['registration_init_time']

male_expiration_df = pd.DataFrame(dict(members_df[members_df.gender == 'male'].groupby('expiration_date')['msno'].coun
male_expiration_df['expiration_date'] = male_expiration_df['expiration_date']

female_expiration_df = pd.DataFrame(dict(members_df[members_df.gender == 'female'].groupby('expiration_date')['msno'].
female_expiration_df['expiration_date'] = female_expiration_df['expiration_date']

male_bd_df = pd.DataFrame(dict(members_df[members_df.gender == 'male'].groupby('bd')['msno'].count()).items(), columns
female_bd_df = pd.DataFrame(dict(members_df[members_df.gender == 'female'].groupby('bd')['msno'].count()).items(), col
```

In [24]:
```
fig = make_subplots(rows=1,
                    cols=3,
                    subplot_titles=['<b>registration_init_time</b>',
                                    '<b>expiration_date</b>',
                                    '<b>bd</b>'])

fig.add_trace(
    go.Histogram(
        x=male_registration_df['registration_init_time'],
        y=male_registration_df['count'],
        name='male registration init time'
    ),
    1, 1
)

fig.add_trace(
    go.Histogram(
        x=female_registration_df['registration_init_time'],
        y=female_registration_df['count'],
        name='female registration init time'
    ),
    1, 1
)

fig.add_trace(
    go.Histogram(
        x=male_expiration_df['expiration_date'],
        y=male_expiration_df['count'],
        name='male expiration date'
    ),
    1, 2
)

fig.add_trace(
    go.Histogram(
        x=female_expiration_df['expiration_date'],
        y=female_expiration_df['count'],
        name='female expiration date'
    ),
    1, 2
)
```

```
fig.add_trace(
    go.Histogram(
        x=male_bd_df['bd'],
        y=male_bd_df['count'],
        name='male age'
    ),
    1, 3
)

fig.add_trace(
    go.Histogram(
        x=female_bd_df['bd'],
        y=female_bd_df['count'],
        name='female age'
    ),
    1, 3
)

fig.update_layout(title_text='<i><b>Members distribution</b></i>', title_x=0.5)

fig.show()
```



Что женщины, что мужчины имеют схожие распределения по представленным выше характеристикам.

train_df

Таблица содержит сводную информацию о действиях пользователя и имеет следующие поля:

- msno - id пользователя;

- song_id - id музыкального трека;
- source_system_tab - имя вкладки, на которой было вызвано событие;
- source_screen_name - имя макета, который видит пользователь;
- source_type - точка входа пользователя, который впервые воспроизводит музыку в мобильных приложениях;
- target - таргет, target = 1 означает, что повторяющееся прослушивание событие(я) запускается в течение месяца после самого первого наблюдаемого события прослушивания пользователя, target = 0 в противном случае;

In [25]:
```
train_df.sample(10)
```

Out[25]:

	msno	song_id	source_system_tab	source_
1018713	AmUm6FabYMPET+GG1qfoG2Y9vWiPP6C6Zb66i0YWeM8=	xsWzslT0QqKGr7/4ctQdV2VFuQGQ9E/CluVRhcFLbxI=	my library	Loc
3387641	8A47sKGKyUF7z9LWDnqc6LXjyzvGOAR54xWNJCP0TI8=	uXapcmex8CvnTUnKhiNb2ExHmKq/r6nji2PdB94KtXM=	my library	Loc
3249065	CxVQr0fcRidsFr36QZpF3jpaGbxOmVShshSJzm9+apI=	P47jK0LEXjUXp34Wn+qoceU2mE9nsJMJOiBOrcDgCvY=	discover	
1855960	se0VD95KE2C+EddU+ioCCpaRal8d0S/DzXB9RpyGHlE=	cwvJcKVG91gQpMoVL5Bvu+OZlDGm1BJWnJjGEMIqw90=	my library	Loc
6729262	wzFaeYMG3tw6TGbSaw5lh/w+kd8KSwImymjauS12huw=	e2Hg57tFjJ94Doqc07PhpiYphmeEfz1MRSC7jnKTnpU=	my library	Loc
3547151	GuMK9gcXglqNr83zo5Tfx1bQwsYqdDXa6n+y/575IPk=	QxFA4zB/FxzlnWsnuWTXPJVEQQDP7L5axAJvuSsX9XE=	discover	Onlir
6340995	BF/EbdrrG4SACpAFQKxO+GbRiZi6lSrEBcy/3Mvlzso=	e/upMhPFcXl3Rxq0Us1abr0FcThlZOeqKJhBobmltBU=	discover	
2968976	Kup6VNdSWBeGmKcPDcwHFrAQeA7yTaLLwTh0PJxTZUg=	UY93nUb/j9W5tt53uvMJtf0MZWtnLoc1XwuuBHV00Nk=	my library	Loc
6111301	hRQzo72cneZRGPQfGA87MwZOSt9o1UkeGieHNkcUwQw=	Lms2TIFpPWFBJlrUbVZvj1Aq+/o52fpzOmF0rP76mqY=	my library	Loc
6446960	bXwFo66yK++4lOZ2qZb45qwDWmfhLc6bfAtd4L3283I=	KZ5hwP74wRO6kRapVIprwodtNdVD2EVD3hkZmmyXFPk=	discover	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7377418 entries, 0 to 7377417
Data columns (total 6 columns):
 #   Column              Dtype 
---  ------              ----- 
 0   msno                object
 1   song_id             object
 2   source_system_tab   object
 3   source_screen_name  object
 4   source_type         object
 5   target              int64 
dtypes: int64(1), object(5)
memory usage: 337.7+ MB
```

