Nome: Luiz Alberto Silva Mota

RA:10436776

Curso: Ciência da Computação

Disciplina: Algoritmos e Programação (Prática)

Orientador: Professor Dr. Pedro Henrique Cacique Braga

Universidade Presbiteriana Mackenzie

ARQUIVO

Primeiro, foi feito um import do arquivo csv em excel, chamado “billboard”, entrando diretamente na manipulação do arquivo no python. Depois, uma função para puxar os dados do arquivo csv.

```python
import csv

#-------------------FUNÇÕES--------------------------
#BUSCA OS DADOS DE UM ARQUIVO CSV. PARÂMETRO: NOME DO ARQUIVO
def getData(filename):
    file = open(filename, "r", encoding="cp437")
    next(file)
    data = list(csv.reader(file, delimiter=","))
    file.close()
    return data
```

O projeto começa a partir de um dicionário feito para o acesso devido a cada opção que o usuário deseja escolher.

```python
#----------------VARIÁVEIS----------------------------
billboard = getData("billboard.csv")
options = {"1":"Top 10 de um mês/ano específico",
           "2":"Top 12 do ano (1º de cada mês)",
           "3":"Top 100 de um mês/ano específico",
           "4":"Busca por artista",
           "5":"Busca por posição na lista no mês/ano",
           "6":"Artista que ficou mais semanas no mês/ano",
           "7":"Artista que mais apareceu na lista",
           "8":"Sair"}
```

FUNÇÕES DO MENU:

```
def rank_top100(year, month):
    top100 = []
    songs_added = []  # lista para armazenar as músicas já adicionadas
    for row in billboard:
        dates = row[0].split("-")
        if year in dates[0] and month in dates[1]:
            song_name = row[2]  # Nome da música
            if song_name not in songs_added:  # Verifica se a música já foi adicionada
                top100.append(row)
                songs_added.append(song_name)  # Adiciona o nome da música ao conjunto de
músicas adicionadas
    # Ordena a lista top_10 com base no valor máximo da coluna 6 (weeks-on-board)
    return sorted(top100, key=lambda x: int(x[6]), reverse=True)
```

rank_top100: Filtra as músicas do arquivo CSV para um ano e mês específicos, evitando duplicatas. Ordena as músicas com base na coluna de "weeks-on-board" em ordem decrescente. O método sort nesse caso altera a própria lista, enquanto o método key especifica uma certa função que extrai o valor da lista para ser usado como uma chave de ordenação simples, enquanto o lambda toma um certo elemento nomeado como “X” e retorna int(x[1]).

```
def top_10_musics(year, month):
    top_10 = rank_top100(year, month)
    for i, song in enumerate(top_10[:10], start=1):
        print(f"{i}º - {song[2]} - {song[3]}")
```

top_10_musics: Chama a função rank_top100 e imprime as top 10 músicas de um mês e ano específicos, já exibidas inumeradas pela função enumerate().

```
def EachMonth(year):
    for month in range(1, 13):
        top_song = rank_top100(year, month=str(month))[0] #pega a primeira linha do top
100, que seria o top1
        print(f"{month}. {top_song[2]} - {top_song[3]}")
```

EachMonth: Itera sobre os meses de um ano específico e imprime a música top 1 de cada mês.

```python
def top_100_musics(year, month):
    top_100 = rank_top100(year, month)
    for i, song in enumerate(top_100[:100], start=1):
        print(f"{i}º - {song[2]} - {song[3]}")
```

top_100_musics: Similar à função top_10_musics, mas imprime as top 100 músicas de um mês e ano específicos.

```python
def artist(name):
    artists = []
    artist_counter = 0
    for i in billboard:
        if name.upper() in i[3].upper():
            print(f"{i[0]} - Rank: {i[1]} - Música: {i[2]}")
            artist_counter+=1
            print()

    if artist_counter == '1':
        print(f"Este artista foi citado {artist_counter} vez.")
    else:
        print(f"Este artista foi citado {artist_counter} vezes.")
    print()
```

artist: Busca todas as instâncias de um artista especificado pelo usuário no dataset e utiliza um contador que somará mais 1 unidade cada vez que o sistema encontrar o artista, assim exibindo no final quantas vezes ele apareceu.

- Name.upper() e i[3].upper converte as strings para letra maiúscula, para que não haja erro na verificação em relação ao arquivo csv.

```python
def PositionSearch():
    year = input("Digite o ano solicitado (1955-2021): ")
    if year == '2021':
        month = input("Digite o mês solicitado (1-11): ")
    else:
        month = input("Digite o mês solicitado (1-12): ")
    position = int(input("Digite o número da posição que você deseja (1-100): "))
    top_100 = rank_top100(year, month)
    for i, song in enumerate(top_100[:100], start=1):
        if i == position:
            print(f"\n{i}. {song[2]} - {song[3]}")
```

PositionSearch: Permite ao usuário buscar uma posição específica (1-100) no ranking de um mês e ano específicos.

