d7ea73a3-4e8c-461e-8c1c-52b223671cff-Copy1

November 20, 2024

1 Déjame escuchar la música

2 Contenido

2.1 Introducción

Como analista de datos, tu trabajo consiste en analizar datos para extraer información valiosa y tomar decisiones basadas en ellos. Esto implica diferentes etapas, como la descripción general de los datos, el preprocesamiento y la prueba de hipótesis.

Siempre que investigamos, necesitamos formular hipótesis que después podamos probar. A veces aceptamos estas hipótesis; otras veces, las rechazamos. Para tomar las decisiones correctas, una empresa debe ser capaz de entender si está haciendo las suposiciones correctas.

En este proyecto, compararás las preferencias musicales de las ciudades de Springfield y Shelbyville. Estudiarás datos reales de transmisión de música online para probar la hipótesis a continuación y comparar el comportamiento de los usuarios y las usuarias de estas dos ciudades.

2.1.1 Objetivo:

Prueba la hipótesis: 1. La actividad de los usuarios y las usuarias difiere según el día de la semana y dependiendo de la ciudad.

2.1.2 Etapas

Los datos del comportamiento del usuario se almacenan en el archivo `/datasets/music_project_en.csv`. No hay ninguna información sobre la calidad de los datos, así que necesitarás examinarlos antes de probar la hipótesis.

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

Tu proyecto consistirá en tres etapas: 1. Descripción de los datos. 2. Preprocesamiento de datos. 3. Prueba de hipótesis.

2.2 Etapa 1. Descripción de los datos

Abre los datos y examínalos.

Necesitarás `pandas`, así que impórtalo.

[7]:
```
import pandas as pd # Importar pandas
```

Lee el archivo `music_project_en.csv` de la carpeta `/datasets/` y guárdalo en la variable `df`:

[8]:
```
df = pd.read_csv("/datasets/music_project_en.csv")  # Leer el archivo y␣
 ↪almacenarlo en df
# Leer el archivo y almacenarlo en df
```

Muestra las 10 primeras filas de la tabla:

[9]:
```
print(df.head(10)) # Obtener las 10 primeras filas de la tabla df
```

```
     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                        Chains          Obladaet  rusrap
6  4CB90AA5                          True      Roman Messer   dance
7  F03E1C1F             Feeling This Way   Polina Griffith   dance
8  8FA1D3BE                     L'estate       Julia Dalia  ruspop
9  E772D5C0                    Pessimist               NaN   dance

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

Obtén la información general sobre la tabla con un comando. Conoces el método que muestra la información general que necesitamos.

[10]:
```
print(df.info())# Obtener la información general sobre nuestros datos
```