In [27]:
```
print(f'Количество дупликатов в таблице train_df: {train_df.duplicated().sum()}')
```

```
Количество дупликатов в таблице train_df: 0
```

In [28]:
```
train_nan_df = pd.DataFrame(dict(train_df.isna().sum()).items(), columns=['column', 'misses_cnt'])
train_nan_df['misses_percent'] = round((train_nan_df['misses_cnt'] / train_df.shape[0]) * 100, 2)
train_nan_df['unique_cnt'] = dict(train_df.nunique()).values()
train_nan_df['unique_percent'] = round((train_nan_df['unique_cnt'] / train_df.shape[0]) * 100, 2)
train_nan_df
```

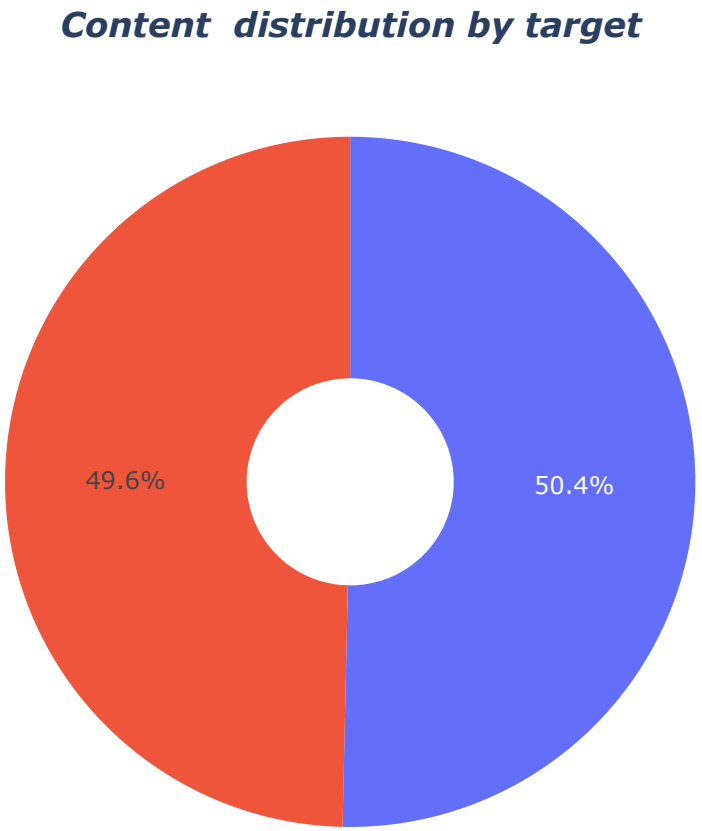
Out[28]:

	column	misses_cnt	misses_percent	unique_cnt	unique_percent
0	msno	0	0.00	30755	0.42
1	song_id	0	0.00	359966	4.88
2	source_system_tab	24849	0.34	8	0.00
3	source_screen_name	414804	5.62	20	0.00
4	source_type	21539	0.29	12	0.00
5	target	0	0.00	2	0.00

