## Обзор данных

Составьте первое представление о данных Яндекс.Музыки.

```
import pandas as pd # импорт библиотеки pandas
```

```
df = pd.read_csv('/datasets/yandex_music_project.csv') # чтение файла с данными и сохранение в df
```

```
df.head(10) # получение первых 10 строк таблицы df
```

| | userID | Track | artist | genre | City | time | Day |
|---|---|---|---|---|---|---|---|
| 0 | FFB692EC | Kamigata To Boots | The Mass Missile | rock | Saint-Petersburg | 20:28:33 | Wednesday |
| 1 | 55204538 | Delayed Because of Accident | Andreas Rönnberg | rock | Moscow | 14:07:09 | Friday |
| 2 | 20EC38 | Funiculì funiculà | Mario Lanza | pop | Saint-Petersburg | 20:58:07 | Wednesday |
| 3 | A3DD03C9 | Dragons in the Sunset | Fire + Ice | folk | Saint-Petersburg | 08:37:09 | Monday |
| 4 | E2DC1FAE | Soul People | Space Echo | dance | Moscow | 08:34:34 | Monday |
| 5 | 842029A1 | Преданная | IMPERVTOR | rusrap | Saint-Petersburg | 13:09:41 | Friday |
| 6 | 4CB90AA5 | True | Roman Messer | dance | Moscow | 13:00:07 | Wednesday |
| 7 | F03E1C1F | Feeling This Way | Polina Griffith | dance | Moscow | 20:47:49 | Wednesday |
| 8 | 8FA1D3BE | И вновь продолжается бой | NaN | ruspop | Moscow | 09:17:40 | Friday |
| 9 | E772D5C0 | Pessimist | NaN | dance | Saint-Petersburg | 21:20:49 | Wednesday |

```
df.info() # получение общей информации о данных в таблице df
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 65079 entries, 0 to 65078
Data columns (total 7 columns):
  userID    65079 non-null object
Track       63848 non-null object
artist      57876 non-null object
genre       63881 non-null object
  City      65079 non-null object
time        65079 non-null object
Day         65079 non-null object
dtypes: object(7)
memory usage: 3.5+ MB
```

Итак, в таблице семь столбцов. Тип данных во всех столбцах — `object`.

Согласно документации к данным:

- `userID` — идентификатор пользователя;
- `Track` — название трека;
- `artist` — имя исполнителя;
- `genre` — название жанра;
- `City` — город пользователя;
- `time` — время начала прослушивания;
- `Day` — день недели.

В названиях колонок видны три нарушения стиля:

1. Строчные буквы сочетаются с прописными.
2. Встречаются пробелы.
3. user_id необходимо записать в змеином регистре.

Количество значений в столбцах различается. Значит, в данных есть пропущенные значения.

**Выводы**

В каждой строке таблицы — данные о прослушанном треке. Часть колонок описывает саму композицию: название, исполнителя и жанр. Остальные данные рассказывают о пользователе: из какого он города, когда он слушал музыку.

Предварительно можно утверждать, что, данных достаточно для проверки гипотез. Но встречаются пропуски в данных, а в названиях колонок — расхождения с хорошим стилем.

Чтобы двигаться дальше, нужно устранить проблемы в данных.

## Предобработка данных

Исправьте стиль в заголовках столбцов, исключите пропуски. Затем проверьте данные на дубликаты.

### Стиль заголовков

```
df.columns # перечень названий столбцов таблицы df
```

```
Index(['  userID', 'Track', 'artist', 'genre', '  City  ', 'time', 'Day'], dtype='object')
```

```
df = df.rename(columns={'  userID':'user_id', 'Track':'track', '  City  ':'city', 'Day':'day'})# переименование столбцов
```

```
df.columns # проверка результатов - перечень названий столбцов
```

```
Index(['user_id', 'track', 'artist', 'genre', 'city', 'time', 'day'], dtype='object')
```