```
<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
None
```

Estas son nuestras observaciones sobre la tabla. Contiene siete columnas. Almacenan los mismos tipos de datos: `object`.

Según la documentación: - `' userID'`: identificador del usuario o la usuaria; - `'Track'`: título de la canción; - `'artist'`: nombre del artista; - `'genre'`: género de la pista; - `'City'`: ciudad del usuario o la usuaria; - `'time'`: la hora exacta en la que se reprodujo la canción; - `'Day'`: día de la semana.

Podemos ver tres problemas con el estilo en los encabezados de la tabla: 1. Algunos encabezados están en mayúsculas, otros en minúsculas. 2. Hay espacios en algunos encabezados. 3. `Detecta el tercer problema por tu cuenta y descríbelo aquí`. el tercer problema que logro observar es en el encabezado "userID", deberia estar separado por un guion bajo (user_id)

2.2.1 Escribe observaciones de tu parte. Estas son algunas de las preguntas que pueden ser útiles:

`1. ¿Qué tipo de datos tenemos a nuestra disposición en las filas?¿Y cómo podemos entender lo que almacenan las columnas?` el tipo de datos en todas las columnas son object, podria cambiarse a un tipo de dato numerico `2. ¿Hay suficientes datos para proporcionar respuestas a nuestra hipótesis o necesitamos más información?` con las columnas "time" y "day" podemos calcular la cantidad de horas que se consume musica y a su vez los dias de la semana donde mas se consume

`3. ¿Notaste algún problema en los datos, como valores ausentes, duplicados o tipos de datos incorrectos?` no posee valores ausentes, pero puede que posea duplicados y en cuanto a los tipos de datos no todos deberian ser str

2.3 Etapa 2. Preprocesamiento de datos

El objetivo aquí es preparar los datos para que sean analizados. El primer paso es resolver cualquier problema con los encabezados. Luego podemos avanzar a los valores ausentes y duplicados. Empecemos.

Corrige el formato en los encabezados de la tabla.

2.3.1 Estilo del encabezado

Muestra los encabezados de la tabla (los nombres de las columnas):

[11]:
```
print(df.columns)# Muestra los nombres de las columnas
```

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

Cambia los encabezados de la tabla de acuerdo con las reglas del buen estilo: * Todos los caracteres deben ser minúsculas. * Elimina los espacios. * Si el nombre tiene varias palabras, utiliza snake_case.

Anteriormente, aprendiste acerca de la forma automática de cambiar el nombre de las columnas. Vamos a aplicarla ahora. Utiliza el bucle for para iterar sobre los nombres de las columnas y poner todos los caracteres en minúsculas. Cuando hayas terminado, vuelve a mostrar los encabezados de la tabla:

[12]:
```
names_lower=[]
for old_name in df.columns:
    name_lower= old_name.lower()
    names_lower.append(name_lower)
df.columns=names_lower# Bucle en los encabezados poniendo todo en minúsculas
print(df.columns)
```

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

Ahora, utilizando el mismo método, elimina los espacios al principio y al final de los nombres de las columnas e imprime los nombres de las columnas nuevamente:

[13]:
```
names_strip=[]
for old_name in df.columns:
    name_strip= old_name.strip()
    names_strip.append(name_strip)
df.columns=names_strip# Bucle en los encabezados poniendo todo en minúsculas
print(df.columns)# Bucle en los encabezados eliminando los espacios
```

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

Necesitamos aplicar la regla de snake_case a la columna `userid`. Debe ser `user_id`. Cambia el nombre de esta columna y muestra los nombres de todas las columnas cuando hayas terminado.

[14]:
```
names_replace=[]
for old_name in df.columns:
    name_replace= old_name.replace('userid','user_id')
    names_replace.append(name_replace)
df.columns=names_replace# Bucle en los encabezados poniendo todo en minúsculas
```

```
print(df.columns)# Cambiar el nombre de la columna "userid"
```

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

Comprueba el resultado. Muestra los encabezados una vez más:

[15]:
```
print(df.columns)# Comprobar el resultado: la lista de encabezados
```

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

Volver a Contenidos

2.3.2 Valores ausentes

Primero, encuentra el número de valores ausentes en la tabla. Debes utilizar dos métodos en una secuencia para obtener el número de valores ausentes.

[16]:
```
print(df.isna().sum())# Calcular el número de valores ausentes
```

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

No todos los valores ausentes afectan a la investigación. Por ejemplo, los valores ausentes en `track` y `artist` no son cruciales. Simplemente puedes reemplazarlos con valores predeterminados como el string `'unknown'` (desconocido).

Pero los valores ausentes en `'genre'` pueden afectar la comparación entre las preferencias musicales de Springfield y Shelbyville. En la vida real, sería útil saber las razones por las cuales hay datos ausentes e intentar recuperarlos. Pero no tenemos esa oportunidad en este proyecto. Así que tendrás que: * rellenar estos valores ausentes con un valor predeterminado; * evaluar cuánto podrían afectar los valores ausentes a tus cómputos;

Reemplazar los valores ausentes en las columnas `'track'`, `'artist'` y `'genre'` con el string `'unknown'`. Como mostramos anteriormente en las lecciones, la mejor forma de hacerlo es crear una lista que almacene los nombres de las columnas donde se necesita el reemplazo. Luego, utiliza esta lista e itera sobre las columnas donde se necesita el reemplazo haciendo el propio reemplazo.

[17]:
```
columns_replace=['track','artist','genre']
for col in columns_replace:
    df[col].fillna('unknown', inplace=True)

