Яндекс.Музыка

Содержание

Введение

Сравнение Москвы и Петербурга окружено мифами. Например:

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

На данных Яндекс.Музыки вы сравните поведение пользователей двух столиц.

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 вы получите из файла `yandex_music_project.csv`. О качестве данных ничего не известно. Поэтому перед проверкой гипотез понадобится обзор данных.

Вы проверите данные на ошибки и оцените их влияние на исследование. Затем, на этапе предобработки вы поищете возможность исправить самые критичные ошибки данных.

Таким образом, исследование пройдёт в три этапа:

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

Назад к «Содержанию»

Этап 1. Обзор данных

Составьте первое представление о данных Яндекс.Музыки.

Основной инструмент аналитика — `pandas`. Импортируйте эту библиотеку.

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

Очень здорово, что ты используешь сокращение pd для Pandas, это общепринятое сокращение для этой библиотеки для удобной дальнейшей работы.

Прочитайте файл `yandex_music_project.csv` из папки `/datasets` и сохраните его в переменной `df`:

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

Выведите на экран первые десять строк таблицы:

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

```
     userID                        Track            artist   genre  \
0  FFB692EC            Kamigata To Boots  The Mass Missile    rock   
1  55204538  Delayed Because of Accident  Andreas Rönnberg    rock   
2    20EC38            Funiculì funiculà       Mario Lanza     pop   
3  A3DD03C9        Dragons in the Sunset        Fire + Ice    folk   
4  E2DC1FAE                   Soul People        Space Echo   dance   
5  842029A1                     Преданная         IMPERVTOR  rusrap   
6  4CB90AA5                          True      Roman Messer   dance   
7  F03E1C1F             Feeling This Way   Polina Griffith   dance   
8  8FA1D3BE     И вновь продолжается бой               NaN  ruspop   
```

```
9  E772D5C0                    Pessimist               NaN   dance

                City      time        Day
0  Saint-Petersburg  20:28:33  Wednesday
1            Moscow  14:07:09     Friday
2  Saint-Petersburg  20:58:07  Wednesday
3  Saint-Petersburg  08:37:09     Monday
4            Moscow  08:34:34     Monday
5  Saint-Petersburg  13:09:41     Friday
6            Moscow  13:00:07  Wednesday
7            Moscow  20:47:49  Wednesday
8            Moscow  09:17:40     Friday
9  Saint-Petersburg  21:20:49  Wednesday
```

Одной командой получить общую информацию о таблице:

```
# получение общей информации о данных в таблице df
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. В названии первого столбца нужен "змеиный регистр", так как при замене всех бук на прописные user и id сольются.

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 пропуски в данных и

Чтобы двигаться дальше, нужно устранить проблемы в данных.

Назад к «Содержанию»

Этап 2. Предобработка данных

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

2.1 Стиль заголовков

Выведите на экран названия столбцов:

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

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

Приведите названия в соответствие с хорошим стилем:

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

Для этого переименуйте колонки так:

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

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

Проверьте результат. Для этого ещё раз выведите на экран названия столбцов:

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

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

Назад к «Содержанию»

2.2 Пропуски значений

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

```
# подсчёт пропусков
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.

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

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

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

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

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

Назад к «Содержанию»

2.3 Дубликаты

Посчитайте явные дубликаты в таблице одной командой:

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

```
3826
```

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

```
# удаление явных дубликатов
df = df.drop_duplicates().reset_index(drop
= True)
```

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

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

```
0
```

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

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

- извлеките нужный столбец датафрейма,
- примените к нему метод сортировки,

- для отсортированного столбца вызовите метод, который вернёт уникальные значения из столбца.

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

```
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)
```

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

Вы увидите следующие неявные дубликаты:

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

Чтобы очистить от них таблицу, напишите 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`.

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

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

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

Проверьте, что неправильные названия устранены. Выведите список уникальных значений столбца `genre`:

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

```
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)
```

Назад к «Содержанию»

2.4 Выводы

Предобработка обнаружила три проблемы в данных:

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

Вы 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.

Назад к «Содержанию»

Этап 3. Проверка гипотез

3.1 Сравнение поведения пользователей двух столиц

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

- Разделите пользователей Москвы и Санкт-Петербурга
- Сравните, сколько треков послушали каждая группы пользователей в понедельник, среду и пятницу.

Для тренировки сначала выполните каждый из расчётов по отдельности.

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

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

```
city
Moscow              42741
Saint-Petersburg    18512
Name: city, 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 сгруппируйте данные по дню недели и подсчитайте прослушивания в понедельник, среду и пятницу:

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

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

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

Вы видели, как работает группировка по городу и по дням недели. Теперь напишите функцию, которая объединит два эти расчёта.

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

- день недели,
- название города.

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

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

Для этого примените последовательную фильтрацию с логической индексацией.

Затем посчитайте значения в столбце `user_id` получившейся таблицы. Результат сохраните в новую переменную. Верните эту переменную из функции.