### Пропуски значений

```
df.isna().sum() # подсчёт пропусков
```

```
user_id       0
```

```
track      1231
artist     7203
genre      1198
city          0
time          0
day           0
dtype: int64
```

```
columns_to_replace = ['track', 'artist', 'genre']
for column in columns_to_replace:
    df[column] = df[column].fillna('unknown')
    # перебор названий столбцов в цикле и замена пропущенных значений на 'unknown'
```

```
df.isna().sum() # подсчёт пропусков
```

```
user_id    0
track      0
artist     0
genre      0
city       0
time       0
day        0
dtype: int64
```

## Дубликаты

```
df.duplicated().sum() # подсчёт явных дубликатов
```

```
3826
```

```
df = df.drop_duplicates().reset_index(drop=True) # удаление явных дубликатов (с удалением старых индексов и формированием новых)
```

```
df.duplicated().sum() # проверка на отсутствие дубликатов
```

```
0
```

```
df['genre'].sort_values().unique() # Просмотр уникальных названий жанров
```

```
array(['acid', 'acoustic', 'action', 'adult', 'africa', 'afrikaans',
       'alternative', 'alternativepunk', 'ambient', 'americana',
       'animated', 'anime', 'arabesk', 'arabic', 'arena',
       'argentinetango', 'art', 'audiobook', 'author', 'avantgarde',
       'axé', 'baile', 'balkan', 'beats', 'bigroom', 'black', 'bluegrass',
       'blues', 'bollywood', 'bossa', 'brazilian', 'breakbeat', 'breaks',
       'broadway', 'cantautori', 'cantopop', 'canzone', 'caribbean',
       'caucasian', 'celtic', 'chamber', 'chanson', 'children', 'chill',
       'chinese', 'choral', 'christian', 'christmas', 'classical',
       'classicmetal', 'club', 'colombian', 'comedy', 'conjazz',
       'contemporary', 'country', 'cuban', 'dance', 'dancehall',
       'dancepop', 'dark', 'death', 'deep', 'deutschrock', 'deutschspr',
       'dirty', 'disco', 'dnb', 'documentary', 'downbeat', 'downtempo',
       'drum', 'dub', 'dubstep', 'eastern', 'easy', 'electronic',
       'electropop', 'emo', 'entehno', 'epicmetal', 'estrada', 'ethnic',
       'eurofolk', 'european', 'experimental', 'extrememetal', 'fado',
       'fairytail', 'film', 'fitness', 'flamenco', 'folk', 'folklore',
       'folkmetal', 'folkrock', 'folktronica', 'forró', 'frankreich',
       'französisch', 'french', 'funk', 'future', 'gangsta', 'garage',
       'german', 'ghazal', 'gitarre', 'glitch', 'gospel', 'gothic',
       'grime', 'grunge', 'gypsy', 'handsup', "hard'n'heavy", 'hardcore',
       'hardstyle', 'hardtechno', 'hip', 'hip-hop', 'hiphop',
       'historisch', 'holiday', 'hop', 'horror', 'house', 'hymn', 'idm',
       'independent', 'indian', 'indie', 'indipop', 'industrial',
       'inspirational', 'instrumental', 'international', 'irish', 'jam',
       'japanese', 'jazz', 'jewish', 'jpop', 'jungle', 'k-pop',
       'karadeniz', 'karaoke', 'kayokyoku', 'korean', 'laiko', 'latin',
       'latino', 'leftfield', 'local', 'lounge', 'loungeelectronic',
       'lovers', 'malaysian', 'mandopop', 'marschmusik', 'meditative',
       'mediterranean', 'melodic', 'metal', 'metalcore', 'mexican',
       'middle', 'minimal', 'miscellaneous', 'modern', 'mood', 'mpb',
       'muslim', 'native', 'neoklassik', 'neue', 'new', 'newage',
       'newwave', 'nu', 'nujazz', 'numetal', 'oceania', 'old', 'opera',
       'orchestral', 'other', 'piano', 'podcasts', 'pop', 'popdance',
       'popelectronic', 'popeurodance', 'poprussian', 'post',
       'posthardcore', 'postrock', 'power', 'progmetal', 'progressive',
```

```
       'psychedelic', 'punjabi', 'punk', 'quebecois', 'ragga', 'ram',
       'rancheras', 'rap', 'rave', 'reggae', 'reggaeton', 'regional',
       'relax', 'religious', 'retro', 'rhythm', 'rnb', 'rnr', 'rock',
       'rockabilly', 'rockalternative', 'rockindie', 'rockother',
       'romance', 'roots', 'ruspop', 'rusrap', 'rusrock', 'russian',
       'salsa', 'samba', 'scenic', 'schlager', 'self', 'sertanejo',
       'shanson', 'shoegazing', 'showtunes', 'singer', 'ska', 'skarock',
       'slow', 'smooth', 'soft', 'soul', 'soulful', 'sound', 'soundtrack',
       'southern', 'specialty', 'speech', 'spiritual', 'sport',
       'stonerrock', 'surf', 'swing', 'synthpop', 'synthrock',
       'sängerportrait', 'tango', 'tanzorchester', 'taraftar', 'tatar',
       'tech', 'techno', 'teen', 'thrash', 'top', 'traditional',
       'tradjazz', 'trance', 'tribal', 'trip', 'triphop', 'tropical',
       'türk', 'türkçe', 'ukrrock', 'unknown', 'urban', 'uzbek',
       'variété', 'vi', 'videogame', 'vocal', 'western', 'world',
       'worldbeat', 'ïîï', 'электроника'], dtype=object)
```

```
def replace_wrong_genres(wrong_genres, correct_genre):
    for wrong_genre in wrong_genres:
        df['genre'] = df['genre'].replace(wrong_genres, correct_genre) # Функция для замены неявных дубликатов
```

```
duplicates = ['hip', 'hop', 'hip-hop']
name = 'hiphop'
replace_wrong_genres(duplicates, name) # Устранение неявных дубликатов
```

```
df['genre'].sort_values().unique() # Проверка на неявные дубликаты
```

```
array(['acid', 'acoustic', 'action', 'adult', 'africa', 'afrikaans',
       'alternative', 'alternativepunk', 'ambient', 'americana',
       'animated', 'anime', 'arabesk', 'arabic', 'arena',
       'argentinetango', 'art', 'audiobook', 'author', 'avantgarde',
       'axé', 'baile', 'balkan', 'beats', 'bigroom', 'black', 'bluegrass',
       'blues', 'bollywood', 'bossa', 'brazilian', 'breakbeat', 'breaks',
       'broadway', 'cantautori', 'cantopop', 'canzone', 'caribbean',
       'caucasian', 'celtic', 'chamber', 'chanson', 'children', 'chill',
       'chinese', 'choral', 'christian', 'christmas', 'classical',
       'classicmetal', 'club', 'colombian', 'comedy', 'conjazz',
       'contemporary', 'country', 'cuban', 'dance', 'dancehall',
```

```
'dancepop', 'dark', 'death', 'deep', 'deutschrock', 'deutschspr',
'dirty', 'disco', 'dnb', 'documentary', 'downbeat', 'downtempo',
'drum', 'dub', 'dubstep', 'eastern', 'easy', 'electronic',
'electropop', 'emo', 'entehno', 'epicmetal', 'estrada', 'ethnic',
'eurofolk', 'european', 'experimental', 'extrememetal', 'fado',
'fairytail', 'film', 'fitness', 'flamenco', 'folk', 'folklore',
'folkmetal', 'folkrock', 'folktronica', 'forró', 'frankreich',
'französisch', 'french', 'funk', 'future', 'gangsta', 'garage',
'german', 'ghazal', 'gitarre', 'glitch', 'gospel', 'gothic',
'grime', 'grunge', 'gypsy', 'handsup', "hard'n'heavy", 'hardcore',
'hardstyle', 'hardtechno', 'hiphop', 'historisch', 'holiday',
'horror', 'house', 'hymn', 'idm', 'independent', 'indian', 'indie',
'indipop', 'industrial', 'inspirational', 'instrumental',
'international', 'irish', 'jam', 'japanese', 'jazz', 'jewish',
'jpop', 'jungle', 'k-pop', 'karadeniz', 'karaoke', 'kayokyoku',
'korean', 'laiko', 'latin', 'latino', 'leftfield', 'local',
'lounge', 'loungeelectronic', 'lovers', 'malaysian', 'mandopop',
'marschmusik', 'meditative', 'mediterranean', 'melodic', 'metal',
'metalcore', 'mexican', 'middle', 'minimal', 'miscellaneous',
'modern', 'mood', 'mpb', 'muslim', 'native', 'neoklassik', 'neue',
'new', 'newage', 'newwave', 'nu', 'nujazz', 'numetal', 'oceania',
'old', 'opera', 'orchestral', 'other', 'piano', 'podcasts', 'pop',
'popdance', 'popelectronic', 'popeurodance', 'poprussian', 'post',
'posthardcore', 'postrock', 'power', 'progmetal', 'progressive',
'psychedelic', 'punjabi', 'punk', 'quebecois', 'ragga', 'ram',
'rancheras', 'rap', 'rave', 'reggae', 'reggaeton', 'regional',
'relax', 'religious', 'retro', 'rhythm', 'rnb', 'rnr', 'rock',
'rockabilly', 'rockalternative', 'rockindie', 'rockother',
'romance', 'roots', 'ruspop', 'rusrap', 'rusrock', 'russian',
'salsa', 'samba', 'scenic', 'schlager', 'self', 'sertanejo',
'shanson', 'shoegazing', 'showtunes', 'singer', 'ska', 'skarock',
'slow', 'smooth', 'soft', 'soul', 'soulful', 'sound', 'soundtrack',
'southern', 'specialty', 'speech', 'spiritual', 'sport',
'stonerrock', 'surf', 'swing', 'synthpop', 'synthrock',
'sängerportrait', 'tango', 'tanzorchester', 'taraftar', 'tatar',
'tech', 'techno', 'teen', 'thrash', 'top', 'traditional',
'tradjazz', 'trance', 'tribal', 'trip', 'triphop', 'tropical',
'türk', 'türkçe', 'ukrrock', 'unknown', 'urban', 'uzbek',
'variété', 'vi', 'videogame', 'vocal', 'western', 'world',
'worldbeat', 'ïîï', 'электроника'], dtype=object)
```

