

# Déjame escuchar la música

## ✓ Contenido

- [Introducción](#)
- [Etapa 1. Descripción de los datos](#)
  - [Conclusiones](#)
- [Etapa 2. Preprocesamiento de datos](#)
  - [2.1 Estilo del encabezado](#)
  - [2.2 Valores ausentes](#)
  - [2.3 Duplicados](#)
  - [2.4 Conclusiones](#)
- [Etapa 3. Prueba de hipótesis](#)
  - [3.1 Hipótesis 1: actividad de los usuarios y las usuarias en las dos ciudades](#)
- [Conclusiones](#)

## ✓ Introducción

### 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.

### 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.

## 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.

[Volver a Contenidos](#)

## ✓ Etapa 1. Descripción de los datos

Abre los datos y examínalos.

### Comentario del revisor

La tabla de contenidos está bien estructurada, pero sería útil si estuviera enlazada a las secciones correspondientes, de manera que al hacer clic se pueda acceder directamente a cada una. Esto facilitaría la navegación.

Un consejo útil: si etiquetas correctamente todas las secciones con sus respectivos números (#), puedes generar automáticamente la tabla de contenidos desde Jupyter Lab. Simplemente ve a la pestaña de herramientas y haz clic en el botón de los puntos y barras (Table of contents). Esto te generará una tabla de contenidos enlazada y visualmente atractiva. El botón está ubicado a la derecha del botón "Validate".

**Aun estoy aprendiendo como usar jupiter, los vinculos parecen funcionar por ratos, no he logrado hacer que funcione al 100, ya revise la opción que me señalaste y es mucho mas cómoda para la navegación.**

Necesitarás pandas , así que impórtalo.

```
# Importar pandas
import pandas as pd
```

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

```
# Leer el archivo y almacenarlo en df
df = pd.read_csv("/datasets/music_project_en.csv")
```

Muestra las 10 primeras filas de la tabla:

```
# Obtener las 10 primeras filas de la tabla df
print(df.head(11))
```

```

➡      userID      Track      artist  genre \
0  FFB692EC      Kamigata To Boots  The Mass Missile  rock
1  55204538  Delayed Because of Accident  Andreas Rönnerberg  rock
2    20EC38      Funiculi 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
10 BC5A3A29      Gool la Mita  Shireen Abdul Wahab  world

      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
10 Springfield  14:08:42  Monday

```

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

```
# Obtener la información general sobre nuestros datos
print(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
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. Se puede observar que hay valores faltantes en algunas columnas.

✓ 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? Tenemos datos almacenados como strings, y los datos que almacenan las columnas se pueden entender con el nombre y observando las primeras filas que a que se refiere cada una.
2. ¿Hay suficientes datos para proporcionar respuestas a nuestra hipótesis o necesitamos más información? La muestra de datos es extensa 65079 filas de datos pueden ser suficientes, habría que identificar con otros aspectos de análisis de datos para ver si la muestra es suficiente o tendra que extenderse.

3. ¿Notaste algún problema en los datos, como valores ausentes, duplicados o tipos de datos incorrectos? Hay varios datos faltantes en la columna genre, artist y Track, se tendrá que analizar posteriormente para revisar que tipo de duplicados tenemos. Además que el nombre de las columnas necesitan ser normalizados para no tener problemas al momento de llamar las columnas en el código.

```
dtype= df.loc[1, " userID"]  
print(type(dtype))
```

```
↩ <class 'str'>
```

[Volver a Contenidos](#)

## ✓ 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.

## ✓ Estilo del encabezado

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

```
# Muestra los nombres de las columnas  
print(df.columns)
```

```
↩ 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:

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:

```
# Bucle en los encabezados poniendo todo en minúsculas
```

```
new_name_col=[] #se crea lista para almacenar los nuevos nombres
for old_name in df.columns: #bucle para iterar sobre los nombres
    name_lowered = old_name.lower() #nueva variable con los cambios
    name_stripped= name_lowered.strip()
    new_name_col.append(name_stripped)
df.columns=new_name_col
```

```
# Bucle en los encabezados eliminando los espacios
print(df.columns)
```

```
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.

```
# Cambiar el nombre de la columna "userid"
```

```
df.rename(columns={"userid":"user_id"}, inplace=True)
print(df.columns[0])
```

```
user_id
```

Comprueba el resultado. Muestra los encabezados una vez más:

```
# Comprobar el resultado: la lista de encabezados
```

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

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

[Volver a Contenidos](#)

## ✓ 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.

```
# Calcular el número de 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 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 cálculos;

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.

```
# Bucle en los encabezados reemplazando los valores ausentes con 'unknown'
columns_to_replace=["track","artist","genre"]
for col in columns_to_replace:
    df[col].fillna("unknown", inplace=True)
```

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.

```
# Contar 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

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.

```
# Contar duplicados explícitos

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

```
→ 3826
```

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

```
# Eliminar duplicados explícitos
df=df.drop_duplicates().reset_index(drop=True)
```

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

```
# Comprobar de nuevo si hay duplicados

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

```
→ 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.

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

```
→ ['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' 'loungeselectronic' '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' 'iii' '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'
   'tarafar' 'gypsy' 'hardtechno' 'shoegazing' 'bossa' 'latino' 'worldbeat'
   'malaysian' 'baile' 'ghazal' 'arabic' 'popelectronic' 'acid' 'kayokyoku'
   'neoklassik' 'tribal' 'tanzorchester' 'native' 'independent' 'cantautori'
   'handsup' 'punjabi' 'synthpop' 'rave' 'französisch' 'quebécois' '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, reemplazar 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.

