Яндекс.Музыка

Сравнение Москвы и Петербурга окружено мифами. Например:

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

Цель исследования — проверьте 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 получим из файла `yandex_music_project.csv`. О качестве данных ничего не известно. Поэтому перед проверкой гипотез понадобится обзор данных.

Проверим данные на ошибки и оцените 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.

Обзор данных

Составим первое представление о данных Яндекс.Музыки.

In [2]:

```
import pandas as pd
```

In [3]:

```
df = pd.read_csv('/datasets/yandex_music_project.csv')
```

Выведим на экран первые десять строк таблицы:

In [4]:

```
display(df.head(10))
```

	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 Gagarina	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

Одной командой получим общую информацию о таблице:

In [5]:

```
df.info()
```

```
<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. Найдите ещё одну проблему в названии колонок и опишите её в этом пункте.

Проблема: Третья проблема заключена в несоблюдении "змеиного" регистра. Колонка `userID` должна быть написана в соответствии данного регистра: `user_id`

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.

данных, а в названиях колонок — расхождения с хорошим стилем.

Чтобы двигаться дальше, нужно устранить проблемы в данных.

Предобработка данных

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

Стиль заголовков

In [6]:

```
df.columns
```

Out[6]:

```
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]:

```
df = df.rename(columns={'  userID': 'user_id', 'Track': 'track', '  City  ': 'city', 'Da
y': 'day'})
```

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

In [7]:

```
print(df.columns)
```

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

Пропуски значений

Сначала посчитаем, сколько в таблице пропущенных значений.

In [8]:

```
print(df.isna().sum())
```

```
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 вашей 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.

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

In [9]:

```
columns_to_replace = ['track', 'artist', 'genre']

for column in columns_to_replace:
    df[column] = df[column].fillna('unknown')
```

Убедитесь, что в таблице не осталось пропусков. Для этого ещё раз посчитайте пропущенные значения.

In [10]:

```
print(df.isna().sum())
```

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

Дубликаты

Посчитаем явные дубликаты в таблице одной командой:

In [11]:

```
print(df.duplicated().sum())
```

```
3826
```

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

In [12]:

```
df = df.drop_duplicates().reset_index(drop=True)
```

Ещё раз посчитайте явные дубликаты в таблице — убедитесь, что полностью от них избавились:

In [13]:

```
print(df.duplicated().sum())
```

```
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:

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

In [12]:

```
print(df['genre'].sort_values().unique())
```

```
['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' 'urban' 'uzbek' 'variété'

 'vi' 'videogame' 'vocal' 'western' 'world' 'worldbeat' 'ïîï'

 'электроника' nan]
```

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

Видим следующие неявные дубликаты:

- ***hip***,
- ***hop***,
- ***hip-hop***.

In [15]:

```
def replace_wrong_genres(wrong_genres, correct_genre):
    for wrong_genre in wrong_genres:
        df['genre'] = df['genre'].replace(wrong_genre, correct_genre)
```

In [16]:

```
duplicates = ['hip', 'hop', 'hip-hop']

replace_wrong_genres(duplicates, 'hiphop')
```

Проверим, что заменили неправильные названия:

- **hip**
- **hop**
- **hip-hop**

Выведем отсортированный список уникальных значений столбца `genre`:

In [13]:

```
print(df['genre'].sort_values().unique())
```

```
['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' 'urban' 'uzbek' 'variété'

 'vi' 'videogame' 'vocal' 'western' 'world' 'worldbeat' 'ïîï'

 'электроника' nan]
```

Выводы

Предобработка обнаружила три проблемы в данных:

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

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

Пропущенные значения вы заменили 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.

In [18]:

```
print(df.groupby('city')['user_id'].count())
```

```
city

Moscow              42741

Saint-Petersburg    18512

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 подсчитайте прослушивания в понедельник, среду и пятницу.

In [19]:

```
```

```
print(df.groupby('day')['user_id'].count())
```

```
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.

In [20]:

```
def number_tracks(day, city):
    track_list = df[df['city'] == city]
    track_list = track_list[track_list['day'] == day]
    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 [21]:

```
# количество прослушиваний в Москве по понедельникам
print(number_tracks('Monday', 'Moscow'))
```

```
15740
```

In [22]:

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

```
5614
```

In [23]:

```
# количество прослушиваний в Москве по средам
print(number_tracks('Wednesday', 'Moscow'))
```

```
11056
```

In [24]:

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

```
7003
```

In [25]:

```
# количество прослушиваний в Москве по пятницам
print(number_tracks('Friday', 'Moscow'))
```

```
15945
```

In [26]:

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

```
5895
```

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

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

In [27]:

```
data = [
    ['Moscow', 15740, 11056, 15945],
    ['Saint-Petersburg', 5614, 7003, 5895]
]
columns = ['city', 'monday', 'wednesday', 'friday']

track_number_result = pd.DataFrame(data=data, columns=columns)

display(track_number_result)
```

	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`.

In [28]:

```
moscow_general = df[df['city'] == 'Moscow']
```

In [29]:

```
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 [30]:

```
def genre_weekday(table, day, time1, time2):
    genre_df = table[table['day'] == day]
```

```
    genre_df = genre_df[genre_df['time'] > time1]
    genre_df = genre_df[genre_df['time'] < time2]
    genre_df_count = genre_df.groupby('genre')['user_id'].count()
    genre_df_sorted = genre_df_count.sort_values(ascending=False)
    return genre_df_sorted.head(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 [31]:

```
# вызов функции для утра понедельника в Москве
print(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: user_id, dtype: int64
```

In [32]:

```
# вызов функции для утра понедельника в Петербурге
print(genre_weekday(spb_general, 'Monday', '07:00', '11:00'))
```

```
genre

pop            218

dance          182

rock           162

electronic     147

hiphop          80

ruspop          64

alternative     58

rusrap          55

jazz            44

classical       40

Name: user_id, dtype: int64
```

In [33]:

```
# вызов функции для вечера пятницы в Москве
```

```
print(genre_weekday(moscow_general, 'Friday', '17:00', '23:00'))
```

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

In [34]:

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

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

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

In [35]:

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

In [36]:

```
print(moscow_genres.head(10))
```

```
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
```

In [37]:

```
spb_genres = spb_general.groupby('genre')['genre'].count().sort_values(ascending=False)
```

In [38]:

```
print(spb_genres.head(10))
```

```
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 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.

1. 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.

1. 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.

In [ ]: