Исследование данных сервиса “Яндекс.Музыка” — сравнение пользователей пользователей двух городов

Описание задачи проекта: на реальных данных Яндекс.Музыки с помощью библиотеки Pandas и её возможностей проверить данные и сравнить поведение и предпочтения пользователей двух столиц — Москвы и Санкт-Петербурга.

Цели исследования: проверить три гипотезы:

1. Активность пользователей зависит от дня недели (в выборке данные за пн, ср, пт). 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.

План исследования:

О качестве исходных данных ничего не известно, поэтому перед проверкой гипотез понадобится проверка данных на ошибки и оценка их влияния на исследование, и по возможности, исправление критичных ошибок.

Таким образом, исследование пройдёт в три основных этапа: обзор данных, предобработка данных, проверка гипотез.

 Навыки и инструменты, применённые в работе: </h3>

- **Преобработка данных:** переименование столбцов, проверка и замена пропусков, проверка и удаление явных и не явных дубликатов
- **Фильтрация, сортировка, группировка данных**
- **Создание функций:** для замены неявных дубликатов, подсчёт значений при изменении 2-х переменных, подсчёт топ-10 жанров тех треков, которые прослушивали в указанный день, в промежутке между двумя отметками времени
- **Создание таблиц:** с помощью конструктора pd.DataFrame

Итоги исследования

В результате проверки гипотез установлено:

1. Первая гипотеза полностью подтвердилась. День недели по-разному влияет на активность пользователей в Москве и Петербурге. 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.

1. Музыкальные предпочтения в городах не сильно меняются в течение недели.Небольшие различия заметны по понедельникам:

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

Таким образом, вторая гипотеза подтвердилась частично. Но при отсутствии пропусков в данных, результат мог быть другим.

1. Третья гипотеза не подтвердилась, т.к. во музыкальных вкусах пользователей Москвы и Петербурга больше общего чем различий. Существующие различия на основной массе пользователей незаметны.

Примечание: в данном исследовании не использовались методы проверки статистических гипотез.

Детализация исследования

Раздел_1.Обзор данных

Составим первое представление о данных Яндекс.Музыки. Для этого импортируем библиотеку pandas и загрузим исходные данные:

In [3]:
```
import pandas as pd
```

In [4]:
```
df = pd.read_csv('yandex_music_project.csv')
```

Выведем на экран первые десять строк таблицы:

	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

Просмотрим общую информацию о таблице:

```
<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
```

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. В столбце `userID` использован 'camel case', вместо 'snake case'.

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

Выводы к разделу "Обзор данных"

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.

Встречаются пропуски в данных, а в названиях колонок — расхождения с хорошим стилем. В следующем разделе устраним найденные проблемы в данных.

Раздел_2.Предобработка данных

2.1.Стиль заголовков

Выведим на экран перечень названий столбцов таблицы df:

Out[7]:
```
Index(['  userID', 'Track', 'artist', 'genre', '  City  ', 'time', 'Day'], dtype='object')
```

Приведём названия в соответствие с хорошим стилем: несколько слов в названии перепишем в «змеином_регистре», все символы сделайем строчными и устраним пробелы.

Проверим результат, для этого ещё раз выведите на экран названия столбцов:

Out[9]:
```
Index(['user_id', 'track', 'artist', 'genre', 'city', 'time', 'day'], dtype='object')
```

2.2.Пропуски значений

Посчитаем, сколько в таблице пропущенных значений. Для этого используем методы `pandas`: isna() и sum()