# Bucle en los encabezados reemplazando los valores ausentes con 'unknown'
```

Ahora comprueba el resultado para asegurarte de que después del reemplazo no haya valores ausentes en el conjunto de datos. Para hacer esto, cuenta los valores ausentes nuevamente.

[18]:
```
print(df.isna().sum())# Contar valores ausentes
```

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

Volver a Contenidos

2.3.3 Duplicados

Encuentra el número de duplicados explícitos en la tabla. Una vez más, debes aplicar dos métodos en una secuencia para obtener la cantidad de duplicados explícitos.

[19]:
```
print(df.duplicated().sum())# Contar duplicados explícitos
```

```
3826
```

Ahora, elimina todos los duplicados. Para ello, llama al método que hace exactamente esto.

[20]:
```
df=df.drop_duplicates()# Eliminar duplicados explícitos
```

Comprobemos ahora si eliminamos con éxito todos los duplicados. Cuenta los duplicados explícitos una vez más para asegurarte de haberlos eliminado todos:

[21]:
```
print(df.duplicated().sum())# Comprobar de nuevo si hay duplicados
```

```
0
```

Ahora queremos deshacernos de los duplicados implícitos en la columna `genre`. Por ejemplo, el nombre de un género se puede escribir de varias formas. Dichos errores también pueden afectar al resultado.

Para hacerlo, primero mostremos una lista de nombres de género únicos, ordenados en orden alfabético. Para ello: * Extrae la columna `genre` del DataFrame. * Llama al método que devolverá todos los valores únicos en la columna extraída.

[22]:
```
print(df["genre"].unique())#Inspeccionar los nombres de géneros únicos
```

```
['rock' 'pop' 'folk' 'dance' 'rusrap' 'ruspop' 'world' 'electronic'
 'unknown' 'alternative' 'children' 'rnb' 'hip' 'jazz' 'postrock' 'latin'
 'classical' 'metal' 'reggae' 'triphop' 'blues' 'instrumental' 'rusrock'
 'dnb' 'türk' 'post' 'country' 'psychedelic' 'conjazz' 'indie'
 'posthardcore' 'local' 'avantgarde' 'punk' 'videogame' 'techno' 'house'
 'christmas' 'melodic' 'caucasian' 'reggaeton' 'soundtrack' 'singer' 'ska'
```

