**Se liga na música**

**Conteúdo**

* [Introdução](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#intro)
* [Etapa 1. Visão geral dos dados](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#data_review)
  + [Conclusões](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#data_review_conclusions)
* [Etapa 2. Pré-processamento de dados](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#data_preprocessing)
  + [2.1 Estilo do cabeçalho](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#header_style)
  + [2.2 Valores ausentes](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#missing_values)
  + [2.3 Duplicados](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#duplicates)
  + [2.4 Conclusões](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#data_preprocessing_conclusions)
* [Etapa 3. Teste da hipótese](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#hypothesis)
  + [3.1 Hipótese 1: atividade dos usuários nas duas cidades](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#activity)
* [Conclusões](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#end)

**1  Introdução**

O trabalho de um analista é analisar dados para obter percepções valiosas dos dados e tomar decisões fundamentadas neles. Esse processo consiste em várias etapas, como visão geral dos dados, pré-processamento dos dados e testes de hipóteses.

Sempre que fazemos uma pesquisa, precisamos formular uma hipótese que depois poderemos testar. Às vezes nós aceitamos essas hipóteses; outras vezes, nós as rejeitamos. Para fazer as escolhas certas, um negócio deve ser capaz de entender se está fazendo as suposições certas ou não.

Neste projeto, você vai comparar as preferências musicais dos habitantes de Springfild e Shelbyville. Você vai estudar os dados de um serviço de streaming de música online para testar a hipótese apresentada abaixo e comparar o comportamento dos usuários dessas duas cidades.

**1.1  Objetivo:**

Teste a hipótese:

1. A atividade dos usuários é diferente dependendo do dia da semana e da cidade.

**1.2  Etapas**

Os dados sobre o comportamento do usuário são armazenados no arquivo /datasets/music\_project\_en.csv. Não há informações sobre a qualidade dos dados, então será necessário examiná-los antes de testar a hipótese.

Primeiro, você avaliará a qualidade dos dados e verá se seus problemas são significativos. Depois, durante o pré-processamento dos dados, você tentará tratar dos problemas mais críticos.

O seu projeto consistirá em três etapas:

1. Visão geral dos dados
2. Pré-processamento de dados
3. Teste da hipótese

[Voltar ao Índice](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#back)

**2  Etapa 1. Visão geral dos dados**

Abra os dados e examine-os.

Você precisará da pandas, então, importe-a.

In [1]:

**import** pandas **as** pd *# importando pandas*

​

Leia o arquivo music\_project\_en.csv da pasta /datasets/ e salve-o na variável df:

In [2]:

df **=** pd.read\_csv('/datasets/music\_project\_en.csv') *# lendo o arquivo e armazenando em df*

​

Imprima as primeiras 10 linhas da tabela:

In [3]:

print(df.head(10)) *# obtenha as 10 primeiras 10 linhas da tabela 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

Obtenha informações gerais sobre a tabela usando um comando. Você conhece o método para exibir informações gerais que precisamos obter.

In [4]:

print(df.info())

*# obtendo informações gerais sobre os nossos dados*

​

<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

Aqui estão as nossas observações sobre a tabela. Ela contém sete colunas. Elas armazenam o mesmo tipo de dado: object.

De acordo com a documentação:

* ' userID' — identificação do usuário
* 'Track' — título da música
* 'artist' — nome do artista
* 'genre' — gênero da música
* 'City' — cidade do usuário
* 'time' — o tempo exato que a música foi reproduzida
* 'Day' — dia da semana

Podemos ver três problemas de estilo nos cabeçalhos da tabela:

1. Alguns cabeçalhos são escritos em letras maiúsculas, outros estão em minúsculas.
2. Alguns cabeçalhos contêm espaços.
3. Detecte o problema e o descreva aqui.

**2.1  Escreva suas observações. Aqui estão algumas perguntas que podem ajudar:**

1. Que tipo de dados temos nas linhas? E como podemos entender as colunas? As linhas trazem a identificação do usuário, total tocado da música, nome do artista, gênero e track da música. As colunas são do tipo object, exceto para a coluna total play que é um número flutuante (float).

2. Esses dados são suficientes para responder à nossa hipótese ou precisamos de mais dados? Se formos analisar o comportamento do usuário quanto ao tipo de música ouvida por gênero e artista e o tempo de cada uma, é possível sim utilizando os dados fornecidos.

