Yandex.Music

Contents

Introducción

En este proyecto, compararemos las preferencias musicales de las ciudades de Springfield y Shelbyville. Probaremos hipótesis de Yandex.Music para comparar el comportamiento del usuario de esas dos ciudades.

Objetivo:

Prueba tres hipótesis:

1. La actividad de los usuarios difiere según el día de la semana y dependiendo de la ciudad.
2. Los lunes por la mañana, los habitantes de Springfield y Shelbyville escuchan diferentes géneros. Lo mismo ocurre con los viernes por la noche.
3. Los oyentes de Springfield y Shelbyville tienen preferencias distintas. En Springfield prefieren el pop mientras que en Shelbyville hay más aficionados al rap.

Etapas

Primero, evaluaremos la calidad de los datos y veremos si los problemas son significativos. Entonces, durante el preprocesamiento de datos, tomarás en cuenta los problemas más críticos.

Este proyecto consiste en tres etapas:

1. Descripción de los datos
2. Preprocesamiento de datos
3. Prueba de hipótesis

Volver a Contenidos

Etapa 1. Descripción de los datos

In [1]:
```
# importando pandas
import pandas as pd
```

In [2]:
```
# leyendo el archivo y almacenándolo en df
df = pd.read_csv("/datasets/music_project_en.csv")
print(df)
```

```
        userID                            Track            artist  \
0      FFB692EC                Kamigata To Boots  The Mass Missile
1      55204538      Delayed Because of Accident  Andreas Rönnberg
2        20EC38                Funiculì funiculà      Mario Lanza
3      A3DD03C9            Dragons in the Sunset        Fire + Ice
4      E2DC1FAE                      Soul People        Space Echo
...         ...                              ...               ...
65074  729CBB09                          My Name            McLean
65075  D08D4A55  Maybe One Day (feat. Black Spade)       Blu & Exile
65076  C5E3A0D5                        Jalopiina               NaN
65077  321D0506                    Freight Train     Chas McDevitt
65078  3A64EF84          Tell Me Sweet Little Lies      Monica Lopez

           genre       City        time        Day
0           rock  Shelbyville  20:28:33  Wednesday
1           rock  Springfield  14:07:09     Friday
2            pop  Shelbyville  20:58:07  Wednesday
3           folk  Shelbyville  08:37:09     Monday
4          dance  Springfield  08:34:34     Monday
...          ...          ...       ...        ...
65074        rnb  Springfield  13:32:28  Wednesday
65075        hip  Shelbyville  10:00:00     Monday
65076 industrial  Springfield  20:09:26     Friday
65077       rock  Springfield  21:43:59     Friday
65078    country  Springfield  21:59:46     Friday

[65079 rows x 7 columns]
```

In [3]:
```
# obteniendo las 10 primeras filas de la tabla df
df.head(10)
```

Out[3]:

	userID	Track	artist	genre	City	time	Day
0	FFB692EC	Kamigata To Boots	The Mass Missile	rock	Shelbyville	20:28:33	Wednesday
1	55204538	Delayed Because of Accident	Andreas Rönnberg	rock	Springfield	14:07:09	Friday
2	20EC38	Funiculì funiculà	Mario Lanza	pop	Shelbyville	20:58:07	Wednesday
3	A3DD03C9	Dragons in the Sunset	Fire + Ice	folk	Shelbyville	08:37:09	Monday
4	E2DC1FAE	Soul People	Space Echo	dance	Springfield	08:34:34	Monday
5	842029A1	Chains	Obladaet	rusrap	Shelbyville	13:09:41	Friday
6	4CB90AA5	True	Roman Messer	dance	Springfield	13:00:07	Wednesday
7	F03E1C1F	Feeling This Way	Polina Griffith	dance	Springfield	20:47:49	Wednesday
8	8FA1D3BE	L'estate	Julia Dalia	ruspop	Springfield	09:17:40	Friday
9	E772D5C0	Pessimist	NaN	dance	Shelbyville	21:20:49	Wednesday

In [4]:
```
# obteniendo información general sobre los datos en df
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65079 entries, 0 to 65078
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0     userID  65079 non-null  object
 1   Track     63736 non-null  object
 2   artist    57512 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
```

Conclusiones

Cada fila de la tabla almacena datos de la pista que fue reproducida. Algunas columnas describen la pista en sí: su título, el artista y el género. El resto transmite la información del usuario: la ciudad de la que viene, el tiempo que ha reproducido la pista.