Рассмотрим распределение таргета в тренировочном датасете:

In [29]:

```
fig = px.pie(
    pd.DataFrame(dict(train_df.groupby('target')['song_id'].count()).items(), columns=['target', 'count']),
    values='count',
    names='target',
    hole=.3,
    title='<i><b>Content  distribution by target</b></i>'
)

fig.update_layout(title_x=0.5)

fig.show()
```



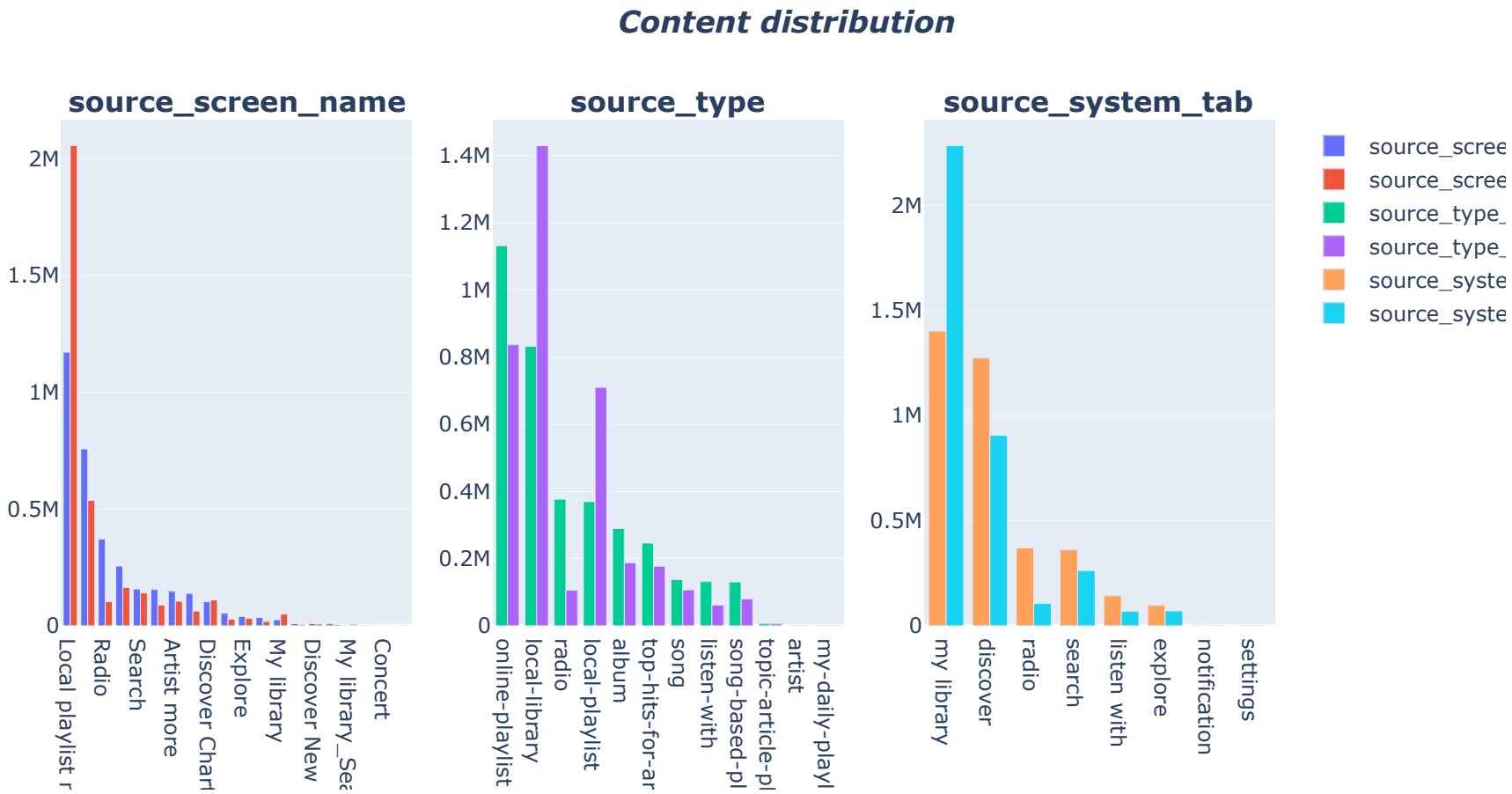
Можно сделать вывод, что датасет относительно таргета является сбалансированным. Теперь отобразим на графиках распределения source_screen_name, source_type и source_system_tab в зависимости от таргета.