```
# <создание функции number_tracks()>
# Объявляется функция с двумя параметрами: day, city.

def number_tracks(day, city):
    track_list = df[(df['city'] == city) & (df['day'] == day)]
    track_list_count = len(track_list)
    return track_list_count
# В переменной track_list сохраняются те строки таблицы df, для
которых

# значение в столбце 'day' равно параметру day и одновременно значение
# в столбце 'city' равно параметру city (используйте последовательную
фильтрацию
# с помощью логической индексации).
# В переменной track_list_count сохраняется число значений столбца
'user_id',
# рассчитанное методом count() для таблицы track_list.
# Функция возвращает число - значение track_list_count.

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

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

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

```
15740
```

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

```
5614
```

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

```
11056
```

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

```
7003
```

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

```
15945
```

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

```
5895
```

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

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

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

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

Вывод

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.

Назад к «Содержанию»

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.

Получите таблицы данных:

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

```
# получение таблицы moscow_general из тех строк таблицы df,
# для которых значение в столбце 'city' равно 'Moscow'
moscow_general = df[df['city'] == 'Moscow'].reset_index(drop
= True)

# 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'].reset_index(drop
= True)
```

Создайте функцию `genre_weekday()` с четырьмя параметрами:

- таблица с данными,
- день недели,
- начальная временная метка в формате 'hh:mm',
- последняя временная метка в формате 'hh:mm'.

Функция должна вернуть информацию о топ-15 жанров тех треков, которые прослушивали в указанный день, в промежутке между двумя отметками времени.

```
def genre_weekday(tab, day, time1, time2):
    genre_df = tab[(tab['day'] == day) & (tab['time'] > time1) &
(tab['time'] < time2)]
    genre_df_count = genre_df.groupby('genre')['genre'].count()
    genre_df_sorted = genre_df_count.sort_values(ascending=False)
    return genre_df_sorted.head(15)

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

```
# 1) в переменную genre_df сохраняются те строки датафрейма df, для
#    которых одновременно:
#    - значение в столбце weekday 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 15 первых значений genre_df_sorted, это будут
топ-15
#    популярных жанров (в указанный день, в заданное время)
```

Сравните 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):

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

```
genre
pop            781
dance          549
electronic     480
rock           474
hiphop         286
ruspop         186
world          181
rusrap         175
alternative    164
unknown        161
classical      157
metal          120
jazz           100
folk            97
soundtrack      95
Name: genre, dtype: int64
```

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

```
genre
pop            218
dance          182
rock           162
electronic     147
hiphop          80
ruspop          64
alternative     58
rusrap          55
jazz            44
classical       40
world           36
rap             32
soundtrack      31
metal           27
rnb             27
Name: genre, dtype: int64
```

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

```
genre
pop            713
rock           517
dance          495
electronic     482
hiphop         273
world          208
ruspop         170
alternative    163
classical      163
rusrap         142
jazz           111
unknown        110
soundtrack     105
rnb             90
metal           88
Name: genre, dtype: int64
```

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

```
genre
pop            256
rock           216
electronic     216
dance          210
hiphop          97
alternative     63
jazz            61
classical       60
```

```
rusrap          59
world           54
ruspop          47
unknown         47
soundtrack      40
metal           39
rap             36
Name: genre, dtype: int64
```

Вывод

Если сравнить топ-15 жанров в понедельник утром, можно сделать такие выводы:

1. В Москве и Петербурге слушают похожую музыку. Единственное отличие — в московский рейтинг вошла русская поп-музыка, а в петербургский — джаз.
2. В Москве пропущенных значений оказалось так много, что значение `'unknown'` заняло одиннадцатое 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.

Назад к «Содержанию»

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.

Сгруппируйте 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`.

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

Выведите на экран первые десять строк `moscow_genres`:

```
# просмотр первых 10 строк moscow_genres
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
```

Задание 3.2.10

Теперь повторите то же и для Петербурга.

Сгруппируйте таблицу `spb_general` по жанру. Посчитайте прослушивания треков каждого жанра. Результат отсортируйте в порядке убывания и сохраните в таблице `spb_genres`:

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

Задание 3.2.11

Выведите на экран первые десять строк `spb_genres`:

```
# просмотр первых 10 строк spb_genres
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
```

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.

Назад к «Содержанию»

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.

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 русскую популярную музыку,
- в Петербурге — джаз.

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.

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.

Назад к «Содержанию»