Яндекс Музыка

Сравнение Москвы и Петербурга окружено мифами. Например:

- Москва — мегаполис, подчинённый жёсткому ритму рабочей недели;
- Петербург — культурная столица, со своими вкусами.

Цель исследования проверить три гипотезы:

1. Активность пользователей зависит от дня недели. Причём в Москве и Петербурге это проявляется по-разному.
2. В понедельник утром в Москве преобладают одни жанры, а в Петербурге — другие. Так же и вечером пятницы преобладают разные жанры — в зависимости от города.
3. Москва и Петербург предпочитают разные жанры музыки. В Москве чаще слушают поп-музыку, в Петербурге — русский рэп.

Ход исследования

1. Обзор данных.
2. Предобработка данных.
3. Проверка гипотез.

Источник данных:

файл `yandex_music_project.csv`

1 Обзор данных

In [1]:
```
#импорт библиотеки pandas
import pandas as pd
```

Прочитаем файл `yandex_music_project.csv` из папки `/datasets` и сохраним его в переменной `df`:

In [2]:
```
# чтение файла с данными и сохранение в df
df = pd.read_csv('/datasets/yandex_music_project.csv')
```

In [3]:
```
# получим первые 10 строк таблицы df
df.head(10)
```

Out[3]:

	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

In [4]:
```
# получим общую информацию о данных в таблице df
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65079 entries, 0 to 65078
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0     userID  65079 non-null  object
 1   Track     63848 non-null  object
 2   artist    57876 non-null  object
 3   genre     63881 non-null  object
 4     City    65079 non-null  object
 5   time      65079 non-null  object
 6   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.

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

В названиях колонок видны нарушения стиля:

- Строчные буквы сочетаются с прописными.
- Встречаются пробелы.

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.

2 Предобработка данных

2.1 Стиль заголовков

In [5]:
```
# перечень названий столбцов таблицы df
df.columns
```

Out[5]:
```
Index(['  userID', 'Track', 'artist', 'genre', '  City  ', 'time', 'Day'], dtype='object')
```

Приведем названия в соответствие с хорошим стилем:

- несколько слов в названии запишите в «змеином_регистре»,
- все символы сделайте строчными,
- устраните пробелы.

Для этого переименуем колонки так:

- `'  userID'` → `'user_id'`;
- `'Track'` → `'track'`;
- `'  City  '` → `'city'`;
- `'Day'` → `'day'`.

In [6]:
```
# переименование столбцов
df = df.rename(columns={'  userID':'user_id', 'Track':'track', '  City  ':'city', 'Day':'day'})
```

In [7]:
```
# проверка результатов - перечень названий столбцов
df.columns
```

Out[7]:
```
Index(['user_id', 'track', 'artist', 'genre', 'city', 'time', 'day'], dtype='object')
```

2.2 Пропуски значений

In [8]:
```
# подсчёт пропусков
df.isna().sum()
```

Out[8]:
```
user_id       0
track      1231
artist     7203
genre      1198
city          0
time          0
day           0
dtype: int64
```

Не все пропущенные значения влияют на исследование. Так в `track` и `artist` пропуски не важны для работы. Достаточно заменить их явными обозначениями.

Но пропуски в `genre` могут помешать сравнению музыкальных вкусов в Москве и Санкт-Петербурге. На практике было бы правильно установить причину пропусков и восстановить данные. Такой возможности нет в учебном проекте. Придётся:

- заполнить и эти пропуски явными обозначениями;
- оценить, насколько они повредят расчётам.

In [9]:
```
# перебор названий столбцов в цикле и замена пропущенных значений на 'unknown'
columns_to_replace = ['track', 'artist', 'genre']
for column in columns_to_replace:
    df[column] = df[column].fillna('unknown')
```

In [10]:
```
# проверка результатов замены пропусков
df.isna().sum()
```

Out[10]:
```
user_id    0
track      0
artist     0
genre      0
city       0
time       0
day        0
dtype: int64
```

2.3 Дубликаты

In [11]:
```
# подсчёт явных дубликатов
df.duplicated().sum()
```

Out[11]:
```
3826
```

In [12]:
```
# удаление явных дубликатов
df = df.drop_duplicates()
```

In [13]:
```
# проверка результатов удаления явных дубликатов
df.duplicated().sum()
```

Out[13]:
```
0
```

Теперь избавимся от неявных дубликатов в колонке `genre` . Например, название одного и того же жанра может быть записано немного по-разному. Такие ошибки тоже повлияют на результат исследования.

Выведем на экран список уникальных названий жанров, отсортированный в алфавитном порядке. Для этого:

1. извлечем нужный столбец датафрейма;
2. применим к нему метод сортировки;
3. для отсортированного столбца вызовем метод, который вернёт уникальные значения из столбца.

In [14]:
```
# Просмотр уникальных названий жанров
df['genre'].sort_values().unique()
```

Out[14]:
```
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)
```

Просмотрим список и найдем неявные дубликаты - названия hiphop. Это могут быть названия с ошибками или альтернативные названия того же жанра.

Видны следующие неявные дубликаты:

hip, hop, hip-hop. Чтобы очистить от них таблицу используем метод replace() с двумя аргументами: списком строк-дубликатов (включающий hip, hop и hip-hop) и строкой с правильным значением. Нужно исправить колонку

In [15]:
```
# Устранение неявных дубликатов
df['genre'] = df['genre'].replace(['hip', 'hop', 'hip-hop'], 'hiphop')
```

Проверим, что заменили неправильные названия:

- hip,
- hop,
- hip-hop.

Выведем отсортированный список уникальных значений столбца `genre` :

In [16]:
```
# Проверка на неявные дубликаты
df['genre'].sort_values().unique()
```

Out[16]:
```
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', 'ïïï', 'электроника'], dtype=object)
```

Выводы

Предобработка обнаружила три проблемы в данных:

- нарушения в стиле заголовков,
- пропущенные значения,

- дубликаты — явные и неявные.

Мы исправили заголовки, чтобы упростить работу с таблицей. Без дубликатов исследование станет более точным.

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

Теперь можно перейти к проверке гипотез.

3 Проверка гипотез

3.1 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.

Оценим активность пользователей в каждом городе. Сгруппируем данные по городу и посчитаем прослушивания в каждой группе.

In [17]:
```
# Подсчёт прослушиваний в каждом городе
df.groupby('city')['genre'].count()
```

Out[17]:
```
city
Moscow              42741
Saint-Petersburg    18512
Name: genre, 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.

Сгруппируем данные по дню недели и посчитаем прослушивания в понедельник, среду и пятницу. В данных есть информация о прослушиваниях только за эти дни.

In [18]:
```
# Подсчёт прослушиваний в каждый из трёх дней
df.groupby(['day', 'city'])['genre'].count()
```

Out[18]:
```
day        city
Friday     Moscow              15945
           Saint-Petersburg     5895
Monday     Moscow              15740
           Saint-Petersburg     5614
Wednesday  Moscow              11056
           Saint-Petersburg     7003
Name: genre, dtype: int64
```

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

Напишем функцию, которая объединит два расчёта.

Создадим функцию `number_tracks()`, которая посчитает прослушивания для заданного дня и города. Ей понадобятся два параметра:

- день недели,
- название города.

В функции сохраним в переменную строки исходной таблицы, у которых значение:

- в колонке `day` равно параметру `day`,

- в колонке `city` равно параметру `city`.

Для этого применим последовательную фильтрацию с логической индексацией.

Затем посчитаем значения в столбце `user_id` получившейся таблицы. Результат сохраним в новую переменную. Вернем эту переменную из функции.

In [19]:
```
# <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 или сложные логические выражения в одну строку, если вы уже знакомы с ними).
# 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

def number_tracks(day, city):
    track_list = df[df['day']==day]
    track_list = track_list[track_list['city']==city]
    track_list_count = track_list['user_id'].count()
    return track_list_count
```

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

In [20]:
```
# количество прослушиваний в Москве по понедельникам
number_tracks('Monday', 'Moscow')
```

Out[20]: 15740

In [21]:
```
# количество прослушиваний в Санкт-Петербурге по понедельникам
number_tracks('Monday', 'Saint-Petersburg')
```

Out[21]: 5614

In [22]:
```
# количество прослушиваний в Москве по средам
number_tracks('Wednesday','Moscow')
```

Out[22]: 11056

In [23]:
```
# количество прослушиваний в Санкт-Петербурге по средам
number_tracks('Wednesday','Saint-Petersburg')
```

Out[23]: 7003

In [24]:
```
# количество прослушиваний в Москве по пятницам
number_tracks('Friday','Moscow')
```

Out[24]: 15945

In [25]:
```
# количество прослушиваний в Санкт-Петербурге по пятницам
number_tracks('Friday','Saint-Petersburg')
```

Out[25]:
```
5895
```

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

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

In [26]:
```
mber_tracks('Monday', 'Saint-Petersburg'), number_tracks('Wednesday','Saint-Petersburg'), number_tracks('Friday','Saint-Petersburg')]], columns = ['city', 'monday', 'wednesday', 'friday'])
```

Out[26]:

	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.

3.2 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`.