Está claro que los datos son suficientes para probar la hipótesis. Sin embargo, hay valores ausentes.

Para continuar, necesitamos preprocesar los datos.

Volver a Contenidos

Etapa 2. Preprocesamiento de datos

Será corregido el formato en los encabezados de las columnas y los valores ausentes.

Estilo del encabezado

In [5]:
```
# la lista de los nombres de las columnas en la tabla df
print(df.columns)
```

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

Unificaremos en un criterio de solo minusculas y sin espacios los encabezados de las columnas

In [6]:
```
# renombra las columnas

df = df.rename(
  columns={
    '  userID':'user_id',
    'Track':'track',
    'artist':'artist',
    'genre':'genre',
    '  City  ':'city',
    'time':'time',
    'Day':'day',
  }
 )
```

In [7]:
```
# comprobando el resultado: la lista de los nombres de las columnas
(df.columns)
```

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

Volver a Contenidos

Valores ausentes

In [8]:
```
# calculando valores ausentes
print(df.isna().sum())
```

```
user_id       0
track      1343
artist     7567
genre      1198
city          0
time          0
day           0
dtype: int64
```

No todos los valores ausentes afectan directamente los objetivos del proyecto en el caso de nuestros valores ausentes serán reemplazados con "unknow" dado que el reemplazo de estos no afecta o cambia el resultado final

In [9]:
```
# recorriendo los nombres de las columnas y reemplazando los valores ausentes con 'unknown'
columns_to_replace= ['track', 'artist', 'genre']

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

```
track
artist
genre
```

In [10]:
```
# contando valores ausentes
print(df.isna().sum())
```

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

Volver a Contenidos

Duplicados

Buscaremos el total de duplicados obvios en la tabla.

In [11]:
```
# contando duplicado obvios
print(df.duplicated().sum())
```

```
3826
```

In [12]:
```
# eliminando duplicados obvios
df=df.drop_duplicates().reset_index(drop=True)
```

In [13]:
```
# comprobando duplicados
print(df.duplicated().sum())
```

```
0
```

Ahora eliminaremos los duplicados implícitos en la columna genre. Dado que el nombre de un género se puede escribir de varias formas. Dichos errores también pueden afectar a resultado.

In [14]:
```
# inspeccionando los nombres de géneros únicos
unique_genres = df['genre'].unique()
print(sorted(unique_genres))
```

```
['acid', 'acoustic', 'action', 'adult', 'africa', 'afrikaans', 'alternative', 'ambient', 'americana', 'animated', 'anime', 'arabesk', 'arabic', 'arena', 'argentinetang
o', 'art', 'audiobook', 'avantgarde', 'axé', 'baile', 'balkan', 'beats', 'bigroom', 'black', 'bluegrass', 'blues', 'bollywood', 'bossa', 'brazilian', 'breakbeat', 'brea
ks', 'broadway', 'cantautori', 'cantopop', 'canzone', 'caribbean', 'caucasian', 'celtic', 'chamber', 'children', 'chill', 'chinese', 'choral', 'christian', 'christmas',
'classical', 'classicmetal', 'club', 'colombian', 'comedy', 'conjazz', 'contemporary', 'country', 'cuban', 'dance', 'dancehall', 'dancepop', 'dark', 'death', 'deep', 'd
eutschrock', 'deutschspr', 'dirty', 'disco', 'dnb', 'documentary', 'downbeat', 'downtempo', 'drum', 'dub', 'dubstep', 'eastern', 'easy', 'electronic', 'electropop', 'em
o', 'entehno', 'epicmetal', 'estrada', 'ethnic', 'eurofolk', 'european', 'experimental', 'extrememetal', 'fado', 'film', 'fitness', 'flamenco', 'folk', 'folklore', 'fol
kmetal', 'folkrock', 'folktronica', 'forró', 'frankreich', 'französisch', 'french', 'funk', 'future', 'gangsta', 'garage', 'german', 'ghazal', 'gitarre', 'glitch', 'gos
pel', 'gothic', 'grime', 'grunge', 'gypsy', 'handsup', "hard'n'heavy", 'hardcore', 'hardstyle', 'hardtechno', 'hip', 'hip-hop', 'hiphop', 'historisch', 'holiday', 'ho
p', 'horror', 'house', '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', 'loungeelectroni
c', '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', 'othe
r', 'piano', 'pop', 'popelectronic', 'popeurodance', 'post', 'posthardcore', 'postrock', 'power', 'progmetal', 'progressive', 'psychedelic', 'punjabi', 'punk', 'quebeco
is', 'ragga', 'ram', 'rancheras', 'rap', 'rave', 'reggae', 'reggaeton', 'regional', 'relax', 'religious', 'retro', 'rhythm', 'rnb', 'rnr', 'rock', 'rockabilly', 'romanc
e', 'roots', 'ruspop', 'rusrap', 'rusrock', 'salsa', 'samba', 'schlager', 'self', 'sertanejo', 'shoegazing', 'showtunes', 'singer', 'ska', 'slow', 'smooth', 'soul', 'so
ulful', 'sound', 'soundtrack', 'southern', 'specialty', 'speech', 'spiritual', 'sport', 'stonerrock', 'surf', 'swing', 'synthpop', 'sängerportrait', 'tango', 'tanzorche
ster', 'taraftar', 'tech', 'techno', 'thrash', 'top', 'traditional', 'tradjazz', 'trance', 'tribal', 'trip', 'triphop', 'tropical', 'türk', 'türkçe', 'unknow', 'urban',
'uzbek', 'variété', 'vi', 'videogame', 'vocal', 'western', 'world', 'worldbeat', 'ïïï']
```

Tenemos duplicados implícitos:

- `hip`
- `hop`
- `hip-hop`

In [15]:
```
# función para reemplazar duplicados implícitos

