```
import random
import string
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import numpy as np

import random
import string
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Extract existing playlists from the given data
import random
import string

songids = ['SONGID' + str(i).zfill(2) for i in range(1, 21)]
userid = [''.join('USER') + str(random.randint(10, 99)).zfill(2) for i in range(100)]
playlist = {i: j for i, j in zip(userid, [random.sample(songids, random.randint(4, 7)) for i in range(len(userid))])}

print(playlist)
```

```
{'USER24': ['SONGID18', 'SONGID02', 'SONGID10', 'SONGID01'], 'USER72': ['SONGID11', 'SONGID07', 'SONGID09', 'SONGID03', 'SONGID19', 'SON
```

Let's say the new user has listened to these 3 songs

```
myplist=['SONGID01','SONGID06','SONGID16']
myplist
```

```
['SONGID01', 'SONGID06', 'SONGID16']
```

```
# Assign empty list to totalplist
totalplist=[]

# Number of songs in new user "myplist"
for i in range(0,len(myplist)):
    songs=myplist[i]

# Find playlists for each songs in myplist
    findplist = ([key for key, value in playlist.items() if songs in value])

# Extend the totalplist with each songs finding.
    totalplist.extend(findplist)
totalplist
```

```
 'USER51',
 'USER80',
 'USER84',
 'USER57',
 'USER70',
 'USER91',
 'USER28',
 'USER31',
 'USER39',
 'USER65',
 'USER71',
 'USER50',
 'USER87',
 'USER25',
 'USER56',
 'USER75',
 'USER11',
 'USER43',
 'USER86',
 'USER93',
 'USER49',
 'USER78',
 'USER96',
 'USER26',
```

```
        'USER34',
        'USER82',
        'USER70',
        'USER31',
        'USER53',
        'USER38',
        'USER94',
        'USER59',
        'USER15',
        'USER87',
        'USER19',
        'USER22',
        'USER41',
        'USER49',
        'USER36',
        'USER26',
        'USER46',
        'USER30',
        'USER92',
        'USER34',
        'USER84',
        'USER70',
        'USER21',
        'USER29',
        'USER55',
        'USER19',
        'USER25',
        'USER20',
        'USER43',
        'USER22',
        'USER41',
        'USER98',
        'USER83']
```

```
# Number of playlist for 3 songs['SONGID01','SONGID06','SONGID16'] is
len(totalplist)
```

```
59
```

From this playlist(s), pick the remaining songs and recommend to the new user (refer below examples for more details)

```
# Assign empty list
songs=[]

# Loop for ittrate songs from all selected playlist
for i in range(0,len(totalplist)):
    nlist = totalplist[i]
    flist=([value for key, value in playlist.items() if nlist in key])
    songs.extend(flist)
songs
```

```
[['SONGID18', 'SONGID02', 'SONGID10', 'SONGID01'],
 ['SONGID09',
  'SONGID01',
  'SONGID12',
  'SONGID13',
  'SONGID14',
  'SONGID05',
  'SONGID02'],
 ['SONGID01', 'SONGID09', 'SONGID20', 'SONGID05', 'SONGID03', 'SONGID07'],
 ['SONGID10', 'SONGID02', 'SONGID01', 'SONGID09', 'SONGID16'],
 ['SONGID20', 'SONGID01', 'SONGID14', 'SONGID11', 'SONGID13', 'SONGID07'],
 ['SONGID06',
  'SONGID16',
  'SONGID03',
  'SONGID01',
  'SONGID18',
  'SONGID13',
  'SONGID19'],
 ['SONGID14', 'SONGID01', 'SONGID03', 'SONGID04', 'SONGID05'],
 ['SONGID01', 'SONGID02', 'SONGID11', 'SONGID15', 'SONGID13'],
 ['SONGID07',
  'SONGID04',
  'SONGID06',
  'SONGID17',
  'SONGID15',
  'SONGID13',
  'SONGID01'],
 ['SONGID01',
  'SONGID04',
  'SONGID11',
```

```
          'SONGID05',
          'SONGID17',
          'SONGID15',
          'SONGID19'],
         ['SONGID10',
          'SONGID13',
          'SONGID01',
          'SONGID20',
          'SONGID07',
          'SONGID09',
          'SONGID04'],
         ['SONGID04',
          'SONGID15',
          'SONGID01',
          'SONGID18',
          'SONGID07',
          'SONGID13',
          'SONGID20'],
         ['SONGID01', 'SONGID12', 'SONGID14', 'SONGID15', 'SONGID20', 'SONGID02'],
         ['SONGID01', 'SONGID09', 'SONGID06', 'SONGID04', 'SONGID19'],
         ['SONGID01',
          'SONGID05',
          'SONGID12',
          'SONGID17',
          'SONGID19',
          'SONGID16',
          'SONGID13'],
         ['SONGID13', 'SONGID11', 'SONGID01', 'SONGID04', 'SONGID18']]
```

```
# processing all songs into a single list
from functools import reduce
song_list = reduce(lambda xs,ys: xs + ys, songs)
song_list
```

```
['SONGID18',
 'SONGID02',
 'SONGID10',
 'SONGID01',
 'SONGID09',
 'SONGID01',
 'SONGID12',
 'SONGID13',
 'SONGID14',
 'SONGID05',
 'SONGID02',
 'SONGID01',
 'SONGID09',
 'SONGID20',
 'SONGID05',
 'SONGID03',
 'SONGID07',
 'SONGID10',
 'SONGID02',
 'SONGID01',
 'SONGID09',
 'SONGID16',
 'SONGID20',
 'SONGID01',
 'SONGID14',
 'SONGID11',
 'SONGID13',
 'SONGID07',
 'SONGID06',
 'SONGID16',
 'SONGID03',
 'SONGID01',
 'SONGID18',
 'SONGID13',
 'SONGID19',
 'SONGID14',
 'SONGID01',
 'SONGID03',
 'SONGID04',
 'SONGID05',
 'SONGID01',
 'SONGID02',
 'SONGID11',
 'SONGID15',
 'SONGID13',
 'SONGID07',
 'SONGID04',
 'SONGID06',
 'SONGID17',
 'SONGID15',
```

```
        'SONGID13',
        'SONGID01',
        'SONGID01',
        'SONGID04',
        'SONGID11',
        'SONGID05',
        'SONGID17',
        'SONGID15',
```

```
# Removing duplicates in the list
uniq_songs = []
for i in song_list:
    if i not in uniq_songs:
        uniq_songs.append(i)

uniq_songs
```

```
     ['SONGID18',
      'SONGID02',
      'SONGID10',
      'SONGID01',
      'SONGID09',
      'SONGID12',
      'SONGID13',
      'SONGID14',
      'SONGID05',
      'SONGID20',
      'SONGID03',
      'SONGID07',
      'SONGID16',
      'SONGID11',
      'SONGID06',
      'SONGID19',
      'SONGID04',
      'SONGID15',
      'SONGID17',
      'SONGID08']
```

```
# Removing new user myplist songs list from recommend_song list
recom_songs = [songs for songs in uniq_songs if songs not in myplist]
recom_songs
```

```
     ['SONGID18',
      'SONGID02',
      'SONGID10',
      'SONGID09',
      'SONGID12',
      'SONGID13',
      'SONGID14',
      'SONGID05',
      'SONGID20',
      'SONGID03',
      'SONGID07',
      'SONGID11',
      'SONGID19',
      'SONGID04',
      'SONGID15',
      'SONGID17',
      'SONGID08']
```

