Яндекс.Музыка

На данных Яндекс.Музыки сравним поведение пользователей двух столиц.

Цель исследования — проверим 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

Данные о поведении пользователей находятся в файле `yandex_music_project.csv` . О качестве данных ничего не известно. Поэтому перед проверкой гипотез понадобится обзор данных.

Проверим данные на ошибки и оценим их влияние на исследование. Затем, на этапе предобработки поищем возможность исправить самые критичные ошибки данных.

Таким образом, исследование пройдёт в три этапа:

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

1 Обзор данных

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

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

In [2]:
```
df = pd.read_csv('путь к файлу') # чтение файла с данными и сохранение в df
```

Первые десять строк таблицы:

In [ ]:
```
df.head(10) # получение первых 10 строк таблицы df
```

Одной командой получим общую информацию о таблице:

In [ ]:
```
df.info() # получение общей информации о данных в таблице df
```

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. Жирафная запись. То есть отклонение от змеиного_регистра.

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.

2 Предобработка данных

Исправим стиль в заголовках столбцов, исключим пропуски. Затем проверим данные на дубликаты.

2.1 Стиль заголовков

Выведем на экран названия столбцов:

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

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

Приведём названия в соответствие с хорошим стилем:

- несколько слов в названии запишем в «змеином_регистре»,
- все символы сделаем строчными,
- устраним пробелы.

Для этого переименуем колонки так:

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

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

Проверим результат. Для этого ещё раз выведем на экран названия столбцов:

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

Out[7]:
```
Index(['user_id', 'track', 'artist', 'genre', 'city', 'time', 'day'], dtype='objec
t')
```

2.2 Пропуски значений

Сначала посчитаем, сколько в таблице пропущенных значений. Для этого достаточно двух методов `pandas`:

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

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 этой 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.

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 [9]:
```
1 # перебор названий столбцов в цикле и замена пропущенных значений на 'unknown'
2 columns_to_replace = ['track', 'artist', 'genre'] # из этого списка в цикле буд
3 for col in columns_to_replace: # для колонки из списка колонок под замену
4     df[col] = df[col].fillna('unknown') # заменить отсутствующие значения в оче
5 df.info() # сведения о таблице
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65079 entries, 0 to 65078
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   user_id  65079 non-null  object
 1   track    65079 non-null  object
 2   artist   65079 non-null  object
 3   genre    65079 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
```

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

In [10]:
```
df.isna().sum() # подсчёт пропусков
```

Out[10]:
```
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 одной командой:

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

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

Вызовем специальный метод `pandas`, чтобы удалить явные дубликаты:

In [12]:
```
df = df.drop_duplicates().reset_index(drop=True)
# удаление явных дубликатов (с удалением старых индексов и формированием новых)
```

Ещё раз посчитаем явные дубликаты в таблице — убедимся, что полностью от них избавились:

In [13]:
```
df.duplicated().sum() # проверка на отсутствие дубликатов
```

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:

- извлечём нужный столбец датафрейма,
- применим к нему метод сортировки,
- для отсортированного столбца вызовем метод, который вернёт уникальные значения из столбца.

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

# africa afrikaans
# folk folklore
# deutschspr german
# hip-hop hiphop hip hop
# latin latino
# neue new
# türk türkçe
# электроника electronic
# вот неявные дубликаты на мой взгляд
```

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',
```

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

Видим следующие неявные дубликаты:

- *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 должна исправить колонку `genre` в таблице `df`: заменить каждое значение из списка `wrong_genres` на значение из `correct_genre`.

In [15]:
```
def replace_wrong_genres(wrong_genres, correct_genre):
    # Функция для замены неявных дубликатов
# список wrong_genres буду вводить явно при вызове функции.
# по одному вызову функции для каждого подсписка дубликатов
    for word in wrong_genres:
        df['genre'] = df['genre'].replace(word, correct_genre)


```

Вызовем `replace_wrong_genres()` и передадим ей такие аргументы, чтобы она устранила неявные дубликаты: вместо `hip`, `hop` и `hip-hop` в таблице должно быть значение `hiphop`:

In [16]:
```
# Устранение неявных дубликатов
replace_wrong_genres(['africa'], 'afrikaans')
replace_wrong_genres(['folklore'], 'folk')
replace_wrong_genres(['deutschspr'], 'german')
replace_wrong_genres(['hip-hop', 'hip', 'hop'], 'hiphop')
replace_wrong_genres(['latin'], 'latino')
replace_wrong_genres(['neue'], 'new')
replace_wrong_genres(['türkçe'], 'türk')
replace_wrong_genres(['электроника'], 'electronic')
```

Проверим, что заменили неправильные названия:

- hip
- hop
- hip-hop

Выведем отсортированный список уникальных значений столбца `genre`:

In [17]:
```
sorted(df['genre'].unique()) # Проверка на неявные дубликаты
```