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]:
```
# eliminando duplicados implícitos
duplicates = ['hip', 'hop', 'hip-hop']
name = 'hiphop'
replace_wrong_genres(duplicates, name)
```

Confirmamos haber eliminado los duplicados implícitos

In [17]:
```
# revisando en busca de duplicados implícitos
unique_genres = df['genre'].unique()
print(sorted(unique_genres))
```

```
['acid', 'acoustic', 'action', 'adult', 'africa', 'afrikaans', 'alternative', 'ambient', 'americana', 'animated', 'anime', 'arabesk', 'arabic', 'arena', 'argentinetang
o', 'art', 'audiobook', 'avantgarde', 'axé', 'baile', 'balkan', 'beats', 'bigroom', 'black', 'bluegrass', 'blues', 'bollywood', 'bossa', 'brazilian', 'breakbeat', 'brea
ks', 'broadway', 'cantautori', 'cantopop', 'canzone', 'caribbean', 'caucasian', 'celtic', 'chamber', 'children', 'chill', 'chinese', 'choral', 'christian', 'christmas',
'classical', 'classicmetal', 'club', 'colombian', 'comedy', 'conjazz', 'contemporary', 'country', 'cuban', 'dance', 'dancehall', 'dancepop', 'dark', 'death', 'deep', 'd
eutschrock', 'deutschspr', 'dirty', 'disco', 'dnb', 'documentary', 'downbeat', 'downtempo', 'drum', 'dub', 'dubstep', 'eastern', 'easy', 'electronic', 'electropop', 'em
o', 'entehno', 'epicmetal', 'estrada', 'ethnic', 'eurofolk', 'european', 'experimental', 'extrememetal', 'fado', 'film', 'fitness', 'flamenco', 'folk', 'folklore', 'fol
kmetal', 'folkrock', 'folktronica', 'forró', 'frankreich', 'französisch', 'french', 'funk', 'future', 'gangsta', 'garage', 'german', 'ghazal', 'gitarre', 'glitch', 'gos
pel', 'gothic', 'grime', 'grunge', 'gypsy', 'handsup', "hard'n'heavy", 'hardcore', 'hardstyle', 'hardtechno', 'hiphop', 'historisch', 'holiday', 'horror', 'house', 'id
m', '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', 'mu
slim', 'native', 'neoklassik', 'neue', 'new', 'newage', 'newwave', 'nu', 'nujazz', 'numetal', 'oceania', 'old', 'opera', 'orchestral', 'other', 'piano', 'pop', 'popelec
tronic', 'popeurodance', 'post', 'posthardcore', 'postrock', 'power', 'progmetal', 'progressive', 'psychedelic', 'punjabi', 'punk', 'quebecois', 'ragga', 'ram', 'ranche
ras', 'rap', 'rave', 'reggae', 'reggaeton', 'regional', 'relax', 'religious', 'retro', 'rhythm', 'rnb', 'rnr', 'rock', 'rockabilly', 'romance', 'roots', 'ruspop', 'rusr
ap', 'rusrock', 'salsa', 'samba', 'schlager', 'self', 'sertanejo', 'shoegazing', 'showtunes', 'singer', 'ska', 'slow', 'smooth', 'soul', 'soulful', 'sound', 'soundtrac
k', 'southern', 'specialty', 'speech', 'spiritual', 'sport', 'stonerrock', 'surf', 'swing', 'synthpop', 'sängerportrait', 'tango', 'tanzorchester', 'taraftar', 'tech',
'techno', 'thrash', 'top', 'traditional', 'tradjazz', 'trance', 'tribal', 'trip', 'triphop', 'tropical', 'türk', 'türkçe', 'unknow', 'urban', 'uzbek', 'variété', 'vi',
'videogame', 'vocal', 'western', 'world', 'worldbeat', 'ïïï']
```