In [30]:

```
source_screen_name_1 = pd.DataFrame(dict(train_df[train_df.target == 1.0].groupby('source_screen_name')['song_id'].cou
source_screen_name_0 = pd.DataFrame(dict(train_df[train_df.target == 0.0].groupby('source_screen_name')['song_id'].cou

source_type_1 = pd.DataFrame(dict(train_df[train_df.target == 1.0].groupby('source_type')['song_id'].count()).items(),
source_type_0 = pd.DataFrame(dict(train_df[train_df.target == 0.0].groupby('source_type')['song_id'].count()).items(),

source_system_tab_1 = pd.DataFrame(dict(train_df[train_df.target == 1.0].groupby('source_system_tab')['song_id'].count
source_system_tab_0 = pd.DataFrame(dict(train_df[train_df.target == 0.0].groupby('source_system_tab')['song_id'].count
```

In [31]:
```
fig = make_subplots(rows=1, cols=3,
                    subplot_titles=['<b>source_screen_name</b>', '<b>source_type</b>', '<b>source_system_tab</b>'])

fig.add_trace(
    go.Bar(
        x=source_screen_name_0['source_screen_name'],
        y=source_screen_name_0['count'],
        name='source_screen_name_0'
    ),
    1, 1
)

fig.add_trace(
    go.Bar(
        x=source_screen_name_1['source_screen_name'],
        y=source_screen_name_1['count'],
        name='source_screen_name_1'
    ),
    1, 1
)


fig.add_trace(
    go.Bar(
        x=source_type_0['source_type'],
        y=source_type_0['count'],
        name='source_type_0'
    ),
    1, 2
)

fig.add_trace(
    go.Bar(
        x=source_type_1['source_type'],
        y=source_type_1['count'],
        name='source_type_1'
    ),
    1, 2
)

fig.add_trace(
```

```
    go.Bar(
        x=source_system_tab_0['source_system_tab'],
        y=source_system_tab_0['count'],
        name='source_system_tab_0'
    ),
    1, 3
)

fig.add_trace(
    go.Bar(
        x=source_system_tab_1['source_system_tab'],
        y=source_system_tab_1['count'],
        name='source_system_tab_1'
    ),
    1, 3
)

fig.update_layout(title_text='<i><b>Content distribution</b></i>', title_x=0.5)

fig.show()
```



Заметно, что по соотношению таргетов в source_screen_name выделяются группы: Local playlist more и Radio, в source_type - local-library, radio, local-playlist, в source_system_tab - my_library и radio.

Relationships between data tables

In [32]:
```
full_songs_df = songs_df.merge(song_extra_info_df, how='outer')
full_train_df = train_df.merge(full_songs_df, how='left', on='song_id')
full_train_df = full_train_df.merge(members_df, how='left', on='msno')
display(full_train_df)
```

	msno	song_id	source_system_tab	source_s
0	FGtllVqz18RPiwJj/edr2gV78zirAiY/9SmYvia+kCg=	BBzumQNXUHKdEBOB7mAJuzok+IJA1c2Ryg/yzTF6tik=	explore	
1	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	bhp/MpSNoqoxOIB+/l8WPqu6jldth4DIpCm3ayXnJqM=	my library	Loca
2	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	JNWfrrC7zNN7BdMpsISKa4Mw+xVJYNnxXh3/Epw7QgY=	my library	Loca
3	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	2A87tzfnJTSWqD7gIZHisolhe4DMdzkbd6LzO1KHjNs=	my library	Loca
4	FGtllVqz18RPiwJj/edr2gV78zirAiY/9SmYvia+kCg=	3qm6XTZ6MOCU11x8FIVbAGH5l5uMkT3/ZalWG1oo2Gc=	explore	
...	...	...	...	
7377413	6xdFzPlrasIDD95mQWXVC3Bg4ptnGYtBl4ztVEZMddU=	VJTxizih/o28kXCbtPbIyWXScoXGvxyYtl6R+0YB5JM=	my library	Loca
7377414	ZxbVmt3Kh/XOH+h58c2Kdj6SjFZk+wnUO006IgWzMQE=	z1mqaU9YOX7T/PFDvUoWozdFq7rC3KwaQP7nFVprjMI=	search	