3. Você notou algum problema nos dados, como valores ausentes, duplicados ou tipos de dados errados Há um total de 7 colunas de acordo com a documentação, no entanto na leitura geral dos dados não estão presentes as linhas referente a 'city' 'time' and 'day'. Sim, há valores duplicados e errados, como aparece na tabela a indicação de "NAN".

**3  Etapa 2. Pré-processamento de dados**

O objetivo aqui é preparar os dados para a análise. O primeiro passo é resolver todos os problemas com o cabeçalho. E então podemos passar para os valores ausentes e duplicados. Vamos começar.

Corrija a formatação nos cabeçalhos da tabela.

**3.1  Estilo do cabeçalho**

Imprima os cabeçalhos da tabela (os nomes das colunas):

In [32]:

print(df.columns) *# imprima os nomes das colunas*

​

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

Mude os cabeçalhos da tabela conforme as boas práticas de estilo:

* Todos os caracteres precisam estar com letras minúsculas
* Exclua espaços
* Se o nome tiver várias palavras, use snake\_case

Anteriormente, você aprendeu sobre uma maneira automatizada de renomear colunas. Vamos usá-la agora. Use o ciclo for para percorrer os nomes das colunas e transformar todos os caracteres em letras minúsculas. Após fazer isso, imprima os cabeçalhos da tabela novamente:

In [2]:

**import** pandas **as** pd

​

df **=** pd.read\_csv('/datasets/music\_project\_en.csv')

​

columns\_new **=** []

**for** old\_name **in** df.columns:

name\_lower **=** old\_name.lower()

columns\_new.append(name\_lower)

df.columns **=** columns\_new

​

print(df.columns)

*# Percorrendo os cabeçalhos e convertendo tudo em minúsculos*

​

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

Agora, usando a mesma abordagem, exclua os espaços no início e no final de cada nome de coluna e imprima os nomes das colunas novamente:

In [3]:

columns\_new **=** []

**for** old\_name **in** df.columns:

name\_cleaned **=** old\_name.strip()

columns\_new.append(name\_cleaned)

df.columns **=** columns\_new

​

print(df.columns)

​

*# Percorrendo os cabeçalhos e removendo os espaços*

​

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

Precisamos aplicar a regra de sublinhado no lugar de espaço à coluna userid. Deveria ser user\_id. Renomeie essa coluna e imprima os nomes de todas as colunas quando terminar.

In [4]:

columns\_new **=** []

**for** old\_name **in** df.columns:

name\_corrected **=** old\_name.replace('userid','user\_id')

columns\_new.append(name\_corrected)

df.columns **=** columns\_new

​

print(df.columns)

*# Renomeando a coluna "userid"*

​

Index(['user\_id', 'track', 'artist', 'genre', 'city', 'time', 'day'], dtype='object')

Verifique o resultado. Imprima os cabeçalhos novamente:

In [5]:

print(df.columns) *# verificando o resultado: a lista de cabeçalhoss*

​

Index(['user\_id', 'track', 'artist', 'genre', 'city', 'time', 'day'], dtype='object')

[Voltar ao Índice](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#back)

**Comentário do revisor:**

* Precisa usar loop para iterar conforme solicitado. Exemplo: for col in ['A', ' B ']:
* Precisa suar funções como replace, strip, lower
* Está importando e lendo csv sem necssidade; uma vez basta

**3.2  Valores Ausentes**

Primeiro, encontre a quantidade de valores ausentes na tabela. Você precisa usar dois métodos em sequência para obter o número de valores ausentes.

In [10]:

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

​

*# calculando o número de valores ausentes*

​

user\_id 0

track 1343

artist 7567

genre 1198

city 0

time 0

day 0

dtype: int64

Nem todos os valores ausentes afetam a pesquisa. Por exemplo, os valores ausentes em track e artist não são críticos. Você pode simplesmente substituí-los por valores padrão, como a string 'unknown'.

Mas valores ausentes em 'genre' podem afetar a comparação de preferências musicais de Springfield e Shelbyville. Na vida real, seria útil descobrir as razões pelas quais os dados estão ausentes e tentar corrigi-los. Mas nós não temos essa possibilidade neste projeto. Então, você terá que:

* Preencha esses valores ausentes com um valor padrão
* Avalie em que medida os valores ausentes podem afetar sua análise

Substitua os valores ausentes nas colunas 'track', 'artist' e 'genre' pela string 'unknown'. Como mostramos nas lições anteriores, a melhor maneira de fazer isso é criar uma lista para armazenar os nomes das colunas nas quais precisamos fazer a substituição. Em seguida, use essa lista e percorra as colunas nas quais a substituição seja necessária e faça a substituição.