Volver a Contenidos

Conclusiones

Detectamos tres problemas con los datos:

- Estilos de encabezados incorrectos
- Valores ausentes
- Duplicados obvios e implícitos

Los encabezados han sido eliminados para conseguir que el procesamiento de la tabla sea más sencillo.

Todos los valores ausentes han sido reemplazados por `'unknown'`. Pero todavía tenemos que ver si los valores ausentes en `'genre'` afectan a nuestros cálculos.

Volver a Contenidos

Etapa 3. Prueba de hipótesis

Hipótesis 1: comparar el comportamiento del usuario en las dos ciudades

De acuerdo con la primera hipótesis, los usuarios de Springfield y Shelbyville escuchan música de forma distinta. Comprueba esto utilizando los datos de tres días de la semana: lunes, miércoles y viernes.

- Divide a los usuarios en grupos por ciudad.
- Compara cuántas pistas reprodujo cada grupo el lunes, el miércoles y el viernes.

In [18]:
```
# contando las pistas reproducidas en cada ciudad

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

Out[18]:
```
city
Shelbyville    18512
Springfield    42741
Name: track, dtype: int64
```

Springfield ha reproducido más pistas que Shelbyville. Pero eso no implica que los ciudadanos de Springfield escuchen música más a menudo. Esta ciudad es simplemente más grande y hay más usuarios.

Ahora agrupa los datos por día de la semana y encuentra el número de pistas reproducidas el lunes, miércoles y viernes.

In [19]:
```
# calculando las pistas reproducidas en cada uno de los tres días

df.groupby('day')['track'].count()
```

Out[19]:
```
day
Friday       21840
Monday       21354
Wednesday    18059
Name: track, dtype: int64
```

El miércoles fue el día más silencioso de todos. Pero si consideramos las dos ciudades por separado podríamos llegar a una conclusión diferente.

In [20]:
```
# <creando la función number_tracks()>
# declararemos la función con dos parámetros: day=, city=.

# deja que la variable track_list almacene las filas df en las que
# el valor en la columna 'day' es igual al parámetro day= y, al mismo tiempo,
# el valor de la columna 'city' es igual al parámetro city= (aplica el filtrado consecutivo
# con indexación lógica).
# deja que la variable track_list_count almacene el número de valores de la columna 'user_id' en track_list
# (encontrado con el método count()).
# permite que la función devuelva un número: el valor de track_list_count.
# la función cuenta las pistas reproducidas en un cierto día y ciudad.
# primero recupera las filas del día deseado de la tabla,
# después filtra las filas de la ciudad deseada del resultado,
# entonces, encuentra el número de valores de 'user_id' en la tabla filtrada,b
# y devuelve ese número.
# para ver lo que devuelve, envuelve la llamada de la función en print().

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

Utilizamos `number_tracks()` seis veces, para obtener los datos de las ciudades en cada día estuadiado.

