20c53b4a-81d8-4d18-bc38-d36d30ae6fba

May 9, 2025

1 Se liga na música

2 Conteúdo

- Introdução
- Etapa 1. Visão geral dos dados
  - Conclusões
- Etapa 2. Pré-processamento de dados
  - 2.1 Estilo do cabeçalho
  - 2.2 Valores ausentes
  - 2.3 Duplicados
  - 2.4 Conclusões
- Etapa 3. Teste da hipótese
  - 3.1 Hipótese 1: atividade dos usuários nas duas cidades
- Conclusões

2.1 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.

2.1.1 Objetivo:

Teste a hipótese: 1. A atividade dos usuários é diferente dependendo do dia da semana e da cidade.

2.1.2 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

2.2 Etapa 1. Visão geral dos dados

Abra os dados e examine-os.

Você precisará da `pandas`, então, importe-a.

[1]:
```
import pandas as pd
```

Leia o arquivo `music_project_en.csv` da pasta `/datasets/` e salve-o na variável `df`:

[2]:
```
    df = pd.read_csv('/datasets/music_project_en.csv')
```

Imprima as primeiras 10 linhas da tabela:

[3]:
```
    print(df.head(10))
```

```
     userID                        Track            artist   genre  \
0  FFB692EC            Kamigata To Boots  The Mass Missile    rock
1  55204538  Delayed Because of Accident  Andreas Rönnberg    rock
2    20EC38            Funiculì funiculà       Mario Lanza     pop
3  A3DD03C9        Dragons in the Sunset        Fire + Ice    folk
4  E2DC1FAE                   Soul People        Space Echo   dance
5  842029A1                        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.

[4]:
```
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
```

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`. *Userid sem padrão snake_case, presença de valores nulos/falatantes/Nan/Na nas colunas genre, track e artist*

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

`1. Que tipo de dados temos nas linhas? E como podemos entender as colunas?` *Identificação do usuário, título de música, nome da música, gênero, cidade dos usuários, tempo da música e dia que música tomou. Através da descrição de cada título.* `2. Esses dados são suficientes para responder à nossa hipótese ou precisamos de mais dados?` *Precisamos de mais dados, quanto mais soubermos a respeito do "assunto" ou "informação", melhor para a aplicação da estratégias* `3. Você notou algum problema nos dados, como valores ausentes, duplicados ou tipos de dados errados` *Sim, não apresenta padronização, organização, e falta de dados.*

Voltar ao Índice

2.3 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.

2.3.1 Estilo do cabeçalho

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

[5]:
```
print(df.columns)
```

```
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:

[6]:
```
new_column_names = [col.lower() for col in df.columns]

df.columns = new_column_names

print(df.columns)
```

```
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:

[7]:
```
df.columns = df.columns.str.strip()

print(df.columns)
```

```
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.

[8]:
```
df.rename(columns={'userid': 'user_id'}, inplace=True)

print(df.columns)
```

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

Comentário do revisor V2:

Correto

Verifique o resultado. Imprima os cabeçalhos novamente:

[9]:
```
 print(df.columns)
```

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

Voltar ao Índice

2.3.2 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.

[10]:
```
missing_values_count = df.isnull().sum()

print(missing_values_count)
```

```
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.

[11]:
```
cols_to_fill = ['track' , 'artist' , 'genre']

for col in cols_to_fill:
    df[col].fillna('unknown', inplace=True)

print(df.isnull().sum())
```

```
user_id    0
track      0
artist     0
genre      0
city       0
```

```
time       0
day        0
dtype: int64
```

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.

[12]:
```
print(df.isnull().sum())
```

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

Voltar ao Índice

2.3.3 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.

[13]:
```
duplicates_count = df.duplicated().sum()


print(f"O número de duplicados explícitos na tabela é: {duplicates_count}")
```

```
O número de duplicados explícitos na tabela é: 3826
```

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

[14]:
```
df.drop_duplicates(inplace=True)
```

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

[15]:
```
print(f"O número de linhas após remover os duplicados é: {len(df)}")
```