**Выводы**

Предобработка обнаружила три проблемы в данных:

- нарушения в стиле заголовков,
- пропущенные значения,
- дубликаты — явные и неявные.

Были исправлены заголовки, чтобы упростить работу с таблицей.

Пропущенные значения заменены на `'unknown'`. Ещё предстоит увидеть, не повредят ли исследованию пропуски в колонке `genre`.

Теперь можно перейти к проверке гипотез.

# Проверка гипотез

## Сравнение поведения пользователей двух столиц

Первая гипотеза утверждает, что пользователи по-разному слушают музыку в Москве и Санкт-Петербурге. Проверим это предположение по данным о трёх днях недели — понедельнике, среде и пятнице. Для этого:

- Разделим пользователей Москвы и Санкт-Петербурга
- Сравним, сколько треков послушала каждая группа пользователей в понедельник, среду и пятницу.

Для тренировки сначала выполните каждый из расчётов по отдельности.

Оцените активность пользователей в каждом городе. Сгруппируйте данные по городу и посчитайте прослушивания в каждой группе.

```
df.groupby('day')['user_id'].count() # Подсчёт прослушиваний в каждом городе
```

```
day
Friday       21840
Monday       21354
Wednesday    18059
Name: user_id, dtype: int64
```

В Москве прослушиваний больше, чем в Петербурге. Из этого не следует, что московские пользователи чаще слушают музыку. Просто самих пользователей в Москве больше.

Теперь сгруппируем данные по дню недели и подсчитаем прослушивания в понедельник, среду и пятницу.

```
df.groupby('day')['user_id'].count()
# Подсчёт прослушиваний в каждый из трёх дней
```

```
day
Friday       21840
Monday       21354
Wednesday    18059
Name: user_id, dtype: int64
```

В среднем пользователи из двух городов менее активны по средам. Но картина может измениться, если рассмотреть каждый город в отдельности.

```
def number_tracks(day, city):
    track_list = df[(df['day'] == day) & (df['city'] == city)]
    track_list_count = track_list['user_id'].count()
    return track_list_count # <создание функции number_tracks()>
# Объявляется функция с двумя параметрами: day, city.
# В переменной track_list сохраняются те строки таблицы df, для которых
# значение в столбце 'day' равно параметру day и одновременно значение
# в столбце 'city' равно параметру city (используйте последовательную фильтрацию
# с помощью логической индексации).
# В переменной track_list_count сохраняется число значений столбца 'user_id',
# рассчитанное методом count() для таблицы track_list.
```

```
# Функция возвращает число - значение track_list_count.

# Функция для подсчёта прослушиваний для конкретного города и дня.
# С помощью последовательной фильтрации с логической индексацией она
# сначала получит из исходной таблицы строки с нужным днём,
# затем из результата отфильтрует строки с нужным городом,
# методом count() посчитает количество значений в колонке user_id.
# Это количество функция вернёт в качестве результата
```

Вызовем `number_tracks()` шесть раз, меняя значение параметров — так, чтобы получить данные для каждого города в каждый из трёх дней.

```
day = 'Monday'
city = 'Moscow'
number_tracks(day, city) # количество прослушиваний в Москве по понедельникам
```

```
15740
```

```
day = 'Monday'
city = 'Saint-Petersburg'
number_tracks(day, city) # количество прослушиваний в Санкт-Петербурге по понедельникам
```

```
5614
```

```
day = 'Wednesday'
city = 'Moscow'
number_tracks(day, city) # количество прослушиваний в Москве по средам
```

```
11056
```

```
day = 'Wednesday'
city = 'Saint-Petersburg'
number_tracks(day, city) # количество прослушиваний в Санкт-Петербурге по средам
```

```
7003
```

```
day = 'Friday'
city = 'Moscow'
number_tracks(day, city) # количество прослушиваний в Москве по пятницам
```

```
15945
```

```
day = 'Friday'
city = 'Saint-Petersburg'
number_tracks(day, city) # количество прослушиваний в Санкт-Петербурге по пятницам
```

```
5895
```

Создаем с помощью конструктора `pd.DataFrame` таблицу, где

- названия колонок — `['city', 'monday', 'wednesday', 'friday']`;
- данные — результаты, которые мы получили с помощью `number_tracks`.

```
columns = ['city', 'monday', 'wednesday', 'friday']
data = [['Moscow', 15740, 11056, 15945],
       ['Saint-Petersburg', 5614, 7003, 5895]]
df_day_city = pd.DataFrame(data=data, columns=columns)
display(df_day_city) # Таблица с результатами
```

| | city | monday | wednesday | friday |
|---|---|---|---|---|
| 0 | Moscow | 15740 | 11056 | 15945 |
| 1 | Saint-Petersburg | 5614 | 7003 | 5895 |

**Выводы**

Данные показывают разницу поведения пользователей:

- В Москве пик прослушиваний приходится на понедельник и пятницу, а в среду заметен спад.
- В Петербурге, наоборот, больше слушают музыку по средам. Активность в понедельник и пятницу здесь почти в равной мере уступает среде.

Значит, данные говорят в пользу первой гипотезы.

## Музыка в начале и в конце недели

Согласно второй гипотезе, утром в понедельник в Москве преобладают одни жанры, а в Петербурге — другие. Так же и вечером пятницы преобладают разные жанры — в зависимости от города.