In [7]:

columns\_to\_replace **=** ['track', 'artist', 'genre']

**for** col **in** columns\_to\_replace:

df[col].fillna('unknown', inplace**=True**)

*# percorrendo os cabeçalhos e substituindo valores ausentes por 'unknown'*

​

**Comentário do revisor:**

Precisa usar loop conforme solicitado

Agora verifique o resultado para ter certeza de que o conjunto de dados não contenha valores ausentes após a substituição. Para fazer isso, conte os valores ausentes novamente.

In [12]:

print(df.isna().sum())*# contando os valores ausentes*

​

user\_id 0

track 0

artist 0

genre 0

city 0

time 0

day 0

dtype: int64

[Voltar ao Índice](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#back)

**3.3  Duplicados**

Encontre o número de duplicados explícitos na tabela. Lembre-se de que você precisa aplicar dois métodos em sequência para obter o número de duplicados explícitos.

In [13]:

**import** pandas **as** pd

df **=** pd.read\_csv('/datasets/music\_log\_raw.csv')

​

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

​

*# contando duplicados explícitos*

​

5313

Agora descarte todos os duplicados. Para fazer isso, chame o método que faz exatamente isso.

In [14]:

df.drop\_duplicates(inplace**=True**)

*# removendo duplicados explícitos*

​

Agora vamos verificar se descartamos todos os duplicados. Conte duplicados explícitos mais uma vez para ter certeza de que você removeu todos eles:

In [15]:

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

*# verificando duplicados novamente*

​

​

0

Agora queremos nos livrar dos duplicados implícitos na coluna genre. Por exemplo, o nome de um gênero pode ser escrito de maneiras diferentes. Alguns erros afetarão também o resultado.

Para fazer isso, vamos começar imprimindo uma lista de nomes de gênero únicos, ordenados em ordem alfabética: Para fazer isso:

* Extraia a coluna genre do DataFrame
* Chame o método que retornará todos os valores únicos na coluna extraída

In [16]:

print(df['genre'].unique()) *# visualizando nomes de gêneros únicos*

​

['pop' 'ambient' 'dance' 'jazz' 'classicmetal' 'electronic' 'indie'

'hiphop' nan 'spoken' 'new' 'latin' 'extrememetal' 'instrumental'

'classical' 'alternative' 'rock' 'german' 'french' 'metal' 'dubstep'

'house' 'miscellaneous' 'rap' 'world' 'country' 'punk' 'rusrap' 'rnb'

'beats' 'ukrrock' 'inspiritual' 'ruspop' 'caucasian' 'rusrock' 'dub'

'soundtrack' 'folk' 'shanson' 'fairytail' 'hard-n-heavy' 'romance'

'religious' 'hardcore' 'orchestral' 'minimal' 'film' 'spiritual'

'melodic' 'trance' 'comedy' 'reggae' 'deep' 'mpb' 'techno' 'reggaeton'

'singer' 'karaoke' 'children' 'adult' 'western' 'psychedelic' 'grime'

'christian' 'holiday' 'argentinetango' 'disco' 'lounge' 'urban' 'local'

'progressive' 'other' 'funk' 'blues' 'easy' 'dancehall' 'tatar' 'conjazz'

'drum' 'chill' 'jpop' 'fitness' 'gospel' 'brazilian' 'vocal' 'chanson'

'gothic' 'irish' 'k-pop' 'acoustic' 'industrial' 'numetal' 'soul'

'experimental' 'relax' 'mexican' 'videogame' 'glitch' 'worldmusic'

'postrock' 'folkmetal' 'bossa' 'black' 'tech' 'schlager' 'chinese'

'educational' 'russian' 'international' "hard'n'heavy" 'animated'

'americana' 'mass' 'downtempo' 'club' 'prog' 'bigroom' 'avantgarde'

'tropical' 'samba' 'jewish' 'epicmetal' 'halloween' 'middle' 'old'

'downbeat' 'gangsta' 'broadway' 'european' 'modern' 'opera' 'türkçe'