```
O número de linhas após remover os duplicados é: 61253
```

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

[16]:
```
genres = df['genre']
wrong_genres = genres.unique()
correct_genres = sorted(wrong_genres)
print(correct_genres)
```

```
['acid', 'acoustic', 'action', 'adult', 'africa', 'afrikaans', 'alternative',
'ambient', 'americana', 'animated', 'anime', 'arabesk', 'arabic', 'arena',
'argentinetango', 'art', 'audiobook', 'avantgarde', 'axé', 'baile', 'balkan',
'beats', 'bigroom', 'black', 'bluegrass', 'blues', 'bollywood', 'bossa',
'brazilian', 'breakbeat', 'breaks', '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', 'deutschrock',
'deutschspr', 'dirty', 'disco', 'dnb', 'documentary', 'downbeat', 'downtempo',
'drum', 'dub', 'dubstep', 'eastern', 'easy', 'electronic', 'electropop', 'emo',
'entehno', 'epicmetal', 'estrada', 'ethnic', 'eurofolk', 'european',
'experimental', 'extrememetal', 'fado', 'film', 'fitness', 'flamenco', 'folk',
'folklore', 'folkmetal', 'folkrock', 'folktronica', 'forró', 'frankreich',
'französisch', 'french', 'funk', 'future', 'gangsta', 'garage', 'german',
'ghazal', 'gitarre', 'glitch', 'gospel', 'gothic', 'grime', 'grunge', 'gypsy',
'handsup', "hard'n'heavy", 'hardcore', 'hardstyle', 'hardtechno', 'hip', 'hip-
hop', 'hiphop', 'historisch', 'holiday', 'hop', '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', 'loungeelectronic',
'lovers', 'malaysian', 'mandopop', 'marschmusik', 'meditative', 'mediterranean',
'melodic', 'metal', 'metalcore', 'mexican', 'middle', 'minimal',
'miscellaneous', 'modern', 'mood', 'mpb', 'muslim', 'native', 'neoklassik',
'neue', 'new', 'newage', 'newwave', 'nu', 'nujazz', 'numetal', 'oceania', 'old',
'opera', 'orchestral', 'other', 'piano', 'pop', 'popelectronic', 'popeurodance',
'post', 'posthardcore', 'postrock', 'power', 'progmetal', 'progressive',
'psychedelic', 'punjabi', 'punk', 'quebecois', 'ragga', 'ram', 'rancheras',
'rap', 'rave', 'reggae', 'reggaeton', 'regional', 'relax', 'religious', 'retro',
'rhythm', 'rnb', 'rnr', 'rock', 'rockabilly', 'romance', 'roots', 'ruspop',
'rusrap', 'rusrock', 'salsa', 'samba', 'schlager', 'self', 'sertanejo',
'shoegazing', 'showtunes', 'singer', 'ska', 'slow', 'smooth', 'soul', 'soulful',
'sound', 'soundtrack', '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',
```

```
'unknown', 'urban', 'uzbek', 'variété', 'vi', 'videogame', 'vocal', 'western',
'world', 'worldbeat', 'ïîï']
```

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.

[17]:
```
# função para substituir 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)
```

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`:

[18]:
```
# removendo duplicados implícitos
wrong_genre = ['hip', 'hop', 'hip-hop']
correct_genre = 'hiphop'
replace_wrong_genres(wrong_genre, correct_genre)
```

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

[19]:
```
# verificando valores duplicados
print(sorted(df['genre'].unique()))
```

```
['acid', 'acoustic', 'action', 'adult', 'africa', 'afrikaans', 'alternative',
'ambient', 'americana', 'animated', 'anime', 'arabesk', 'arabic', 'arena',
'argentinetango', 'art', 'audiobook', 'avantgarde', 'axé', 'baile', 'balkan',
'beats', 'bigroom', 'black', 'bluegrass', 'blues', 'bollywood', 'bossa',
'brazilian', 'breakbeat', 'breaks', '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', 'deutschrock',
'deutschspr', 'dirty', 'disco', 'dnb', 'documentary', 'downbeat', 'downtempo',
'drum', 'dub', 'dubstep', 'eastern', 'easy', 'electronic', 'electropop', 'emo',
'entehno', 'epicmetal', 'estrada', 'ethnic', 'eurofolk', 'european',
'experimental', 'extrememetal', 'fado', 'film', 'fitness', 'flamenco', 'folk',
'folklore', 'folkmetal', 'folkrock', 'folktronica', 'forró', 'frankreich',
```