```
# Función para reemplazar duplicados implícitos
def replace_wrong_genres(column, wrong_genres, correct_genre): #funcion para reemplazar los v
    for wrong_value in wrong_genres:
        df[column]=df[column].replace(wrong_genre, correct_genre)
    return df
```

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`:

```
# Eliminar duplicados implícitos
wrong_genres=["hip","hop","hip-hop"] #valores incorrectos
correct_genre=["hiphop"] #valor correcto
df= replace_wrong_genres("genre","duplicates","hiphop")
```

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

```
# Comprobación de duplicados implícitos
print(df["genre"].unique())
```

```
➡ ['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' 'loungeselectronic' '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' 'ïïï' '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](#)

## ✓ Tus observaciones

Describe brevemente lo que has notado al analizar duplicados, cómo abordaste sus eliminaciones y qué resultados obtuviste.

Se encontraron valores duplicados en las columnas genere, resultando tres casos para el genero Hiphop (hip, hop, hip\_hop). Se creo una función para reemplazar dichos valores duplicados por "hiphop", asignando los valores a cambiar en una variable llamada "wrong\_genre". Ejecutando dicha funcion obtuvimos una columna con valores libres de duplicados implícitos.

[Volver a Contenidos](#)

## ✓ Etapa 3. Prueba de hipótesis

## ✓ 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.

```
# Contar las canciones reproducidas en cada ciudad
print(df.groupby(by="city")["track"].count())
```

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

Comenta tus observaciones aquí Al aplicar el método de agrupamiento por ciudad podemos deducir que los usuarios en Springfield escuchan el doble de canciones a comparación que los usuarios de 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.

```
# Calcular las canciones reproducidas en cada uno de los tres días
print(df.groupby(by="day")["track"].count())
```

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

Comenta tus observaciones aquí Con los resultados obtenidos por la agrupacion de datos, obtenemos que los días viernes son los días con mas reproducciones. Y en ultimo lugar tenemos a los dias miercoles.

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.

```
def number_tracks(day,city):# Declara la función number_tracks() con dos parámetros: day= y
    filtered_day= df[df["day"]==day] # Almacena las filas del DataFrame donde el valor en la
    filtered_city= filtered_day[filtered_day["city"]==city] # Filtra las filas donde el valor
    played_tracks=filtered_city["user_id"].count() # Extrae la columna 'user_id' de la tabla
    return played_tracks # Devuelve 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.


```
# El número de canciones reproducidas en Springfield el lunes
print(number_tracks("Monday","Springfield"))
```

 15740

```
# El número de canciones reproducidas en Shelbyville el lunes  
print(number_tracks("Monday","Shelbyville"))
```

 5614


```
# El número de canciones reproducidas en Springfield el miércoles  
print(number_tracks("Wednesday","Springfield"))
```

 11056


```
# El número de canciones reproducidas en Shelbyville el miércoles  
print(number_tracks("Wednesday","Shelbyville"))
```

 7003

```
# El número de canciones reproducidas en Springfield el viernes  
print(number_tracks("Friday","Springfield"))
```

 15945

```
# El número de canciones reproducidas en Shelbyville el viernes  
print(number_tracks("Friday","Shelbyville"))
```

 5895

## Conclusiones

Comenta si la hipótesis es correcta o se debe rechazar. Explica tu razonamiento. La hipótesis se aprueba, los usuarios de cada ciudad tienen diferentes comportamientos en el consumo de la música. Por un lado:

- Los días lunes: En Springfield es el segundo día que más se consume música, sin embargo, en el caso de Shelbyville es el día que menos se consume,
- Los días miércoles: En Springfield es el día de menor consumo de música, por otro lado, en Shelbyville es el día que más se consume música.
- Los días viernes: En Springfield es el día de mayor consumo de música, mientras que, en Shelbyville es el segundo día con más consumo. En conclusión, con los comentarios anteriores se puede ver una clara diferencia en los días que consumen música los usuarios de las diferentes ciudades. Se debería realizar un estudio a mayor profundidad para identificar algunos otros factores, por ejemplo: si la duración de las canciones tiene un impacto en la cantidad de canciones escuchadas por usuario.

[Volver a Contenidos](#)

## ✓ Conclusiones

Resume aquí tus conclusiones sobre la hipótesis. En conclusión, la hipótesis se cumplió, al agrupar datos y analizar el comportamiento de los usuarios obtuvimos una diferencia en la tendencia de consumo de música como se planteaba en la hipótesis, se requiere un análisis más profundo para identificar posibles sesgos en nuestro primer análisis, debido a que los datos de la tabla carecen de valores cualitativos y cuantitativos que nos pudieran dar un resultado más preciso, con respecto a la prueba de nuestra hipótesis.

## ✓ Nota

En proyectos de investigación reales, la prueba de hipótesis estadística es más precisa y cuantitativa. También ten en cuenta que no siempre se pueden sacar conclusiones sobre una ciudad entera a partir de datos de una sola fuente.

Aprenderás más sobre la prueba de hipótesis en el sprint de análisis estadístico de datos.

[Volver a Contenidos](#)

Comienza a programar o [generar](#) con IA.