```
'salsa' 'ambient' 'film' 'western' 'rap' 'beats' "hard'n'heavy"
'progmetal' 'minimal' 'tropical' 'contemporary' 'new' 'soul' 'holiday'
'german' 'jpop' 'spiritual' 'urban' 'gospel' 'nujazz' 'folkmetal'
'trance' 'miscellaneous' 'anime' 'hardcore' 'progressive' 'korean'
'numetal' 'vocal' 'estrada' 'tango' 'loungeelectronic' 'classicmetal'
'dubstep' 'club' 'deep' 'southern' 'black' 'folkrock' 'fitness' 'french'
'disco' 'religious' 'hiphop' 'drum' 'extrememetal' 'türkçe'
'experimental' 'easy' 'metalcore' 'modern' 'argentinetango' 'old' 'swing'
'breaks' 'eurofolk' 'stonerrock' 'industrial' 'funk' 'middle' 'variété'
'other' 'adult' 'christian' 'thrash' 'gothic' 'international' 'muslim'
'relax' 'schlager' 'caribbean' 'nu' 'breakbeat' 'comedy' 'chill' 'newage'
'specialty' 'uzbek' 'k-pop' 'balkan' 'chinese' 'meditative' 'dub' 'power'
'death' 'grime' 'arabesk' 'romance' 'flamenco' 'leftfield' 'european'
'tech' 'newwave' 'dancehall' 'mpb' 'piano' 'top' 'bigroom' 'opera'
'celtic' 'tradjazz' 'acoustic' 'epicmetal' 'hip-hop' 'historisch'
'downbeat' 'downtempo' 'africa' 'audiobook' 'jewish' 'sängerportrait'
'deutschrock' 'eastern' 'action' 'future' 'electropop' 'folklore'
'bollywood' 'marschmusik' 'rnr' 'karaoke' 'indian' 'rancheras'
'afrikaans' 'rhythm' 'sound' 'deutschspr' 'trip' 'lovers' 'choral'
'dancepop' 'retro' 'smooth' 'mexican' 'brazilian' 'ïîï' 'mood' 'surf'
'gangsta' 'inspirational' 'idm' 'ethnic' 'bluegrass' 'broadway'
'animated' 'americana' 'karadeniz' 'rockabilly' 'colombian' 'self' 'hop'
'sertanejo' 'japanese' 'canzone' 'lounge' 'sport' 'ragga' 'traditional'
'gitarre' 'frankreich' 'emo' 'laiko' 'cantopop' 'glitch' 'documentary'
'oceania' 'popeurodance' 'dark' 'vi' 'grunge' 'hardstyle' 'samba'
'garage' 'art' 'folktronica' 'entehno' 'mediterranean' 'chamber' 'cuban'
'taraftar' 'gypsy' 'hardtechno' 'shoegazing' 'bossa' 'latino' 'worldbeat'
'malaysian' 'baile' 'ghazal' 'arabic' 'popelectronic' 'acid' 'kayokyoku'
'neoklassik' 'tribal' 'tanzorchester' 'native' 'independent' 'cantautori'
'handsup' 'punjabi' 'synthpop' 'rave' 'französisch' 'quebecois' 'speech'
'soulful' 'jam' 'ram' 'horror' 'orchestral' 'neue' 'roots' 'slow'
'jungle' 'indipop' 'axé' 'fado' 'showtunes' 'arena' 'irish' 'mandopop'
'forró' 'dirty' 'regional']
```

Busca en la lista para encontrar duplicados implícitos del género `hiphop`. Estos pueden ser nombres escritos incorrectamente o nombres alternativos para el mismo género.

Verás los siguientes duplicados implícitos: * `hip` * `hop` * `hip-hop`

Para deshacerte de ellos, crea una función llamada `replace_wrong_genres()` con dos parámetros: * `wrong_genres=`: esta es una lista que contiene todos los valores que necesitas reemplazar. * `correct_genre=`: este es un string que vas a utilizar como reemplazo.

Como resultado, la función debería corregir los nombres en la columna `'genre'` de la tabla `df`, es decir, remplazar cada valor de la lista `wrong_genres` por el valor en `correct_genre`.

Dentro del cuerpo de la función, utiliza un bucle `'for'` para iterar sobre la lista de géneros incorrectos, extrae la columna `'genre'` y aplica el método `replace` para hacer correcciones.

[23]:
```
def replace_wrong_genres( wrong_genres, correct_genre):
    for wrong_genre in wrong_genres:
        df['genre'] = df['genre'].replace(wrong_genre, correct_genre)

    return df # Función para reemplazar duplicados implícitos
```

Ahora, llama a `replace_wrong_genres()` y pásale tales argumentos para que retire los duplicados implícitos (`hip`, `hop` y `hip-hop`) y los reemplace por `hiphop`:

[25]:
```
wrong_genre= ["hip", "hop" , "hip-hop"]
correct_genre="hiphop"
genres= replace_wrong_genres(wrong_genre, correct_genre)
print(genres)
```

```
        user_id                                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           unknown
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     hiphop  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

[61253 rows x 7 columns]
```

Asegúrate de que los nombres duplicados han sido eliminados. Muestra la lista de valores únicos de la columna `'genre'` una vez más:

[26]:
```
print(df['genre'].unique())# Comprobación de duplicados implícitos
```