```
'acoustic',
'action',
'adult',
'afrikaans',
'alternative',
'alternativepunk',
'ambient',
'americana',
'animated',
'anime',
'arabesk',
'arabic',
'arena',
'argentinetango',
'art',
'audiobook',
'author',
'avantgarde',
'axé',
'baile',
```

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 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.

Для тренировки сначала выполним каждый из расчётов по отдельности.

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

In [18]:

```
# Подсчёт прослушиваний в каждом городе

# формирую таблички по городам

df_msk = df[df['city'] == 'Moscow']
display(df_msk.head()) # вывод 5 первых строк для проверки

df_spb = df[df['city'] == 'Saint-Petersburg']
display(df_spb.head()) # вывод 5 первых строк для проверки

# в датасет НЕТ продолжительности звучания треков.
# значит ,на вопрос "Сколько слушают..." ответ будет
# не в днях/часах/минутах/сек
# а в штуках

all_tracks_spb = df_spb['time'].count() # подсчитываем Питер
print("Всего прослушиваний в СПБ:", all_tracks_spb) # выводим результат

all_tracks_msk = df_msk['time'].count() # подсчитываем
print("Всего прослушиваний в МСК:", all_tracks_msk) # выводим результат

```

	user_id	track	artist	genre	city	time	day
1	55204538	Delayed Because of Accident	Andreas Rönnberg	rock	Moscow	14:07:09	Friday
4	E2DC1FAE	Soul People	Space Echo	dance	Moscow	08:34:34	Monday
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	unknown	ruspop	Moscow	09:17:40	Friday

	user_id	track	artist	genre	city	time	day
0	FFB692EC	Kamigata To Boots	The Mass Missile	rock	Saint-Petersburg	20:28:33	Wednesday
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
5	842029A1	Преданная	IMPERVTOR	rusrap	Saint-Petersburg	13:09:41	Friday
9	E772D5C0	Pessimist	unknown	dance	Saint-Petersburg	21:20:49	Wednesday

```
Всего прослушиваний в СПБ: 18512
Всего прослушиваний в МСК: 42741
```

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. Учтём, что в данных есть информация только о прослушиваниях только за эти дни.

In [19]:

```
# Подсчёт прослушиваний в каждый из трёх дней
# надо добавить отбор по дню недели

# Питер понедельник
monday_tracks_spb = df_spb[df_spb['day'] == 'Monday']['time'].count()
print('Прослушивания в Питере в понедельник:', monday_tracks_spb)

# Питер среда
wednesday_tracks_spb = df_spb[df_spb['day'] == 'Wednesday']['time'].count()
print('Прослушивания в Питере в среду:', wednesday_tracks_spb)

# Питер пятница
friday_tracks_spb = df_spb[df_spb['day'] == 'Friday']['time'].count()
print('Прослушивания в Питере в пятницу:', friday_tracks_spb)

# и Москва:

# Москва понедельник
monday_tracks_msk = df_msk[df_msk['day'] == 'Monday']['time'].count()
print('Прослушивания в Москве в понедельник:', monday_tracks_msk)

# Москва среда
wednesday_tracks_msk = df_msk[df_msk['day'] == 'Wednesday']['time'].count()
print('Прослушивания в Москве в среду:', wednesday_tracks_msk)

# Москва пятница
friday_tracks_msk = df_msk[df_msk['day'] == 'Friday']['time'].count()
print('Прослушивания в Москве в пятницу:', friday_tracks_msk)

# много одинаковых конструкций. вероятно, фунуции вводить или упрощать как-то з
```

```
Прослушивания в Питере в понедельник: 5614
Прослушивания в Питере в среду: 7003
Прослушивания в Питере в пятницу: 5895
Прослушивания в Москве в понедельник: 15740
Прослушивания в Москве в среду: 11056
Прослушивания в Москве в пятницу: 15945
```

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

Вот и получилось, что в Питере в среду - пик.

Мы увидели, как работает группировка по городу и по дням недели. Теперь напишем функцию, которая объединит два эти 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. Вернём эту переменную из функции.

In [21]:
```
# <создание функции number_tracks()>

def number_tracks(day, city):
# 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).

# не получается что-то у меня в одной строке всё сделать.
# а если вот так, последовательно?
    track_list = df[df['day'] == day]
    track_list = track_list[track_list['city'] == city]

# В переменной track_list_count сохраняется число значений столбца 'user_id',
# рассчитанное методом count() для таблицы track_list.

    track_list_count = track_list['time'].count()

# Функция возвращает число - значение track_list_count.

    return 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.

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

Out[22]: 15740

In [23]:
```
number_tracks('Monday', 'Saint-Petersburg') # количество прослушиваний в Санкт-
```

Out[23]: 5614

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

Out[24]: 11056

In [25]:
```
number_tracks('Wednesday', 'Saint-Petersburg') # количество прослушиваний в Сан
```

Out[25]: 7003

In [26]:
```
number_tracks('Friday', 'Moscow') # количество прослушиваний в Москве по пятниц
```

Out[26]: 15945

In [27]:
```
number_tracks('Friday', 'Saint-Petersburg') # количество прослушиваний в Санкт-
```

Out[27]: 5895

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

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

In [28]:
```
# Таблица с результатами

# подготовлю списки
# колонки
names_columns = ['city', 'monday', 'wednesday', 'friday']

# print(type(names_columns))
# display(names_columns)
# сработало. выводтся список

# данные
data_set = [
    ['Moscow', number_tracks('Monday', 'Moscow'), number_tracks('Wednesday', 'M
         number_tracks('Friday', 'Moscow')],
    ['Saint-Petersburg', number_tracks('Monday', 'Saint-Petersburg'), \
        number_tracks('Wednesday', 'Saint-Petersburg'), \
            number_tracks('Friday', 'Saint-Petersburg')]
    ]

# таблица
table_counts_tracks_by_days_and_cities = pd.DataFrame(data=data_set, columns=na
# печать
display(table_counts_tracks_by_days_and_cities)
```

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

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 [29]:

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

# а вот это я и делал
# df_msk и df_spb
# что ж. переименую готовые датасеты

moscow_general = df_msk

moscow_general

```

Out[29]:

	user_id	track	artist	genre	city	time	day
1	55204538	Delayed Because of Accident	Andreas Rönnberg	rock	Moscow	14:07:09	Friday
4	E2DC1FAE	Soul People	Space Echo	dance	Moscow	08:34:34	Monday
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	unknown	ruspop	Moscow	09:17:40	Friday
...	...	...	...	...	...	...	...
61247	83A474E7	I Worship Only What You Bleed	The Black Dahlia Murder	extrememetal	Moscow	21:07:12	Monday
61248	729CBB09	My Name	McLean	rnb	Moscow	13:32:28	Wednesday
61250	C5E3A0D5	Jalopiina	unknown	industrial	Moscow	20:09:26	Friday
61251	321D0506	Freight Train	Chas McDevitt	rock	Moscow	21:43:59	Friday
61252	3A64EF84	Tell Me Sweet Little Lies	Monica Lopez	country	Moscow	21:59:46	Friday

42741 rows × 7 columns

In [30]:

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

Out[30]:

	user_id	track	artist	genre	city	time	day
0	FFB692EC	Kamigata To Boots	The Mass Missile	rock	Saint-Petersburg	20:28:33	Wednesday
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
5	842029A1	Преданная	IMPERVTOR	rusrap	Saint-Petersburg	13:09:41	Friday
9	E772D5C0	Pessimist	unknown	dance	Saint-Petersburg	21:20:49	Wednesday
...	...	...	...	...	...	...	...
61239	D94F810B	Theme from the Walking Dead	Proyecto Halloween	film	Saint-Petersburg	21:14:40	Monday
61240	BC8EC5CF	Red Lips: Gta (Rover Rework)	Rover	electronic	Saint-Petersburg	21:06:50	Monday
61241	29E04611	Bre Petrunko	Perunika Trio	world	Saint-Petersburg	13:56:00	Monday
61242	1B91C621	(Hello) Cloud Mountain	sleepmakeswaves	postrock	Saint-Petersburg	09:22:13	Monday
61249	D08D4A55	Maybe One Day (feat. Black Spade)	Blu & Exile	hiphop	Saint-Petersburg	10:00:00	Monday

18512 rows × 7 columns

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 [31]:

```python
# Объявление функции genre_weekday() с параметрами table, day, time1, time2,
# которая возвращает информацию о самых популярных жанрах в указанный день в
# заданное время:
def genre_weekday(table, day, time1, time2):

# 1) в переменную genre_df сохраняются те строки переданного датафрейма table, 
#    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.
    genre_df = table[table['day'] == day]
    genre_df = genre_df[genre_df['time'] > time1]
    genre_df = genre_df[genre_df['time'] < time2]


# 2) сгруппируем датафрейм genre_df по столбцу genre, взяв один из его
#    столбцов и посчитаем методом count() количество записей для каждого из
#    присутствующих жанров, получившийся Series запишем в переменную
#    genre_df_count
    genre_df_count = genre_df.groupby('genre')['time'].count()

# 3) отсортируем genre_df_count по убыванию встречаемости и сохраним
#    в переменную genre_df_sorted
    genre_df_sorted = genre_df_count.sort_values(ascending=False)

# 4) вернём Series из 10 первых значений genre_df_sorted, это будут топ-10
#    популярных жанров (в указанный день, в заданное время)
    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 [32]:

```python
# вызов функции для утра понедельника в Москве (вместо df — таблица moscow_gene
# 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[32]:

```
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 [33]:
```
# вызов функции для утра понедельника в Петербурге (вместо df — таблица spb_gen
genre_weekday(spb_general, 'Monday', '07:00', '11:00')
```

Out[33]:
```
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 [34]:
```
# вызов функции для вечера пятницы в Москве
genre_weekday(moscow_general, 'Friday', '17:00', '23:00')
```

Out[34]:
```
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 [35]:
```
# вызов функции для вечера пятницы в Петербурге
genre_weekday(spb_general, 'Friday', '17:00', '23:00')
```

Out[35]:
```
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 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 посчитаем прослушивания треков каждого жанра методом `count()`. Затем отсортируем результат в порядке убывания и сохраним его в таблице `moscow_genres`.

In [36]:
```
# одной строкой: группировка таблицы moscow_general по столбцу 'genre',

# подсчёт числа значений 'genre' в этой группировке методом count(),
moscow_genres = moscow_general.groupby('genre')['time'].count()

# сортировка получившегося Series в порядке убывания и сохранение в moscow_genre
moscow_genres = moscow_genres.sort_values(ascending=False)

```

Выведем на экран первые десять строк `moscow_genres`:

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

Out[37]:
```
genre
pop            5892
dance          4435
rock           3965
electronic     3786
hiphop         2096
classical      1616
world          1432
alternative    1379
ruspop         1372
rusrap         1161
Name: time, 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 сохраним в таблице `spb_genres`:

In [38]:
```
# одной строкой: группировка таблицы spb_general по столбцу 'genre',
# подсчёт числа значений 'genre' в этой группировке методом count(),
spb_genres = spb_general.groupby('genre')['time'].count()

# сортировка получившегося Series в порядке убывания и сохранение в spb_genres
spb_genres = spb_genres.sort_values(ascending=False)
```

Выведем на экран первые десять строк `spb_genres` :

In [39]:
```
# просмотр первых 10 строк spb_genres
spb_genres[:10]
```

Out[39]:
```
genre
pop            2431
dance          1932
rock           1879
electronic     1737
hiphop          960
alternative     649
classical       646
rusrap          564
ruspop          538
world           515
Name: time, 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.

4 Итоги исследования

Вы 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. С методами проверок гипотез вы ещё познакомитесь в следующих темах.