Out[10]:
```
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 в данной работе, в данном случае будет достаточно заменить их явными обозначениями.

Пропуски в поле `genre` могут помешать сравнению музыкальных предпочтний в Москве и Санкт-Петербурге. На практике было бы правильно установить причину пропусков. В данном учебном проекте такой возможности нет. В связи с чем:

- заполним и эти пропуски явными обозначениями,
- оценим, насколько они повредят расчётам.

Заменим пропущенные значения в столбцах `track`, `artist` и `genre` на строку `'unknown'`. Для этого создадим список `columns_to_replace`, переберём его элементы циклом `for` и для каждого столбца выполним замену пропущенных значений:

In [11]:
```
# 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')
```

Убедимся, что в таблице не осталось пропусков. Для этого ещё раз посчитаем пропущенные значения.

Out[12]:
```
user_id    0
track      0
artist     0
genre      0
city       0
time       0
day        0
dtype: int64
```

2.3.Дубликаты

Посчитаем явные дубликаты в таблице и удалим явные дубликаты:

Out[13]:
```
3826
```

In [14]:
```
# удаление явных дубликатов (с удалением старых индексов и формированием новых)
df = df.drop_duplicates().reset_index(drop=True)
```

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

Теперь избавимся от неявных дубликатов в колонке `genre`. Это связано с тем, что название одного и того же жанра может быть

записано немного по-разному. Такие ошибки тоже повлияют на результат исследования.

Отсортируем данные в столбце `genre` и вернём все его уникальные значения.

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', '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)
```

В списке видим неявные дубликаты названия `hiphop`. Это могут быть названия с ошибками или альтернативные названия того же жанра: *hip*, *hop*, *hip-hop*.

Чтобы очистить от них таблицу, напишем функцию `replace_wrong_genres()` с двумя параметрами:

- `wrong_genres` — список дубликатов,
- `correct_genre` — строка с правильным значением.

Функция будет исправлять колонку `genre` в таблице `df`: заменять каждое значение из списка `wrong_genres` на значение из `correct_genre`.

In [17]:
```
# Функция для замены неявных дубликатов

def replace_wrong_genres(wrong_genres,correct_genre): # функция получает список дубликатов и строку с правильным
    for wrong_genre in wrong_genres: #перебор неправильных жанров
        df['genre'] =  df['genre'].replace(wrong_genres,correct_genre) #для каждого неправильно написанного жанра
```

Вызовим функцию `replace_wrong_genres()` и заменим значения `hip`, `hop` и `hip-hop` на значение `hiphop`:

In [18]:
```
# Устранение неявных дубликатов

duplicates = ['hip','hop','hip-hop'] # список неявных дубликатов жатра hiphop
```

```
right_genre = 'hiphop' # правильное написание жарна hiphop
replace_wrong_genres( duplicates, right_genre) #вызов функции для замены дубликатов на правильное написание жанра
```

Проверим правильность замены:

Out[19]:
```
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 к разделу "Предобработка данных"

Предобработка обнаружила три проблемы в данных:

- нарушения в стиле заголовков - были исправлены, чтобы упростить работу с таблицей;
- пропущенные значения-заменили на `'unknown'`. Далее необходимо проверить, как влияют на исследование пропуски в колонке `genre`;
- дубликаты — явные и неявные, без дубликатов исследование будет более точным.

Раздел_3.Проверка гипотез

Проверка гипотезы 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
- и сравним, сколько треков послушала каждая группа пользователей в понедельник, среду и пятницу.

Для тренировки выполним каждый из расчётов по отдельности.

Оценим активность пользователей в каждом городе: для этого сгруппируем данные по городу и посчитаем прослушивания в каждой группе.

Out[20]:
```
city
Moscow              42741
Saint-Petersburg    18512
Name: time, dtype: int64
```

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

Теперь сгруппируем данные по дню недели и подсчитаем прослушивания в понедельник, среду и пятницу. Необходимо учесть, что в данных есть информация только о прослушиваниях только за эти дни.

Out[21]:
```
day
Friday       21840
Monday       21354
Wednesday    18059
Name: time, dtype: int64
```

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

In [23]:
```
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 [24]:
```
# количество прослушиваний в Москве по понедельникам
day = 'Monday'
city= 'Moscow'
number_tracks(day,city)
```

Out[24]:
```
15740
```

In [25]:
```
# количество прослушиваний в Санкт-Петербурге по понедельникам
day = 'Monday'
city= 'Saint-Petersburg'
number_tracks(day,city)
```

Out[25]:
```
5614
```

In [26]:
```
# количество прослушиваний в Москве по средам
day = 'Wednesday'
city= 'Moscow'
number_tracks(day,city)
```

Out[26]:
```
11056
```

In [27]:
```
# количество прослушиваний в Санкт-Петербурге по средам
day = 'Wednesday'
city= 'Saint-Petersburg'
number_tracks(day,city)
```

Out[27]:
```
7003
```

In [28]:
```
# количество прослушиваний в Москве по пятницам
day = 'Friday'
city= 'Moscow'
```

```
number_tracks(day,city)
```

Out[28]: 15945

In [29]:
```
# количество прослушиваний в Санкт-Петербурге по пятницам
day = 'Friday'
city= 'Saint-Petersburg'
number_tracks(day,city)
```

Out[29]: 5895

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

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

Out[30]:

	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.