In [21]:
```
# el número de canciones reproducidas en Springfield el lunes
print(number_tracks('Monday','Springfield'))
```

```
15740
```

In [22]:
```
# el número de canciones reproducidas en Shelbyville el lunes
print(number_tracks('Monday', 'Shelbyville'))
```

```
5614
```

In [23]:
```
# el número de canciones reproducidas en Springfield el miércoles
print(number_tracks('Wednesday', 'Springfield'))
```

```
11056
```

In [24]:
```
# el número de canciones reproducidas en Shelbyville el miércoles
print(number_tracks('Wednesday', 'Shelbyville'))
```

```
7003
```

In [25]:
```
# el número de canciones reproducidas en Springfield el viernes
print(number_tracks('Friday','Springfield'))
```

```
15945
```

In [26]:
```
# el número de canciones reproducidas en Shelbyville el viernes
print(number_tracks('Friday','Shelbyville'))
```

```
5895
```

In [1]:
```
#Almacenamos los datos recuperados
header=['city','monday','wednesday','friday']
new_data = [['Springfield',15740,11056,15945],
           ['Shelbyville',5614,7003,5895]]
```

In [28]:
```
# tabla con los resultados
new_tabla= pd.DataFrame(data=new_data, columns=header)
print(new_tabla)
```

```
          city  monday  wednesday  friday
0  Springfield   15740      11056   15945
1  Shelbyville    5614       7003    5895
```

Conclusiones

Los datos revelan las diferencias en el comportamiento de los usuarios:

- En Springfield, el número de canciones reproducidas alcanzan el punto máximo los lunes y viernes mientras que los miércoles hay un descenso de la actividad.
- En Shelbyville, al contario, los usuarios escuchan más música los miércoles. La actividad de los usuarios los lunes y viernes es menor.

Así que la primera hipótesis parece ser correcta.

Volver a Contenidos

Hipótesis 2: música al principio y al final de la semana

De acuerdo con la segunda hipótesis, los lunes por la mañana y los viernes por la noche los ciudadanos de Springfield escuchan géneros que difieren de aquellos que los usuarios de Shelbyville disfrutan.

In [29]:
```
# obteniendo la tabla spr_general de las filas de df,
# donde los valores en la columna 'city' es 'Springfield'

spr_general= df[df['city']== 'Springfield']
print(spr_general)
```

```
       user_id                         track                  artist  \
1      55204538    Delayed Because of Accident         Andreas Rönnberg
4      E2DC1FAE                    Soul People               Space Echo
6      4CB90AA5                           True             Roman Messer
7      F03E1C1F               Feeling This Way          Polina Griffith
8      8FA1D3BE                        L’estate              Julia Dalia
...         ...                            ...                      ...
61247  83A474E7  I Worship Only What You Bleed  The Black Dahlia Murder
61248  729CBB09                        My Name                   McLean
61250  C5E3A0D5                      Jalopiina                   unknow
61251  321D0506                  Freight Train            Chas McDevitt
61252  3A64EF84      Tell Me Sweet Little Lies             Monica Lopez

              genre         city      time        day
1              rock  Springfield  14:07:09     Friday
4             dance  Springfield  08:34:34     Monday
6             dance  Springfield  13:00:07  Wednesday
7             dance  Springfield  20:47:49  Wednesday
8            ruspop  Springfield  09:17:40     Friday
...             ...          ...       ...        ...
61247  extrememetal  Springfield  21:07:12     Monday
61248           rnb  Springfield  13:32:28  Wednesday
61250    industrial  Springfield  20:09:26     Friday
61251          rock  Springfield  21:43:59     Friday
61252       country  Springfield  21:59:46     Friday

[42741 rows x 7 columns]
```

In [30]:
```
# obteniendo shel_general de las filas df,
# donde el valor de la columna 'city' es 'Shelbyville'

shel_general= df[df['city']== 'Shelbyville']
print(shel_general)
```

```
       user_id                              track              artist  \
0      FFB692EC                  Kamigata To Boots    The Mass Missile
2        20EC38                  Funiculì funiculà         Mario Lanza
3      A3DD03C9              Dragons in the Sunset          Fire + Ice
5      842029A1                             Chains            Obladaet
9      E772D5C0                          Pessimist              unknow
...         ...                                ...                 ...
61239  D94F810B        Theme from the Walking Dead  Proyecto Halloween
61240  BC8EC5CF       Red Lips: Gta (Rover Rework)               Rover
61241  29E04611                        Bre Petrunko       Perunika Trio
61242  1B91C621             (Hello) Cloud Mountain     sleepmakeswaves
61249  D08D4A55  Maybe One Day (feat. Black Spade)         Blu & Exile

            genre         city      time        day
0            rock  Shelbyville  20:28:33  Wednesday
2             pop  Shelbyville  20:58:07  Wednesday
3            folk  Shelbyville  08:37:09     Monday
5          rusrap  Shelbyville  13:09:41     Friday
9           dance  Shelbyville  21:20:49  Wednesday
...           ...          ...       ...        ...
61239        film  Shelbyville  21:14:40     Monday
61240  electronic  Shelbyville  21:06:50     Monday
61241       world  Shelbyville  13:56:00     Monday
61242    postrock  Shelbyville  09:22:13     Monday
61249      hiphop  Shelbyville  10:00:00     Monday

[18512 rows x 7 columns]
```