Сохраним таблицы с данными в две переменные:

- по Москве — в `moscow_general`;
- по Санкт-Петербургу — в `spb_general`.

```
moscow_general = df[df['city'] == 'Moscow']
# получение таблицы moscow_general из тех строк таблицы df,
# для которых значение в столбце 'city' равно 'Moscow'
```

```
spb_general = df[df['city'] == 'Saint-Petersburg']
# получение таблицы spb_general из тех строк таблицы df,
# для которых значение в столбце 'city' равно 'Saint-Petersburg'
```

Создаем функцию `genre_weekday()` с четырьмя параметрами:

- таблица (датафрейм) с данными,
- день недели,

- начальная временная метка в формате 'hh:mm',
- последняя временная метка в формате 'hh:mm'.

Функция должна вернуть информацию о топ-10 жанров тех треков, которые прослушивали в указанный день, в промежутке между двумя отметками времени.

```
def genre_weekday(table, day, time1, time2):
    genre_df = table[(table['day'] == day) & (table['time'] > time1) & (table['time'] < time2)]
    genre_df_count = genre_df.groupby('genre')['day'].count()
    genre_df_sorted = genre_df_count.sort_values(ascending=False)
    return genre_df_sorted.head(10)
    # Объявление функции genre_weekday() с параметрами table, day, time1, time2,
# которая возвращает информацию о самых популярных жанрах в указанный день в
# заданное время:
# 1) в переменную genre_df сохраняются те строки переданного датафрейма table, для
#    которых одновременно:
#    - значение в столбце day равно значению аргумента day
#    - значение в столбце time больше значения аргумента time1
#    - значение в столбце time меньше значения аргумента time2
#    Используем последовательную фильтрацию с помощью логической индексации.
# 2) сгруппировать датафрейм genre_df по столбцу genre, взять один из его
#    столбцов и посчитать методом count() количество записей для каждого из
#    присутствующих жанров, получившийся Series записать в переменную
#    genre_df_count
# 3) отсортировать genre_df_count по убыванию встречаемости и сохранить
#    в переменную genre_df_sorted
# 4) вернуть Series из 10 первых значений genre_df_sorted, это будут топ-10
#    популярных жанров (в указанный день, в заданное время)
```

Сравним результаты функции `genre_weekday()` для Москвы и Санкт-Петербурга в понедельник утром (с 7:00 до 11:00) и в пятницу вечером (с 17:00 до 23:00):

```
genre_weekday(moscow_general, 'Monday', '07:00', '11:00') # вызов функции для утра понедельника в Москве (вместо df — таблица moscow_
# объекты, хранящие время, являются строками и сравниваются как строки
```

```
# пример вызова: genre_weekday(moscow_general, 'Monday', '07:00', '11:00')
```

```
genre
pop            781
dance          549
electronic     480
rock           474
hiphop         286
ruspop         186
world          181
rusrap         175
alternative    164
unknown        161
Name: day, dtype: int64
```

```
genre_weekday(spb_general, 'Monday', '07:00', '11:00') # вызов функции для утра понедельника в Петербурге (вместо df — таблица spb_ge
```

```
genre
pop            218
dance          182
rock           162
electronic     147
hiphop          80
ruspop          64
alternative     58
rusrap          55
jazz            44
classical       40
Name: day, dtype: int64
```

```
genre_weekday(moscow_general, 'Friday', '17:00', '23:00') # вызов функции для вечера пятницы в Москве
```

```
genre
pop            713
rock           517
dance          495
electronic     482
hiphop         273
world          208
```

```
ruspop         170
alternative    163
classical      163
rusrap         142
Name: day, dtype: int64
```

```
genre_weekday(spb_general, 'Friday', '17:00', '23:00') # вызов функции для вечера пятницы в Петербурге
```

```
genre
pop            256
rock           216
electronic     216
dance          210
hiphop          97
alternative     63
jazz            61
classical       60
rusrap          59
world           54
Name: day, dtype: int64
```

**Выводы**

Если сравнить топ-10 жанров в понедельник утром, можно сделать такие выводы:

1. В Москве и Петербурге слушают похожую музыку. Единственное отличие — в московский рейтинг вошёл жанр “world”, а в петербургский — джаз и классика.
2. В Москве пропущенных значений оказалось так много, что значение `'unknown'` заняло десятое место среди самых популярных жанров. Значит, пропущенные значения занимают существенную долю в данных и угрожают достоверности исследования.