	msno	song_id	source_system_tab	source_s
7377415	ZxbVmt3Kh/XOH+h58c2Kdj6SjFZk+wnUO006IgWzMQE=	750RprmFfLV0bymtDH88g24pLZGVi5VpBAI300P6UOA=	search	
7377416	0aH4Hd3ziPSRHClRX8rkeOEaAG5EPPkW1mKGCdXEok0=	G8wgqObgeAMER/rVCIlgcNeQ8mm0CzF/GsxiMK8TTnA=	discover	Di
7377417	0aH4Hd3ziPSRHClRX8rkeOEaAG5EPPkW1mKGCdXEok0=	Ju0VGkjWeBUZCd7r5Az2hUImhMoWxWLUicOedsmvG0g=	discover	Di

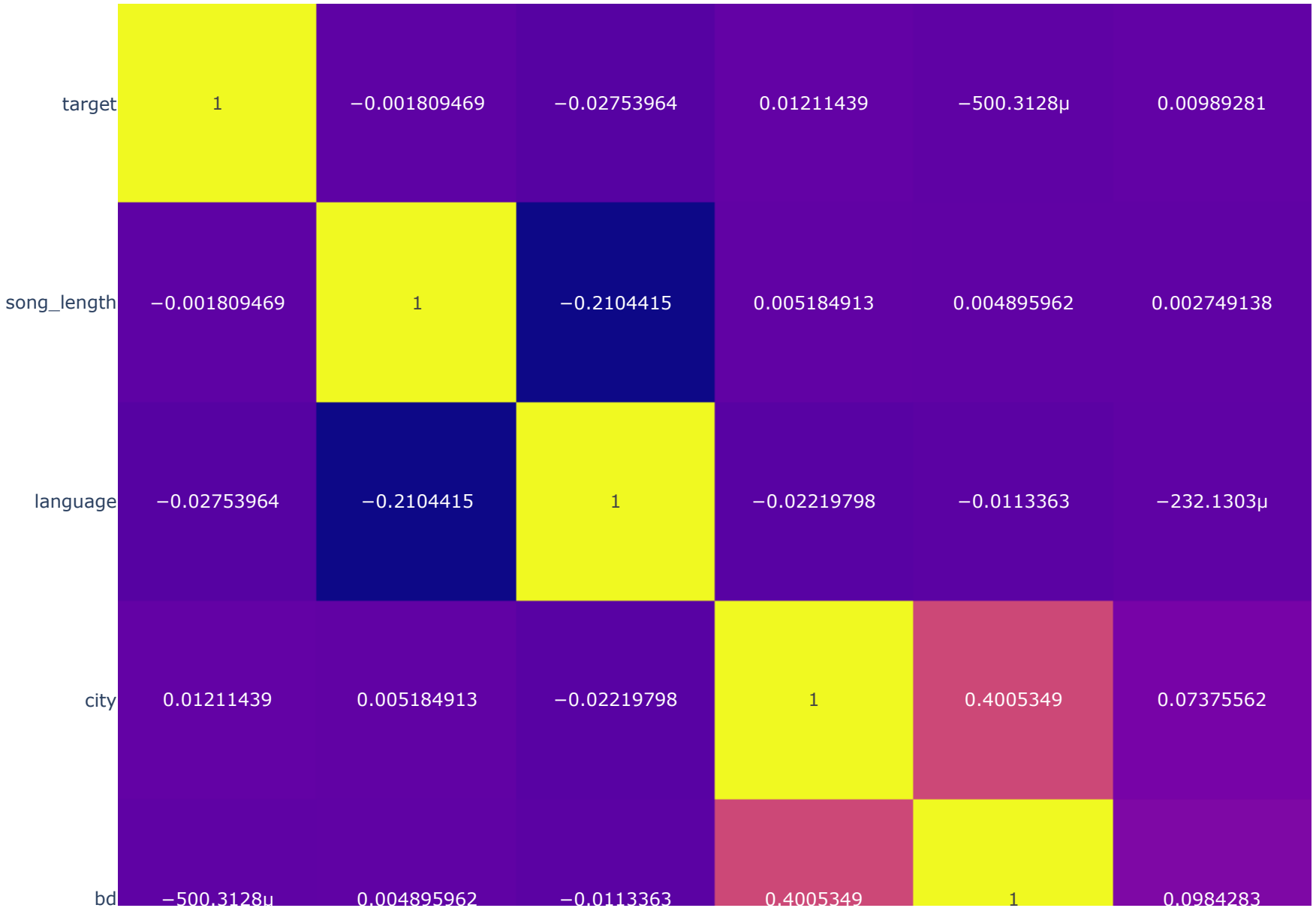
7377418 rows × 20 columns

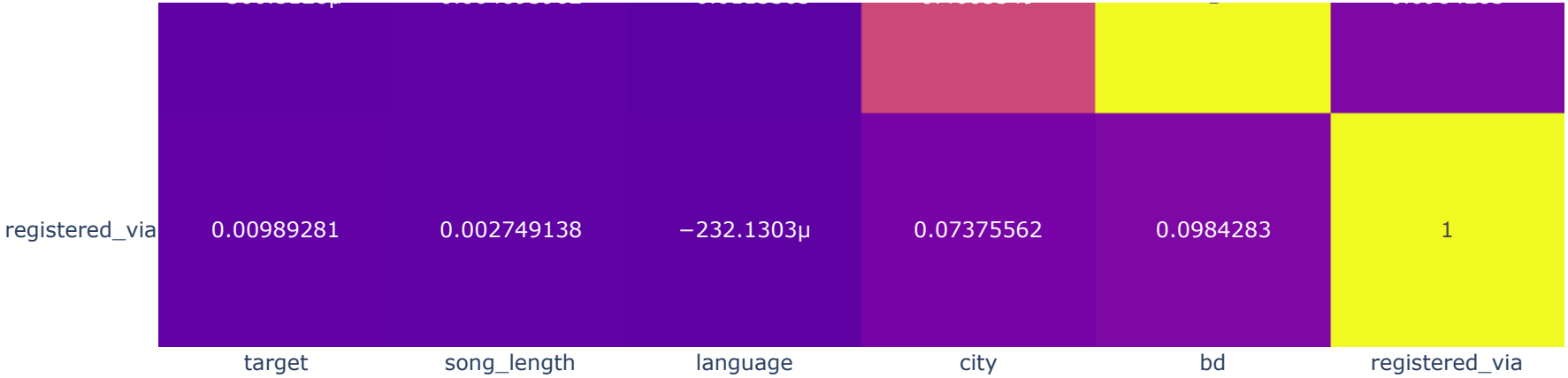
In [33]:
```
full_train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7377418 entries, 0 to 7377417
Data columns (total 20 columns):
 #   Column                  Dtype
---  ------                  -----
 0   msno                    object
 1   song_id                 object
 2   source_system_tab       object
 3   source_screen_name      object
 4   source_type             object
 5   target                  int64
 6   song_length             float64
 7   genre_ids               object
 8   artist_name             object
 9   composer                object
 10  lyricist                object
 11  language                float64
 12  name                    object
 13  isrc                    object
 14  city                    int64
 15  bd                      int64
 16  gender                  object
 17  registered_via          int64
 18  registration_init_time  object
 19  expiration_date         object
dtypes: float64(2), int64(4), object(14)
memory usage: 1.1+ GB
```

In [34]:

```
numeric_columns = full_train_df.select_dtypes(exclude=['object']).columns.to_list()

fig = px.imshow(
    full_train_df[numeric_columns].corr().values,
    x=numeric_columns,
    y=numeric_columns,
    title='<i><b>Correlation matrix</b></i>',
    width=1000,
    height=1000,
    text_auto=True
)

fig.update_layout(title_x=0.5)

fig.show()
```