'estrada' 'deutschrock' 'posthardcore' 'progmetal' 'axé' 'sport' 'art'

'indian' 'future' 'ska' 'grunge' 'salsa' 'dark' 'synthrock' 'ethnic'

'folkrock' 'trip' 'asiapop' 'speech' 'bassline' 'funky' 'bluegrass'

'fusion' 'chillhouse' 'breaks' 'death' 'thrash' 'power' 'variété' 'post'

'celtic' 'rnr' 'self' 'nu' 'caribbean' 'balkan' 'cantopop' 'breakbeat'

'indonesia' 'regional' 'piano' 'inspirational' 'southern' 'traditional'

'mood' 'rockabilly' 'action' 'uzbek' 'contemporary' 'christmas' 'fado'

'solo' 'handsup' 'electropop' 'canzone' 'lecture' 'bollywood' 'dancepop'

'anime' 'korean' 'eurofolk' 'audiobook' 'garage' 'smooth' 'newage' 'top'

'rancheras' 'native' 'tradjazz' 'cuba' 'arabesk' 'synthpop' 'boleros'

'rhythm' 'swing' 'african' 'folklore' 'colombian' 'stonerrock' 'jam'

'specialty' 'quebecois' 'i̇slami' 'wort' 'afro' 'sertanejo' 'afrikaans'

'baile' 'eastern' 'synthi' 'surf' 'gabba' 'italo' 'tango' 'französisch'

'muslim' 'электроника' 'symphonic' 'dnb' 'japanese' 'cuban' 'acid'

'cantautori' 'idm' 'general' 'devotional' 'skarock' 'yoga' 'mandopop'

'bass' 'flamenco' 'türk' 'jiddisch' 'punjabi' 'symphony' 'entehno'

'latino' 'swingbeat' 'leftfield' 'rhythmic' 'hardstyle' 'crunk'

'hardtrance' 'goth' 'newwave' 'concerto' 'musique' 'meditative' 'bacheta'

'requiem' 'metalcore' 'freestyle' 'standards' 'emo' 'choral' 'drone'

'nujazz' 'roots' 'jumpstyle' 'merengue' 'chamber' 'dirty' 'kürtçe'

'oceania' 'soulful' 'gypsy' 'afrika' 'indipop' 'hymn' 'cajun' 'turbo'

'psychobilly' 'trumpet' 'broken' 'kayokyoku' 'doom' 'polka' 'blip'

'orgel' 'ranchera' 'deutscherock' 'retro' 'oratorio' 'alternativepunk'

'ukr' 'bolero' 'dj' 'acapella' 'djent' 'independent' 'şiir' 'shoegazing'

'rave']

Olhe a lista e encontre duplicados implícitos do gênero hiphop. Esses podem ser nomes escritos incorretamente, ou nomes alternativos para o mesmo gênero.

Você verá os seguintes duplicados implícitos:

* hip
* hop
* hip-hop

Para se livrar deles, crie uma função replace\_wrong\_genres() com dois parâmetros:

* wrong\_genres= — essa é uma lista que contém todos os valores que você precisa substituir
* correct\_genre= — essa é uma string que você vai usar para a substituição

Como resultado, a função deve corrigir os nomes na coluna 'genre' da tabela df, isto é, substituindo cada valor da lista wrong\_genres por valores de correct\_genre.

Dentro do corpo da função, use um ciclo 'for' para percorrer a lista de gêneros errados, extrair a coluna 'genre' e aplicar o método replace para fazer as correções.

In [22]:

**def** replace\_wrong\_genres(df, column, wrong\_genres, correct\_genre):

**for** wrong\_genre **in** wrong\_genres:

df[column] **=** df[column].replace(wrong\_genre, correct\_genre)

**return** df

​

print(df)

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 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]

Agora, chame a função replace\_wrong\_genres() e passe argumentos apropriados para que ela limpe duplicados implícitos (hip, hop e hip-hop) substituindo-os por hiphop:

In [18]:

df **=** df.drop\_duplicates(subset**=**['genre'])

print(df)

user\_id total play Artist \

0 BF6EA5AF 92.851388 Marina Rei

1 FB1E568E 282.981000 Stive Morgan

3 EF15C7BA 8.966000 NaN

5 4166D680 3.007000 Henry Hall & His Gleneagles Hotel Band

6 F4F5677 0.100000 NaN

... ... ... ...