Проверка гипотезы 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.

Сохраните таблицы с данными в две переменные:

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

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 [33]:
```
# 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
#    Используется последовательная фильтрация с помощью логической индексации.
# 2) группируется датафрейм genre_df по столбцу genre, берётся столбе
#    столбцов и посчитать методом count() количество записей для каждого из
#    присутствующих жанров, получившийся Series записывается в переменную
#    genre_df_count
# 3) сортируется genre_df_count по убыванию встречаемости и сохранить
#    в переменную genre_df_sorted
# 4) возвращается Series из 10 первых значений genre_df_sorted, это будут топ-10
#    популярных жанров (в указанный день, в заданное время)
def genre_weekday(table, day, time1, time2):
```

```
    #genre_df = table[table['day'] == day]           # последовательная фильтрация пошагово
    #genre_df = genre_df[genre_df['time'] > time1]
    #genre_df = genre_df[genre_df['time'] < time2]
    genre_df = table[                                # та же последовательная фильтрация объединённым выражением
    (table['day'] == day)
    &(table['time'] > time1)
    &(table['time'] < time2)
    ]
    genre_df_count = genre_df.groupby('genre')['time'].count()
    genre_df_sorted = genre_df_count.sort_values(ascending = False)
    return genre_df_sorted.head(10)
```

In [34]:

```
# 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')
table = moscow_general #проиписывем значения каждой переменной, назовём их так же как параметры. Можно значения з
day = 'Monday'
time1 = '07:00'
time2 = '11:00'
genre_weekday(table, day, time1, time2) # можно переменные не объявлять, а записать значения сразу в скобках в ст
```

Out[34]:

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

In [35]:

```
# вызов функции для утра понедельника в Петербурге (вместо df — таблица spb_general)
table = spb_general
day = 'Monday'
time1 = '07:00'
time2 = '11:00'
genre_weekday(table, day, time1, time2)
```

Out[35]:

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

In [36]:

```
# вызов функции для вечера пятницы в Москве  17:00 до 23:00
table = moscow_general
day = 'Friday'
time1 = '17:00'
time2 = '23:00'
genre_weekday(table, day, time1, time2)
```

Out[36]:

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

In [37]:

```
#вызов функции для вечера пятницы в Петербурге
table = spb_general
day = 'Friday'
```

```
time1 = '17:00'
time2 = '23:00'
genre_weekday(table, day, time1, time2)
```

Out[37]:
```
genre
pop            256
electronic     216
rock           216
dance          210
hiphop          97
alternative     63
jazz            61
classical       60
rusrap          59
world           54
Name: time, 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 достаточно много, т.к.значение `'unknown'` заняло десятое место среди самых популярных жанров. Значит, пропущенные значения занимают существенную долю в данных и влияют на достоверность исследования.

Вечер пятницы похожая ситуация. 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 полученный результат. В Москве рейтинг топ-10 мог бы выглядеть иначе, если бы не пропуски.

Проверка гипотезы 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 посчитаем прослушивания треков каждого жанра методом `count()`. Затем отсортируем результат в порядке убывания и сохраним его в таблице `moscow_genres`.

In [38]:
```
moscow_genres = moscow_general.groupby('genre')['genre'].count().sort_values(ascending = False)
```

Выведите на экран первые десять строк `moscow_genres`:

In [39]:
```
moscow_genres.head(10)
```

Out[39]:
```
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
```

Повторим то же и для Петербурга.

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

Выведим на экран первые десять строк `spb_genres`:

Out[41]:
```
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
```

Выводы

Гипотеза 3 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.