```
['rock' 'pop' 'folk' 'dance' 'rusrap' 'ruspop' 'world' 'electronic']
```

```
'unknown' 'alternative' 'children' 'rnb' 'hiphop' 'jazz' 'postrock'
'latin' 'classical' 'metal' 'reggae' 'triphop' 'blues' 'instrumental'
'rusrock' 'dnb' 'türk' 'post' 'country' 'psychedelic' 'conjazz' 'indie'
'posthardcore' 'local' 'avantgarde' 'punk' 'videogame' 'techno' 'house'
'christmas' 'melodic' 'caucasian' 'reggaeton' 'soundtrack' 'singer' 'ska'
'salsa' 'ambient' 'film' 'western' 'rap' 'beats' "hard'n'heavy"
'progmetal' 'minimal' 'tropical' 'contemporary' 'new' 'soul' 'holiday'
'german' 'jpop' 'spiritual' 'urban' 'gospel' 'nujazz' 'folkmetal'
'trance' 'miscellaneous' 'anime' 'hardcore' 'progressive' 'korean'
'numetal' 'vocal' 'estrada' 'tango' 'loungeelectronic' 'classicmetal'
'dubstep' 'club' 'deep' 'southern' 'black' 'folkrock' 'fitness' 'french'
'disco' 'religious' 'drum' 'extrememetal' 'türkçe' 'experimental' 'easy'
'metalcore' 'modern' 'argentinetango' 'old' 'swing' 'breaks' 'eurofolk'
'stonerrock' 'industrial' 'funk' 'middle' 'variété' 'other' 'adult'
'christian' 'thrash' 'gothic' 'international' 'muslim' 'relax' 'schlager'
'caribbean' 'nu' 'breakbeat' 'comedy' 'chill' 'newage' 'specialty'
'uzbek' 'k-pop' 'balkan' 'chinese' 'meditative' 'dub' 'power' 'death'
'grime' 'arabesk' 'romance' 'flamenco' 'leftfield' 'european' 'tech'
'newwave' 'dancehall' 'mpb' 'piano' 'top' 'bigroom' 'opera' 'celtic'
'tradjazz' 'acoustic' 'epicmetal' 'historisch' 'downbeat' 'downtempo'
'africa' 'audiobook' 'jewish' 'sängerportrait' 'deutschrock' 'eastern'
'action' 'future' 'electropop' 'folklore' 'bollywood' 'marschmusik' 'rnr'
'karaoke' 'indian' 'rancheras' 'afrikaans' 'rhythm' 'sound' 'deutschspr'
'trip' 'lovers' 'choral' 'dancepop' 'retro' 'smooth' 'mexican'
'brazilian' 'ïîï' 'mood' 'surf' 'gangsta' 'inspirational' 'idm' 'ethnic'
'bluegrass' 'broadway' 'animated' 'americana' 'karadeniz' 'rockabilly'
'colombian' 'self' 'sertanejo' 'japanese' 'canzone' 'lounge' 'sport'
'ragga' 'traditional' 'gitarre' 'frankreich' 'emo' 'laiko' 'cantopop'
'glitch' 'documentary' 'oceania' 'popeurodance' 'dark' 'vi' 'grunge'
'hardstyle' 'samba' 'garage' 'art' 'folktronica' 'entehno'
'mediterranean' 'chamber' 'cuban' 'taraftar' 'gypsy' 'hardtechno'
'shoegazing' 'bossa' 'latino' 'worldbeat' 'malaysian' 'baile' 'ghazal'
'arabic' 'popelectronic' 'acid' 'kayokyoku' 'neoklassik' 'tribal'
'tanzorchester' 'native' 'independent' 'cantautori' 'handsup' 'punjabi'
'synthpop' 'rave' 'französisch' 'quebecois' 'speech' 'soulful' 'jam'
'ram' 'horror' 'orchestral' 'neue' 'roots' 'slow' 'jungle' 'indipop'
'axé' 'fado' 'showtunes' 'arena' 'irish' 'mandopop' 'forró' 'dirty'
'regional']
```

Volver a Contenidos

Volver a Contenidos

2.4 Etapa 3. Prueba de hipótesis

2.4.1 Hipótesis: comparar el comportamiento del usuario o la usuaria en las dos ciudades

La hipótesis afirma que existen diferencias en la forma en que los usuarios y las usuarias de Springfield y Shelbyville consumen música. Para comprobar esto, usa los datos de tres días de la semana: lunes, miércoles y viernes.

- Agrupa a los usuarios y las usuarias por ciudad.
- Compara el número de canciones que cada grupo reprodujo el lunes, el miércoles y el viernes.

Realiza cada cálculo por separado.

El primer paso es evaluar la actividad del usuario en cada ciudad. Recuerda las etapas dividir-aplicar-combinar de las que hablamos anteriormente en la lección. Tu objetivo ahora es agrupar los datos por ciudad, aplicar el método apropiado para contar durante la etapa de aplicación y luego encontrar la cantidad de canciones reproducidas en cada grupo especificando la columna para obtener el recuento.

A continuación se muestra un ejemplo de cómo debería verse el resultado final: `df.groupby(by='....')['column'].method()`Realiza cada cálculo por separado.

Para evaluar la actividad de los usuarios y las usuarias en cada ciudad, agrupa los datos por ciudad y encuentra la cantidad de canciones reproducidas en cada grupo.

[27]:
```
df.groupby("city")['track'].count() # Contar las canciones reproducidas en cada␣
↪ciudad
```

[27]:
```
city
Shelbyville    18512
Springfield    42741
Name: track, dtype: int64
```

`Comenta tus observaciones aquí` #! se puede concluir que en springfield cosumen mucho mas musica que en shelbyville

Ahora agrupemos los datos por día de la semana y encontremos el número de canciones reproducidas el lunes, miércoles y viernes. Utiliza el mismo método que antes, pero ahora necesitamos una agrupación diferente.

[28]:
```
df.groupby('day')['track'].count()# Calcular las canciones reproducidas en cada␣
↪uno de los tres días
```

[28]:
```
day
Friday       21840
Monday       21354
Wednesday    18059
Name: track, dtype: int64
```

`Comenta tus observaciones aquí` #! se puede concluir que el viernes es el dia donde las personas escuchan mas musica

Ya sabes cómo contar entradas agrupándolas por ciudad o día. Ahora necesitas escribir una función que pueda contar entradas según ambos criterios simultáneamente.

Crea la función `number_tracks()` para calcular el número de canciones reproducidas en un determinado día **y** ciudad. La función debe aceptar dos parámetros:

- `day`: un día de la semana para filtrar. Por ejemplo, `'Monday'` (lunes).
- `city`: una ciudad para filtrar. Por ejemplo, `'Springfield'`.

Dentro de la función, aplicarás un filtrado consecutivo con indexación lógica.

Primero filtra los datos por día y luego filtra la tabla resultante por ciudad.

Después de filtrar los datos por dos criterios, cuenta el número de valores de la columna 'user_id' en la tabla resultante. Este recuento representa el número de entradas que estás buscando. Guarda el resultado en una nueva variable y devuélvelo desde la función.

[39]:
```
def number_tracks (day,city):# Declara la función number_tracks() con dos␣
↪parámetros: day= y city=.
    filter_day = df[df["day"] == day]
    filter_city = filter_day[filter_day["city"] == city]
    filtro = filter_city["user_id"].count()
    return filtro # Filtra las filas donde el valor en la columna 'city' es␣
↪igual al parámetro city=