62093 58B65057 14.899127 Mr.Taylor

62895 AA5C3449 6.212761 Adam Barber

64762 72984E9 2.043000 İsmail Hakkı Akgün

65229 7FAAD332 2.875000 Slowdive

65993 E0987B68 154.122449 Dr. Onionskin

genre track

0 pop Musica

1 ambient Love Planet

3 dance Loving Every Minute

5 jazz Home

6 classicmetal NaN

... ... ...

62093 djent Fractal

62895 independent Love Is Blind

64762 şiir Evvel Benem Ahir Benem

65229 shoegazing Souvlaki Space Station

65993 rave & Thats How Its Going To B

[305 rows x 5 columns]

Certifique-se que os nomes duplicados foram removidos. Imprima a lista de valores únicos da coluna 'genre' mais uma vez:

In [24]:

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

['rock' 'pop' 'folk' 'dance' 'rusrap' 'ruspop' 'world' 'electronic' nan

'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']

**Comentário do revisor:**

Precisa usar print(df['genre'].unique())

[Voltar ao Índice](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#back)

**3.4  Suas observações**

Descreva brevemente o que você reparou ao analisar duplicados, bem como a abordagem que usou para eliminá-los e os resultados que alcançou.

[Voltar ao Índice](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#back)

**4  Etapa 3. Teste da hipótese**

**4.1  Hipótese: comparação do comportamento dos usuários nas duas cidades**

A hipótese afirma que existem diferenças no consumo de música pelos usuários em Springfield e em Shelbyville. Para testar a hipótese, use os dados dos três dias da semana: segunda-feira (Monday), quarta-feira (Wednesday) e sexta-feira (Friday).

* Agrupe os usuários por cidade.
* Compare o número de músicas tocadas por cada grupo na segunda, quarta e sexta.

Execute cada cálculo separadamente.

O primeiro passo é avaliar a atividade dos usuários em cada cidade. Não se esqueça das etapas "divisão-aplicação-combinação" sobre as quais falamos anteriormente na lição. Agora seu objetivo é agrupar os dados por cidade, aplicar o método de contagem apropriado durante a etapa de aplicação e então encontrar o número de músicas tocadas por cada grupo, especificando a coluna para a qual você quer obter a contagem.

Veja um exemplo de como o resultado final deve ser: df.groupby(by='....')['column'].method() Execute cada cálculo separadamente.

Para avaliar a atividade dos usuários em cada cidade, agrupe os dados por cidade e encontre o número de músicas reproduzidas em cada grupo.

In [8]:

groups\_by\_city **=** df.groupby('city')['user\_id'].count()

print(groups\_by\_city)

​

*# Contando as músicas tocadas em cada cidade*

​

city

Shelbyville 19719

Springfield 45360

Name: user\_id, dtype: int64

**Comentário do revisor:**

* código repetido e desnecessário: leitura importação

In [26]:

*#Há mais pessoas pessoas que ouvem música em Springfield do que em Shelbyville.*

Comente sobre suas observações aqui

Agora vamos agrupar os dados por dia da semana e encontrar a quantidade de músicas tocadas na segunda, quarta e sexta-feira. Use a mesma abordagem que antes, mas agora precisamos agrupar os dados de uma forma diferente.

In [9]:

days\_of\_the\_week **=** df.groupby('day')['city'].count()

print(days\_of\_the\_week)

​

*# Calculando as músicas escutadas em cada um desses três dias*

​

day

Friday 23149

Monday 22697

Wednesday 19233

Name: city, dtype: int64

**Comentário do revisor:**

* código repetido e desnecessário: leitura importação

Você acabou de aprender como contar entradas agrupando-as por cidade ou por dia. E agora você precisa escrever uma função que possa contar entradas simultaneamente com base em ambos os critérios.

Crie a função number\_tracks() para calcular o número de músicas tocadas em um determinado dia **e** em uma determinada cidade. A função deve aceitar dois parâmetros:

* day: um dia da semana pelo qual precisamos filtrar os dados. Por exemplo, 'Monday'.
* city: uma cidade pela qual precisamos filtrar os dados. Por exemplo, 'Springfield'.

Dentro da função, você vai aplicar uma filtragem consecutiva com indexação lógica.