Вечер пятницы не меняет эту картину. Некоторые жанры поднимаются немного выше, другие спускаются, но в целом топ-10 остаётся тем же самым.

Таким образом, вторая гипотеза подтвердилась лишь частично:

- Пользователи слушают похожую музыку в начале недели и в конце.
- Разница между Москвой и Петербургом не слишком выражена. В Москве чаще слушают русскую популярную музыку, в Петербурге — джаз.

Однако пропуски в данных ставят под сомнение этот результат. В Москве их так много, что рейтинг топ-10 мог бы выглядеть иначе, если бы не утерянные данные о жанрах.

## Жанровые предпочтения в Москве и Петербурге

Гипотеза: Петербург — столица рэпа, музыку этого жанра там слушают чаще, чем в Москве. А Москва — город контрастов, в котором, тем не менее, преобладает поп-музыка.

Сгруппируем таблицу `moscow_general` по жанру и посчитайте прослушивания треков каждого жанра методом `count()`. Затем отсортируйте результат в порядке убывания и сохраните его в таблице `moscow_genres`.

```
moscow_genres = moscow_general.groupby('genre')['track'].count().sort_values(ascending=False) # одной строкой: группировка таблицы mo
# подсчёт числа значений 'genre' в этой группировке методом count(),
# сортировка получившегося Series в порядке убывания и сохранение в moscow_genres
```

Выведем на экран первые десять строк `moscow_genres`:

```
display(moscow_genres.head(10)) # просмотр первых 10 строк moscow_genres
```

```
genre
pop            5892
dance          4435
```

Теперь повторим то же и для Петербурга.

Сгруппируем таблицу `spb_general` по жанру. Посчитаем прослушивания треков каждого жанра. Результат отсортируем в порядке убывания и сохраним в таблице `spb_genres`:

```
ruspop         1372
```

```
spb_genres = spb_general.groupby('genre')['track'].count().sort_values(ascending=False) # одной строкой: группировка таблицы spb_gene
# подсчёт числа значений 'genre' в этой группировке методом count(),
# сортировка получившегося Series в порядке убывания и сохранение в spb_genres
```

Выведем на экран первые десять строк `spb_genres`:

```
display(spb_genres.head(10)) # просмотр первых 10 строк spb_genres
```

```
genre
pop            2431
dance          1932
rock           1879
electronic     1736
hiphop          960
alternative     649
classical       646
rusrap          564
ruspop          538
world           515
Name: track, dtype: int64
```

**Выводы**

Гипотеза частично подтвердилась:

- Поп-музыка — самый популярный жанр в Москве, как и предполагала гипотеза. Более того, в топ-10 жанров встречается близкий жанр — русская популярная музыка.
- Вопреки ожиданиям, рэп одинаково популярен в Москве и Петербурге.

## Итоги исследования

Мы проверили три гипотезы и установили:

1. День недели по-разному влияет на активность пользователей в Москве и Петербурге.

Первая гипотеза полностью подтвердилась.

2. Музыкальные предпочтения не сильно меняются в течение недели — будь то Москва или Петербург. Небольшие различия заметны в начале недели, по понедельникам:

- в Москве слушают музыку жанра “world”,
- в Петербурге — джаз и классику.

Таким образом, вторая гипотеза подтвердилась лишь отчасти. Этот результат мог оказаться иным, если бы не пропуски в данных.

3. Во вкусах пользователей Москвы и Петербурга больше общего чем различий. Вопреки ожиданиям, предпочтения жанров в Петербурге напоминают московские.

Третья гипотеза не подтвердилась. Если различия в предпочтениях и существуют, на основной массе пользователей они незаметны.