In [31]:
```
# declarando la función genre_weekday() con los parámetros day=, time1= y time2=. Debería
# devolver información sobre los géneros más populares de un determinado día a una determinada hora:

# 1) Deja que la variable genre_df almacene las filas que cumplen varias condiciones:
#    - el valor de la columna 'day' es igual al valor del argumento day=
#    - el valor de la columna 'time' es mayor que el valor del argumento time1=
#    - el valor en la columna 'time' es menor que el valor del argumento time2=
#    Utiliza un filtrado consecutivo con indexación lógica.

# 2) Agrupa genre_df por la columna 'genre', toma una de sus columnas,
#    y utiliza el método count() para encontrar el número de entradas por cada uno de
#    los géneros representados; almacena los Series resultantes en
#    la variable genre_df_count

# 3) Ordena genre_df_count en orden descendente de frecuencia y guarda el resultado
#    en la variable genre_df_sorted

# 4) Devuelve un objeto Series con los primeros 15 valores de genre_df_sorted - los 15
#    géneros más populares (en un determinado día, en un determinado periodo de tiempo)

# Escribe tu función aquí

def genre_weekday(df, day, time1, time2):


    # filtrado consecutivo
    # genre_df solo almacenará aquellas filas df en las que el día sea igual a day=
    genre_df = df[(df['day'] == day)]

    # genre_df solo almacenará aquellas filas df en las que el tiempo sea menos que time2=
    genre_df = genre_df[(genre_df ['time'] < time2)]

    # genre_df solo almacenará aquellas filas df en las que el tiempo sea mayor que time1=
    genre_df = genre_df[(genre_df ['time'] > time1)]

    # agrupa el DataFrame filtrado por la columna con los nombres de los géneros, toma la columna de género, y encuentra el número de filas por cada género con el métod
    genre_df_grouped = genre_df.groupby('genre')['genre'].count()

    # ordenaremos el resultado en orden descendente (por lo que los géneros más populares aparecerán primero en el objeto Series)
    genre_df_sorted = genre_df_grouped.sort_values(ascending=False)

    # devolveremos el objeto Series que almacena los 15 géneros más populares en un día determinado en un periodo de tiempo determinado
    return genre_df_sorted[:15]
```

Comparamos los resultados de la función `genre_weekday()` para Springfield y Shelbyville el lunes por la mañana (de 7 a 11) y el viernes por la tarde (de 17:00 a 23:00):

In [32]:
```
# llamando a la función para el lunes por la mañana en Springfield (utilizando spr_general en vez de la tabla df)
genre_weekday(spr_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
unknow         161
classical      157
metal          120
jazz           100
folk            97
soundtrack      95
Name: genre, dtype: int64
```

In [33]:
```
# llamando a la función para el lunes por la mañana en Shelbyville (utilizando shel_general en vez de la tabla df)
genre_weekday(shel_general, 'Monday', '7:00', '11:00')
```

Out[33]:
```
Series([], Name: genre, dtype: int64)
```

In [34]:
```
# llamando a la función para el viernes por la tarde en Springfield
genre_weekday(spr_general, 'Friday', '17:00', '23:00')
```

Out[34]:
```
genre
pop            713
rock           517
dance          495
electronic     482
hiphop         273
world          208
ruspop         170
classical      163
alternative    163
rusrap         142
jazz           111
unknow         110
soundtrack     105
rnb             90
metal           88
Name: genre, dtype: int64
```

In [35]:
```
# llamando a la función para el viernes por la tarde en Shelbyville
genre_weekday(shel_general, 'Friday', '17:00', '23:00')
```

Out[35]:
```
genre
pop            256
rock           216
electronic     216
dance          210
hiphop          97
alternative     63
jazz            61
classical       60
rusrap          59
world           54
unknow          47
ruspop          47
soundtrack      40
metal           39
rap             36
Name: genre, dtype: int64
```

Conclusión

Habiendo comparado los 15 géneros más populares del lunes por la mañana podemos concluir lo siguiente:

1. Los usuarios de Springfield y Shelbyville escuchan música similar. Los cinco géneros más populares son los mismos, solo rock y electrónica han intercambiado posiciones.