Primeiro, filtre os dados por dia e então filtre a tabela resultante por cidade.

Depois de filtrar os dados usando os dois critérios, conte o número de valores na coluna 'user\_id' da tabela resultante. O resultado da contagem representará o número de entradas que você quer encontrar. Armazene o resultado em uma nova variável e imprima-o.

In [28]:

*# Declare a função number\_tracks() com dois parâmetros: day= e city=.*

​

*# Armazene as linhas do DataFrame em que o valor na coluna 'day' é igual ao parâmetro day=*

​

*# Filtre as linhas em que o valor na coluna 'city' é igual ao parâmetro city=*

​

*# Extraia a coluna 'user\_id' da tabela filtrada e aplique o método count()*

​

*# Retorne o número dos valores da coluna 'user\_id'*

​

**def** number\_tracks(day, city):

day\_filtered **=** df[df['day'] **==** day]

city\_filtered **=** day\_filtered[day\_filtered['city'] **==** city]

user\_id\_filtered **=** city\_filtered['user\_id'].count()

**return** user\_id\_filtered

​

print(number\_tracks)

<function number\_tracks at 0x7fe50dbcac10>

Chame a função number\_tracks() seis vezes, mudando os valores dos parâmetros, para que você possa recuperar os dados de ambas as cidades para cada um dos três dias.

In [29]:

*# a quantidade de músicas tocadas em Springfield na segunda-feira*

​

number\_tracks('Monday', 'Springfield')

Out[29]:

16715

In [30]:

number\_tracks('Monday', 'Shelbyville')

*# a quantidade de músicas tocadas em Shelbyville na segunda-feira*

​

Out[30]:

5982

In [31]:

number\_tracks('Wednesday', 'Springfield')

​

*# a quantidade de músicas tocadas em Springfield na quarta-feira*

​

Out[31]:

11755

In [32]:

number\_tracks('Wednesday', 'Shelbyville')

​

*# a quantidade de músicas tocadas em Shelbyville na quarta-feira*

​

Out[32]:

7478

In [33]:

number\_tracks('Friday', 'Springfield')

​

*# a quantidade de músicas tocadas em Springfield na sexta-feira*

​

Out[33]:

16890

In [34]:

number\_tracks('Friday', 'Shelbyville')

​

*# a quantidade de músicas tocadas em Shelbyville na sexta-feira*

​

Out[34]:

6259

In [36]:

*#a análise podemos perceber que os usuários de Springfield ouvem mais músicas que a cidade de Shelbyville, considerando o total de músicas tocadas. No entanto, há uma diferença de comportamento entre as cidades. Springfield ouve mais múiscas às sextas-feiras com um total de 16.715 tracks, seguida da segunda-feira com uma quantidade de 16.715 tracks. Apesar de Shelbyville possuir um total de 6.259 tocadas às sextas-feiras, a maior audiência se concentra nas quartas-feiras com um total de 7.478 tracks tocadas.*

**Conclusões**

Comente sobre se a terceira hipótese está correta ou deve ser rejeitada. Explique seu raciocínio.

[Voltar ao Índice](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#back)

**Conclusões**

In [ ]:

Na análise podemos perceber que os usuários de Springfield ouvem mais músicas que a cidade de Shelbyville, considerando o total de músicas tocadas. No entanto, há uma diferença de comportamento entre **as** cidades. Springfield ouve mais múiscas às sextas**-**feiras com um total de 16.715 tracks, seguida da segunda**-**feira com uma quantidade de 16.715 tracks. As cidades de Springfield e Shelbyville possuem hábitos diferentes ao relação ao volume de músicas tocadas e dia da semana, confirmando a hipótese de diferença de comportamento entre essas cidades.

Resuma suas conclusões sobre a hipótese aqui

**4.2  Importante**

Em projetos de pesquisas reais, o teste estatístico de hipóteses é mais preciso e quantitativo. Observe também que conclusões sobre uma cidade inteira nem sempre podem ser tiradas a partir de dados de apenas uma fonte.

Você aprenderá mais sobre testes de hipóteses no sprint sobre a análise estatística de dados.

[Voltar ao Índice](https://jupyterhub.tripleten-services.com/user/user-3-9f333257-ff23-47e4-86e4-b6f3e46fa00b/notebooks/20c53b4a-81d8-4d18-bc38-d36d30ae6fba-Copy1.ipynb#back)