- If year == ‘2021’, pois no arquivo “Billboard”, só foi feito o rank até o mês 11 de 2021, sendo impossível puxar dados do mês 12, caso o usuário quisesse.

```
def top_music(year, month):
    top100artist = []
    artist_added = []  # Conjunto para armazenar os artistas já adicionados
    for row in billboard:
        dates = row[0].split("-")
        if year in dates[0] and month in dates[1]:
            artist_name = row[3]  # Nome do artista
            if artist_name not in artist_added:  # Verifica se o artista já foi adicionado
                top100artist.append(row)
                artist_added.append(artist_name)  # Adiciona o nome do artista ao conjunto de artistas adicionados
        # Ordena a lista top_10 com base no valor máximo da coluna 6 (weeks-on-board)
    art = sorted(top100artist, key=lambda x: int(x[6]), reverse=True)
    for i, artist in enumerate(art[:1], start=1):
       print(f"O artista mais famoso do mês {month} foi {artist[3]} com {artist[6]} semanas seguidas no top 100.")
```

top_music: Utiliza quase o mesmo esquema que a função rank_top100, a diferença é que esta função encontra o artista que ficou mais semanas no top 100 em um mês e ano específicos.

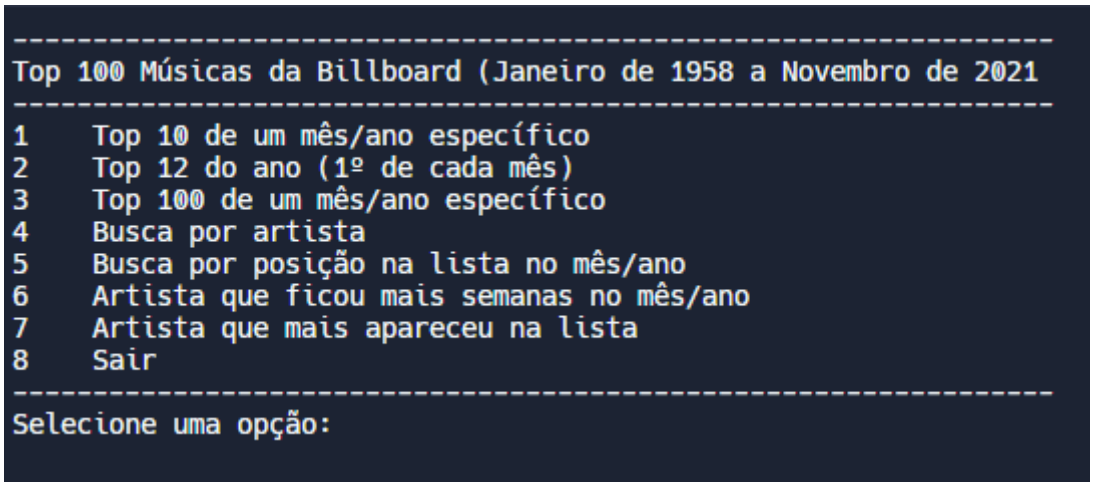
```
def famous_music():
    artist_counter  = {}
    for row in billboard:
        artist = row[3]
        if artist in artist_counter:
            artist_counter[artist] += 1
        else:
            artist_counter[artist] = 1
    max_artist = max(artist_counter, key=lambda x: artist_counter[x])
    times = artist_counter[max_artist]
    print(f"Artista: {max_artist}")
    print(f"Quantidade de vezes que apareceu no top 100: {times}")
```

famous_music: Encontra e imprime o nome do artista que mais apareceu no top 100 ao longo do período coberto pelo dataset. Utiliza-se a variável times para puxar, da chave do dicionário, o número de vezes que aquele artista apareceu no top 100.

- A função max() faz com que ele ordene o dicionário de forma decrescente com base no maior valor da variável artist_counter.

executeMenu: Recebe a opção selecionada pelo usuário e chama a função correspondente para realizar a análise ou busca solicitada.

FUNCIONAMENTO DO CÓDIGO



Aqui está o menu disponível para o usuário.

Opção 1 – Insira o ano e o mês específico, retornando, assim, o top 10 daquele período de tempo.

Opção 2 – Insira o ano desejado, retornando o top 1 de cada mês daquele ano específico.

Opção 3 – Insira o ano e o mês específico, retornando, assim, o top 100 daquele período de tempo.

Opção 4 – Insira o nome do artista (não se importe com as letras maiúsculas e minúsculas do nome) e o código devolverá todas as aparições daquele artista apresentada no arquivo “Billboard”.

Opção 5 – Insira o ano, mês e posição específica, retornando a música/artista presente nos dados desejados.

Opção 6 – Insira o ano e o mês específico, retornando, assim, o artista que ficou mais semanas presente no top 100 músicas da Billboard naquele período.

Opção 7 – Não há necessidade de inserir nada nesta opção. Apenas digite “7” no “Selecione uma opção”, e retornará o artista mais presente de todo este arquivo csv.

Opção 8 – O programa é instantaneamente desligado.