2. En Springfield el número de valores ausentes resultaron ser tan altos que el valor `'unknown'` llegó al décimo. Esto significa que los valores ausentes forman una parte considerable de los datos, lo que podría ser la base de la cuestión sobre la fiabilidad de nuestras conclusiones.

Para el viernes por la tarde, la situación es similar. Los géneros individuales varían algo pero, en general, los 15 más populares son parecidos en las dos ciudades.

De esta forma, la segunda hipótesis ha sido parcialmente demostrada:

- Los usuarios escuchan música similar al principio y al final de la semana.
- No hay una gran diferencia entre Springfield y Shelbyville. En ambas ciudades, el pop es el género más popular.

Sin embargo, el número de valores ausentes hace este resultado un tanto cuestionable. En Springfield, hay tantos que afectan a nuestros 15 más populares. De no faltarnos esos valores, las cosas podrían parecer diferentes.

Volver a Contenidos

Hipótesis 3: preferencias de género en Springfield y Shelbyville

Hipótesis: Shelbyville ama la música rap. A los ciudadanos de Springfield les gusta más el pop.

In [36]:
```
# en una línea: agrupa la tabla spr_general por la columna 'genre',
# cuenta los valores 'genre' con count() en la agrupación,
# ordena el Series resultante en orden descendiente, y almacénalo en spr_genres

spr_genres=spr_general.groupby('genre')['genre'].count().sort_values(ascending=False)
spr_genres
```

Out[36]:
```
genre
pop            5892
dance          4435
rock           3965
electronic     3786
hiphop         2096
               ...
metalcore         1
marschmusik       1
malaysian         1
lovers            1
ïïï               1
Name: genre, Length: 250, dtype: int64
```

In [37]:
```
# imprimiendo las 10 primeras filas de spr_genres
spr_genres.head(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: genre, dtype: int64
```

In [38]:
```
# en una línea: agrupa la tabla shel_general por la columna 'genre',
# cuenta los valores 'genre' en el agrupamiento con count(),
# ordena el Series resultante en orden descendente y guárdalo en shel_genres

shel_genres=shel_general.groupby('genre')['genre'].count().sort_values(ascending=False)
shel_genres
```

Out[38]:
```
genre
pop           2431
dance         1932
rock          1879
electronic    1736
hiphop         960
              ...
mandopop         1
leftfield        1
laiko            1
jungle           1
worldbeat        1
Name: genre, Length: 202, dtype: int64
```

In [39]:
```
# imprimiendo las 10 primeras filas de shel_genres
shel_genres.head(10)
```

Out[39]:
```
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
```

Conclusión

La hipótesis ha sido parcialmente demostrada:

- La música pop es el género más popular en Springfield, tal como se esperaba.
- Sin embargo, la música pop ha resultado ser igual de popular en Springfield que en Shelbyville y el rap no estaba entre los 5 más populares en ninguna de las ciudades.

Volver a Contenidos

Conclusiones

Hemos probado las siguientes tres hipótesis:

1. La actividad de los usuarios difiere dependiendo del día de la semana y de las distintas ciudades.
2. Los lunes por la mañana los residentes de Springfield y Shelbyville escuchan géneros distintos. Lo mismo ocurre con los viernes por la noche.
3. Los oyentes de Springfield y Shelbyville tienen distintas preferencias. En ambas ciudades, Springfield y Shelbyville, se prefiere el pop.

Tras analizar los datos, concluimos:

1. La actividad del usuario en Springfield y Shelbyville depende del día de la semana aunque las ciudades varían de diferentes formas.

La primera hipótesis ha sido aceptada completamente.

1. Las preferencias musicales no varían significativamente en el transcurso de la semana en Springfield y Shelbyville. Podemos observar pequeñas diferencias en el orden los lunes, pero:

- En Springfield y Shelbyville la gente lo que más escucha es la música pop.

Así que no podemos aceptar esta hipótesis. También debemos tener en cuenta que el resultado podría haber sido diferente si no fuera por los valores ausentes.

1. Resulta que las preferencias musicales de los usuarios de Springfield y Shelbyville son bastante parecidas.

La tercera hipótesis es rechazada. Si hay alguna diferencia en las preferencias no se puede observar en los datos.

Volver a Contenidos