```
'französisch', 'french', 'funk', 'future', 'gangsta', 'garage', 'german',
'ghazal', 'gitarre', 'glitch', 'gospel', 'gothic', 'grime', 'grunge', 'gypsy',
'handsup', "hard'n'heavy", 'hardcore', 'hardstyle', 'hardtechno', 'hiphop',
'historisch', 'holiday', 'horror', 'house', 'idm', 'independent', 'indian',
'indie', 'indipop', 'industrial', 'inspirational', 'instrumental',
'international', 'irish', 'jam', 'japanese', 'jazz', 'jewish', 'jpop', 'jungle',
'k-pop', 'karadeniz', 'karaoke', 'kayokyoku', 'korean', 'laiko', 'latin',
'latino', 'leftfield', 'local', 'lounge', 'loungeelectronic', 'lovers',
'malaysian', 'mandopop', 'marschmusik', 'meditative', 'mediterranean',
'melodic', 'metal', 'metalcore', 'mexican', 'middle', 'minimal',
'miscellaneous', 'modern', 'mood', 'mpb', 'muslim', 'native', 'neoklassik',
'neue', 'new', 'newage', 'newwave', 'nu', 'nujazz', 'numetal', 'oceania', 'old',
'opera', 'orchestral', 'other', 'piano', 'pop', 'popelectronic', 'popeurodance',
'post', 'posthardcore', 'postrock', 'power', 'progmetal', 'progressive',
'psychedelic', 'punjabi', 'punk', 'quebecois', 'ragga', 'ram', 'rancheras',
'rap', 'rave', 'reggae', 'reggaeton', 'regional', 'relax', 'religious', 'retro',
'rhythm', 'rnb', 'rnr', 'rock', 'rockabilly', 'romance', 'roots', 'ruspop',
'rusrap', 'rusrock', 'salsa', 'samba', 'schlager', 'self', 'sertanejo',
'shoegazing', 'showtunes', 'singer', 'ska', 'slow', 'smooth', 'soul', 'soulful',
'sound', 'soundtrack', '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',
'unknown', 'urban', 'uzbek', 'variété', 'vi', 'videogame', 'vocal', 'western',
'world', 'worldbeat', 'ïîï']
```

Voltar ao Índice

2.3.4 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.` *Pouco gênero como "hip hop" (mostrou um "hiphop" junto). Pelo contrário, mostrou outros tipos de gênero de músicas.*

Voltar ao Índice

2.4 Etapa 3. Teste da hipótese

2.4.1 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.

[20]:
```
# Contando as músicas tocadas em cada cidade

df_selected_days = df[df['day'].isin(['Monday', 'Wednesday', 'Friday'])].copy()
activity_per_city = df_selected_days.groupby(by='city')['genre'].count()
print("Número total de músicas tocadas por cidade (segunda, quarta e sexta):")
print(activity_per_city)
```

```
Número total de músicas tocadas por cidade (segunda, quarta e sexta):
city
Shelbyville    18512
Springfield    42741
Name: genre, dtype: int64
```

`Comente sobre suas observações aqui` - * Avaliação total de músicas tocadas por cidade, indica que a cidade de Springfield, os usuários tem o costume de ouvir um número maior de músicas durante a segunda, quarta e sexta.*

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.

[21]:
```
# Calculando as músicas escutadas em cada um desses três dias

monday_activity = df_selected_days[df_selected_days['day'] == 'Monday'].
 ↪groupby(by='city')['genre'].count()
print("\nNúmero de músicas tocadas por cidade na segunda-feira:")
print(monday_activity)

wednesday_activity = df_selected_days[df_selected_days['day'] == 'Wednesday'].
 ↪groupby(by='city')['genre'].count()
print("\nNúmero de músicas tocadas por cidade na quarta-feira:")
print(wednesday_activity)

friday_activity = df_selected_days[df_selected_days['day'] == 'Friday'].
 ↪groupby(by='city')['genre'].count()
print("\nNúmero de músicas tocadas por cidade na sexta-feira:")
print(friday_activity)
```