Correlation matrix
target
song_length
language
city
bd
1
−0.001809469
−0.02753964
0.01211439
−500.3128µ
0.00989281
−0.001809469
1
−0.2104415
0.005184913
0.004895962
0.002749138
−0.02753964
−0.2104415
1
−0.02219798
−0.0113363
−232.1303µ
0.01211439
0.005184913
−0.02219798
1
0.4005349
0.07375562
−500.3128µ
0.004895962
−0.0113363
0.4005349
1
0.0984283



Как видно из матрицы корреляций выше, между числовыми признаками в датасете нет мультиколленеарности.

Data Quality

Проведем исследование на предмет соответствия распределений признаков между тренировочными и тестовыми данными, данный шаг необходим для того, чтобы мы были уверены, что обученная модель будет предсказывать на данных из того же распределения, иначе процесс предсказания на тестовых данных будет являться некорректным. Для этого выдвинем гипотезу для каждой пары признаков, где H0 будет означать, что данные имеют одинаковые(схожие) характеристики/распределения, H1 - обратно. Будем использовать тест Колмогорова-Смирнова и критерий Уилкоксона.

In [35]:
```
full_test_df = test_df.merge(full_songs_df, how='left', on='song_id')
full_test_df = full_test_df.merge(members_df, how='left', on='msno')
full_test_df.rename(columns={"index":"interaction_id"}, inplace=True)
full_test_df.sample(10)
```

Out[35]:

	id	msno	song_id	source_system_tal
365518	365518	gYZDAw85YMD2SYFperTGMxiaGWr3WOXoCQkuOm5xrsc=	IKMFuL0f5Y8c63Hg9BXkeNJjE0z8yf3gMt/tOxF4QNE=	discove
2241028	2241028	ISkUSnCm60kDLdEsnAVp6jwr2HkyIhDCEZLX82djjHg=	/+nHaBFjEnpfg6RxHoDrm9oelvLFTC4jV5Q6++48kDs=	listen wit
1067394	1067394	zBDf8RGVk0AqZgu50rQQhr6+2YB3w3qXI+6dXvWTx58=	OngdcKvNN+JTcoMwsB4U1LafHqeRyT4/pZa/8ZvN8gQ=	discove
1413250	1413250	e17Xzs36ff5Ro3rJOxI8s9VMo6ekpFJYSOgGJjtAV4Q=	66NCw1qHAROy0PiUC1tPNmGYqRvBSfvGUsO+46BooDc=	discove
1209838	1209838	8zKlJ2ZeNezauLqL5nvOeT8uiylVyHfwirD17LuWtEs=	yzXG3v5fWuLlgftXsbiLYpPjPE9V9GiKp7y36z7JQZA=	my librar
541644	541644	hwV8F+NZyYjOn+wsSNkqx5y78WcIai8RzNYD4fKWgX4=	17NI614y+KVbLs2vQUzx7gw1qWOhH0YmPvtp1yvDkNQ=	my librar
2037926	2037926	s8W6I0THBbthQ8Tc2id6heNlIK1YGVUCNjiVlvEnG78=	6Ky23/bBCeOsaz+iOkPjktEfjDz7Yaj0pQYQsQ3iXrs=	discove
602219	602219	RxSwsOQ/XleMoeAMgqRwqZf6n0rtgIAYLhaXAgoWAmc=	kNJTDbiTR6FZzdw67xEv6xmpud04T8C7ho1l3+lxkoI=	my librar
1254487	1254487	DHqJNl/XxAcF3xvEVmXzyT02hjYfUzc2u/NsWluj780=	DGxOa5t8tAwsA6JjtMP4pcFt6u/TR3PJVfdFHy+JYso=	discove
831613	831613	kdutUTwnF2xGfXPr2nHIehY5t/aoNM/H4eG4KqAfhAI=	7CLDq4FSjIVBKXJiUsgDmBjx2NQoxf1aFCqNJKTAaWg=	my librar

In [36]:
```
features = full_train_df.drop(['target'], axis=1).columns.to_list()
features
```