In [27]:
```
# получение таблицы moscow_general из тех строк таблицы df,
# для которых значение в столбце 'city' равно 'Moscow'
moscow_general = df[df['city']=='Moscow']
```

In [28]:
```
# получение таблицы spb_general из тех строк таблицы df,
# для которых значение в столбце 'city' равно 'Saint-Petersburg'
spb_general = df[df['city']=='Saint-Petersburg']
```

Создадим 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.

In [29]:
```
# 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
#    Используйте 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)

def genre_weekday(df, day, time1, time2):
    # последовательная фильтрация
    # оставляем в genre_df только те строки df, у которых день равен day
    genre_df = df[df['day']==day]# ваш код здесь
    # оставляем в genre_df только те строки genre_df, у которых время меньше time2
    genre_df = genre_df[genre_df['time']<time2]# ваш код здесь
    # оставляем в genre_df только те строки genre_df, у которых время больше time1
    genre_df = genre_df[genre_df['time']>time1]# ваш код здесь
    # сгруппируем отфильтрованный датафрейм по столбцу с названиями жанров, возьмём столбец genre и посчитаем кол-во строк для каждого жанра методом count()
    genre_df_grouped = genre_df.groupby('genre')['genre'].count()# ваш код здесь
    # отсортируем результат по убыванию (чтобы в начале Series оказались самые популярные жанры)
    genre_df_sorted = genre_df_grouped.sort_values(ascending=False)# ваш код здесь
    # вернём Series с 10 самыми популярными жанрами в указанный отрезок времени заданного дня
    return genre_df_sorted[:10]
```

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):

In [30]:
```
# вызов функции для утра понедельника в Москве (вместо df — таблица moscow_general)
# объекты, хранящие время, являются строками и сравниваются как строки
# пример вызова: genre_weekday(moscow_general, 'Monday', '07:00', '11:00')
genre_weekday(moscow_general, 'Monday', '07:00', '11:00')
```

Out[30]:
```
genre
pop            781
dance          549
electronic     480
rock           474
hiphop         286
ruspop         186
world          181
rusrap         175
alternative    164
unknown        161
Name: genre, dtype: int64
```

In [31]:
```
# вызов функции для утра понедельника в Петербурге (вместо df — таблица spb_general)
genre_weekday(spb_general, 'Monday', '07:00', '11:00')
```

Out[31]:
```
genre
pop            218
dance          182
rock           162
electronic     147
hiphop          80
ruspop          64
alternative     58
rusrap          55
jazz            44
classical       40
Name: genre, dtype: int64
```

In [32]:
```
# вызов функции для вечера пятницы в Москве
genre_weekday(moscow_general, 'Friday', '17:00', '23:00')
```

Out[32]:
```
genre
pop            713
rock           517
dance          495
electronic     482
hiphop         273
world          208
ruspop         170
alternative    163
classical      163
rusrap         142
Name: genre, dtype: int64
```

In [33]:
```
# вызов функции для вечера пятницы в Петербурге
genre_weekday(spb_general, 'Friday', '17:00', '23:00')
```

Out[33]:
```
genre
pop            256
electronic     216
rock           216
dance          210
hiphop          97
alternative     63
jazz            61
classical       60
rusrap          59
world           54
Name: genre, 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.

3.3 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`.

In [34]:
```
# одной строкой: группировка таблицы moscow_general по столбцу 'genre',
# подсчёт числа значений 'genre' в этой группировке методом count(),
# сортировка получившегося Series в порядке убывания и сохранение в moscow_genres
moscow_genres = moscow_general.groupby('genre')['genre'].count().sort_values(ascending=False)
```

In [35]:
```
# просмотр первых 10 строк moscow_genres
moscow_genres[:10]
```

Out[35]:
```
genre
pop            5892
dance          4435
rock           3965
electronic     3786
hiphop         2096
classical      1616
world          1432
alternative    1379
ruspop         1372
rusrap         1161
Name: genre, dtype: int64
```

Теперь повторим то же и для Петербурга.

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

In [36]:
```
# одной строкой: группировка таблицы spb_general по столбцу 'genre',
# подсчёт числа значений 'genre' в этой группировке методом count(),
# сортировка получившегося Series в порядке убывания и сохранение в spb_genres
spb_genres = spb_general.groupby('genre')['genre'].count().sort_values(ascending=False)
```

In [37]:
```
# просмотр первых 10 строк spb_genres
spb_genres[:10]
```

Out[37]:
```
genre
pop            2431
dance          1932
rock           1879
electronic     1736
hiphop          960
alternative     649
classical       646
rusrap          564
ruspop          538
world           515
Name: genre, dtype: int64
```

Вывод:

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.

4 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.

На практике исследования содержат проверки статистических гипотез. Из данных одного сервиса не всегда можно сделать вывод о всех жителях города. Проверки статистических гипотез покажут, насколько они достоверны, исходя из имеющихся данных.