How can we identify playlists that contain all songs in a new user's input list?

```
# Given input list of songs for a new user
new_user_songs = ['SONGID03', 'SONGID07', 'SONGID12']

# Identify playlists containing all songs from the new user's input list
matching_playlists = [user for user, songs in playlist.items() if all(song in songs for song in new_user_songs)]

print("Matching Playlists:", matching_playlists)
```

```
     Matching Playlists: ['USER72']
```

How can we recommend songs from the matching playlists to the new user?

```
# Extract songs from the matching playlists
recommended_songs = []
```

```
for user in matching_playlists:
    recommended_songs.extend([song for song in playlist[user] if song not in new_user_songs])

print("Recommended Songs:", recommended_songs)
```

```
Recommended Songs: ['SONGID11', 'SONGID09', 'SONGID19']
```

```
# Ensure recommended songs are unique
unique_recommended_songs = list(set(recommended_songs))

print("Unique Recommended Songs:", unique_recommended_songs)
```

```
Unique Recommended Songs: ['SONGID09', 'SONGID19', 'SONGID11']
```

How can we limit the number of recommendations to a certain maximum count?

```
# Limit the number of recommendations to a maximum count, e.g., 10 songs
max_recommendations = 10
limited_recommended_songs = unique_recommended_songs[:max_recommendations]

print("Limited Recommended Songs:", limited_recommended_songs)
```

```
Limited Recommended Songs: ['SONGID09', 'SONGID19', 'SONGID11']
```

How can we encapsulate the recommendation process into a function?

```
# Create a function to generate recommendations for a new user
def generate_recommendations(new_user_songs, max_count=10):
    matching_playlists = [user for user, songs in playlist.items() if all(song in songs for song in new_user_songs)]
    recommended_songs = []

    for user in matching_playlists:
        recommended_songs.extend([song for song in playlist[user] if song not in new_user_songs])

    unique_recommended_songs = list(set(recommended_songs))
    limited_recommended_songs = unique_recommended_songs[:max_count]

    return limited_recommended_songs

# Example usage:
new_user_songs = ['SONGID03', 'SONGID07', 'SONGID12']
recommended_songs = generate_recommendations(new_user_songs)
print("Recommended Songs:", recommended_songs)
```

```
Recommended Songs: ['SONGID09', 'SONGID19', 'SONGID11']
```

How can we ensure that the code is modular and well-documented?

```
# Modular code with comments

# Function to identify matching playlists
def find_matching_playlists(new_user_songs):
    matching_playlists = [user for user, songs in playlist.items() if all(song in songs for song in new_user_songs)]
    return matching_playlists

# Function to generate recommendations
def generate_recommendations(new_user_songs, max_count=10):
    matching_playlists = find_matching_playlists(new_user_songs)
    recommended_songs = []

    for user in matching_playlists:
        recommended_songs.extend([song for song in playlist[user] if song not in new_user_songs])

    unique_recommended_songs = list(set(recommended_songs))
    limited_recommended_songs = unique_recommended_songs[:max_count]

    return limited_recommended_songs

# Example usage:
```

```
# Example usage:
new_user_songs = ['SONGID03', 'SONGID07', 'SONGID12']
recommended_songs = generate_recommendations(new_user_songs)
print("Recommended Songs:", recommended_songs)
```

```
Recommended Songs: ['SONGID09', 'SONGID19', 'SONGID11']
```