```
Número de músicas tocadas por cidade na segunda-feira:
city
Shelbyville     5614
Springfield    15740
Name: genre, dtype: int64

Número de músicas tocadas por cidade na quarta-feira:
city
Shelbyville     7003
Springfield    11056
Name: genre, dtype: int64

Número de músicas tocadas por cidade na sexta-feira:
city
Shelbyville     5895
Springfield    15945
Name: genre, dtype: int64
```

'Comente sobre suas observações aqui' - *Novamente a cidade de Springfield tocam músicas em quantidades maiores que a cidade de Shelbyville. Comparado as semanas na cidade de Springfield, de segunda e sexta são os dias que os usários mais tocam músicas. Já na cidade de Shelbyville a semana que os usuários mais tocam as músicas é de quarta-feira.*

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.

[22]:
```python
def number_tracks(day, city):
    day_filtered = df[df['day'] == day]
    city_filtered = day_filtered[day_filtered['city'] == city]
    track_count = city_filtered['user_id'].count()
    print(f"Número de músicas tocadas em {city} na {day}: {track_count}")
    return track_count
```

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.

[ ]:

[23]:
```
# a quantidade de músicas tocadas em Springfield na segunda-feira
number_tracks('Monday', 'Springfield')
```

```
Número de músicas tocadas em Springfield na Monday: 15740
```

[23]: 15740

[24]:
```
# a quantidade de músicas tocadas em Shelbyville na segunda-feira
number_tracks('Monday', 'Shelbyville')
```

```
Número de músicas tocadas em Shelbyville na Monday: 5614
```

[24]: 5614

[25]:
```
# a quantidade de músicas tocadas em Springfield na quarta-feira
number_tracks('Wednesday', 'Springfield')
```

```
Número de músicas tocadas em Springfield na Wednesday: 11056
```

[25]: 11056

[26]:
```
# a quantidade de músicas tocadas em Shelbyville na quarta-feira
number_tracks('Wednesday', 'Shelbyville')
```

```
Número de músicas tocadas em Shelbyville na Wednesday: 7003
```

[26]: 7003

[27]:
```
# a quantidade de músicas tocadas em Springfield na sexta-feira
number_tracks('Friday', 'Springfield')
```

```
Número de músicas tocadas em Springfield na Friday: 15945
```

[27]: 15945

[28]:
```
# a quantidade de músicas tocadas em Shelbyville na sexta-feira
number_tracks('Friday', 'Shelbyville')
```

```
Número de músicas tocadas em Shelbyville na Friday: 5895
```

[28]: 5895

Conclusões

`Comente sobre se a terceira hipótese está correta ou deve ser rejeitada. Explique seu raciocínio.-` *A parte de chamar a variável number_tracks seis vezes é desnecessária, porque as atividades anteriores já responde o total de músicas de cada cidade e responde os dias e números que foram tocadas as músicas.* A questão seria saber o por que dos usuários da cidade de Shebyville ter essa grande diferença de músicas tocadas?! Número de habitantes? Idade dos usuários? Qual o gênero de uma cidade pra outra?!*

Voltar ao Índice

3 Conclusões

`Resuma suas conclusões sobre a hipótese aqui` *Não podemos concluir de fato os dados dessa hipótese porque faltam dados importantes para ter uma afirmação correta. A única coisa que podemos afirmar, é que a cidade de Springfield - os usuários tocam em quantidades maiores (mais que o dobro de vezes) em comparação com a os usuários da cidade de Shelbyville. Não podendo concluir o porque existe essa diferença de dados?! O ideal seria destrinchar

3.0.1 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