    ## Extrae la columna 'user_id' de la tabla filtrada y aplica el método␣
↪count()

    # Devolve el número de valores de la columna 'user_id'
```

Llama a `number_tracks()` seis veces, cambiando los valores de los parámetros para que recuperes los datos de ambas ciudades para cada uno de los tres días.

[33]:
```
number_tracks(day='Monday', city='Springfield')# El número de canciones␣
↪reproducidas en Springfield el lunes
```

[33]:
```
15740
```

[34]:
```
number_tracks(day='Monday', city='Shelbyville')# El número de canciones␣
↪reproducidas en Shelbyville el lunes
```

[34]:
```
5614
```

[35]:
```
number_tracks(day='Wednesday', city='Springfield')# El número de canciones␣
↪reproducidas en Springfield el miércoles
```

[35]:
```
11056
```

[36]:
```
number_tracks(day='Wednesday', city='Shelbyville')# El número de canciones␣
↪reproducidas en Shelbyville el miércoles
```

[36]: 7003

[37]:
```
number_tracks(day='Friday', city='Springfield')# El número de canciones␣
↪reproducidas en Springfield el viernes
```

[37]: 15945

[38]:
```
number_tracks(day='Friday', city='Shelbyville')# El número de canciones␣
↪reproducidas en Shelbyville el viernes
```

[38]: 5895