Project - 01 - Codus of Delhi

(*) Read data :

```
import json

# writing a function to load the data -

def load_data(filname):
        with open(filname, "r") as f:
            data = json.load(f)

        return data

data = load_data("data.json")
print(data)
print(type(data))
```

write a function to display user and their connection.

```
def display_users(data):
    print("users and their connections")
    for users in data["users"]:
        print(f"ID: {users['id']} - {users['name']} is friends
            with: {users['friends']} and liked pages are {users['liked pages']}")

    print("\n page information")
    for pages in data['pages']:
        print(f"{pages['id']}: {page['name']}")

display_user(data).
```

Writting a function to load the data -

```
def load_data (filename):
    with open (filename, "r") as f:
        data = json.load (f)
    return data

data = load_data ('data2.json')
print (data)
print ('\n')
print (type (data))
```

(*) Cleaning and Structure the data -

```
import json
def clean_data(data):
    # remove users with missing name
    data["users"] = [user for user in data["users"] if user["name"].strip()]
    #remove duplicate friends
    for user in data["users"]:
        user["friends"] = list(set(user["friends"]))

    # Remove inactive users
    data["users"] = [user for user in data["users"] if user["friend"]or user["liked_pages"]]

    # Remove duplicate pages
    unique_pages = {}
    for page in data["pages"]:
        unique_pages[page["id"]] = page
    data["pages"] = list(unique_page.values())

    return data

#load, clean, and display the cleaned data -
data = json.load(open("Codibook_data.j
data = clean_data(data)
json.dump(data, open("cleaned_codibook_data.json", "w"), indent = 4)
print("Data is cleaned...")
```

⊛ finding "People You May know" -

```
import json
def load_data(filname):
    with open(filnamu, "r") as fil:
        return json.load(fil)
def find_people_you_may_know(usur_id, data):
    usur_friends = {}
    for usur in data['usurs']:
        usur_friends[usur['id']] = set(usur['friends'])

    if usur_id not in usur_friends:
        return []

    direct_friends = usur_friends[usur_id]
    suggestions = {}

    for friends in direct_friends:
        # for all friends of friends
        for mutual in usur_friends[friend]:
            # if mutual id is not the samu usur and not already a
              direct friend of usur -
            if mutual ≠ usur_id and mutual not in direct_friends:
                # Count mutual friends -
                suggestions[mutual] = suggestions.get(mutual, 0) + 1
sorted_suggestions = sorted(suggestions.items(), key = lambda x: x[1], reversed =
                                                                        True)
return [usur_id for usur_id, _ in sorted_suggestions]
```

```
#Load data
data = load_data("cleaned_codibook-data.json")
user_id = 1 # Example: finding suggestions for Amit
recommendations = find_people-you_may_know(user_id, data)
print(f"People You may know for User {user_id}: {recomendations}")
```

⑧ "Pages You Might Like" -

```
import json.
# function to load json data from a file -
def load_data(filname):
    with open(filname, "r") as file:
        return json.load(file)
# function to find pages a user might like based on common interest -
def find_pages_you_might_like(user_id, data):
    # Dictonary to store user interaction with pages -
    user_page = {}
    for user in data["users"]:
        user_pages[user['id']] = set(user["liked_pages"])
    # if the user is not found, return an empty list -
    if user_id not in user_pages:
        return []

    user_liked_pages = user_pages[user_id]
    page_suggestions = {}

    for other_user, pages in user_pages.item():
        if other_user ≠ user.id:
            shared_pages = user_liked_pages.intersection(pages)
            for page not in user_liked_pages:
                page_suggestions[page] = page_suggestions.get(page, 0)
                                        + len(shared_pages)

# Sort recommended pages based on the no. of shared interactions -
sorted_pages = sorted(pages_suggestions.items(), key = lambda x: x[1],
                      reversed = True)
return [page_id for page_id, _ in sorted_pages]
```

Load data -

```
data = load_data("cleaned_codebook_data.json")
user_id = 1
page_recommendations = find_pages_you_might_like(user_id, data)
print(f"Pages you might like for user {user_id}: {page_recommendation}")
```