Out[36]:
```
['msno',
 'song_id',
 'source_system_tab',
 'source_screen_name',
 'source_type',
 'song_length',
 'genre_ids',
 'artist_name',
 'composer',
 'lyricist',
 'language',
 'name',
 'isrc',
 'city',
 'bd',
 'gender',
 'registered_via',
 'registration_init_time',
 'expiration_date']
```

In [37]:
```
features_dict = {'pairs' : [], 'p_value_ks' : [], 'ks_decision' : [], 'p_value_wilcoxon' : [], 'wilcoxon_decision' : [
checked_features = []

for feature in tqdm(features):
    data_group_1 = full_train_df.groupby(feature)[feature].count()
    data_group_2 = full_test_df.groupby(feature)[feature].count()
    n = min(len(data_group_1), len(data_group_2))
    _, p_val_ks = sts.ks_2samp(data_group_1, data_group_2)
    _, p_val_wilcoxon = sts.wilcoxon(data_group_1.sample(n), data_group_2.sample(n))
    features_dict['pairs'].append(f'{feature}')
    features_dict['p_value_ks'].append(p_val_ks)
    features_dict['ks_decision'].append('Не отклоняем H0' if p_val_ks >= 0.05 else 'Отклоняем H0')
    features_dict['p_value_wilcoxon'].append(p_val_wilcoxon)
    features_dict['wilcoxon_decision'].append('Не отклоняем H0' if p_val_wilcoxon >= 0.05 else 'Отклоняем H0')
    if p_val_ks <= 0.05 and p_val_wilcoxon <= 0.05:
        checked_features.append(feature)

features_df = pd.DataFrame(features_dict)
features_df
```

```
 16%|███████████                                                      | 3/19 [00:11<01:02,  3.90s/i
t]C:\Users\lavre\anaconda3\Lib\site-packages\scipy\stats\_morestats.py:3414: UserWarning:

Exact p-value calculation does not work if there are zeros. Switching to normal approximation.

100%|████████████████████████████████████████████████████████████████| 19/19 [00:41<00:00,  2.18s/i
t]
```

Out[37]:

	pairs	p_value_ks	ks_decision	p_value_wilcoxon	wilcoxon_decision
0	msno	0.000000e+00	Отклоняем H0	0.000000e+00	Отклоняем H0
1	song_id	1.153116e-204	Отклоняем H0	1.135856e-272	Отклоняем H0
2	source_system_tab	9.801088e-01	Не отклоняем H0	5.468750e-01	Не отклоняем H0
3	source_screen_name	1.992095e-01	Не отклоняем H0	2.598350e-01	Не отклоняем H0
4	source_type	5.360978e-01	Не отклоняем H0	1.098633e-01	Не отклоняем H0
5	song_length	1.058416e-71	Отклоняем H0	2.035700e-93	Отклоняем H0
6	genre_ids	2.505201e-02	Отклоняем H0	3.687659e-02	Отклоняем H0
7	artist_name	6.578931e-05	Отклоняем H0	2.650266e-05	Отклоняем H0
8	composer	3.126381e-46	Отклоняем H0	2.129279e-62	Отклоняем H0
9	lyricist	8.550692e-49	Отклоняем H0	6.769461e-56	Отклоняем H0
10	language	7.869298e-01	Не отклоняем H0	4.316406e-01	Не отклоняем H0
11	name	6.913280e-160	Отклоняем H0	1.312757e-185	Отклоняем H0
12	isrc	1.398328e-106	Отклоняем H0	1.560515e-155	Отклоняем H0
13	city	4.108850e-02	Отклоняем H0	7.598019e-02	Не отклоняем H0
14	bd	5.283286e-02	Не отклоняем H0	8.886356e-03	Отклоняем H0
15	gender	3.333333e-01	Не отклоняем H0	5.000000e-01	Не отклоняем H0
16	registered_via	2.380952e-01	Не отклоняем H0	1.250000e-01	Не отклоняем H0
17	registration_init_time	5.246276e-238	Отклоняем H0	1.243184e-277	Отклоняем H0
18	expiration_date	5.469900e-27	Отклоняем H0	1.673412e-17	Отклоняем H0

Из результатов проведенных тестов заметно, что некоторые характеристики в 2 датасетах имеют разные распределения, что может плохо сказаться на качестве предсказаний при включении их в модель.