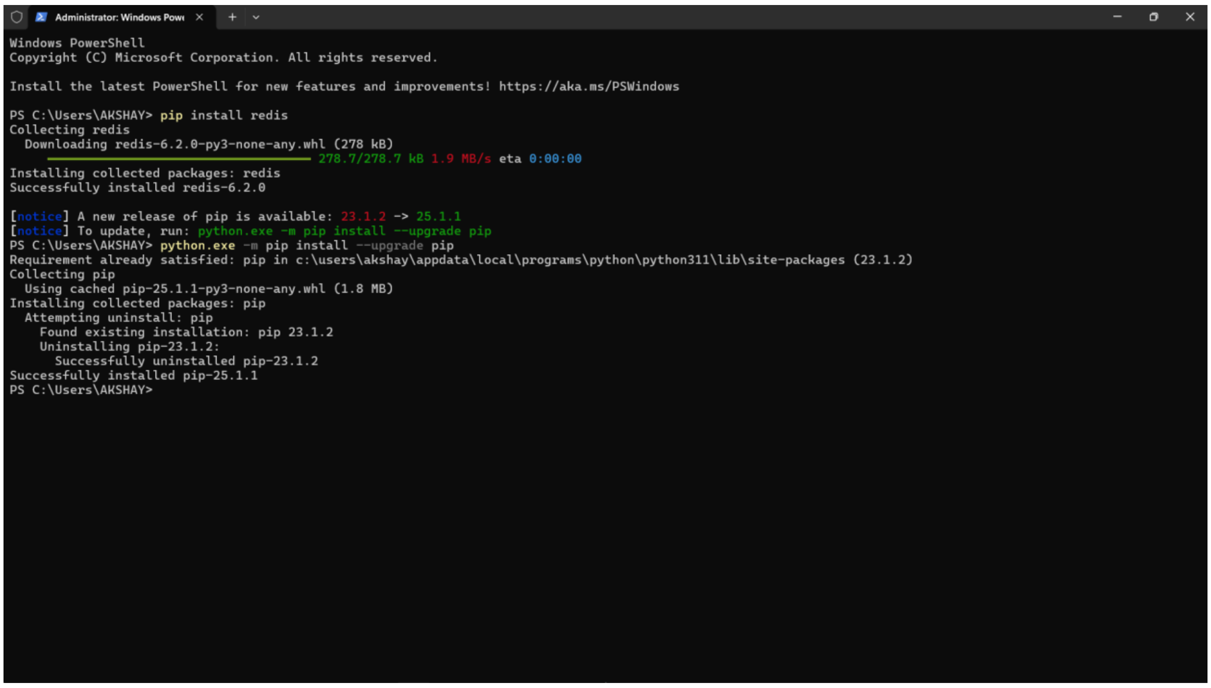
Big Data Management - Assignment 5

Redis Data Management

Akshay Kumar (G24AI1033)

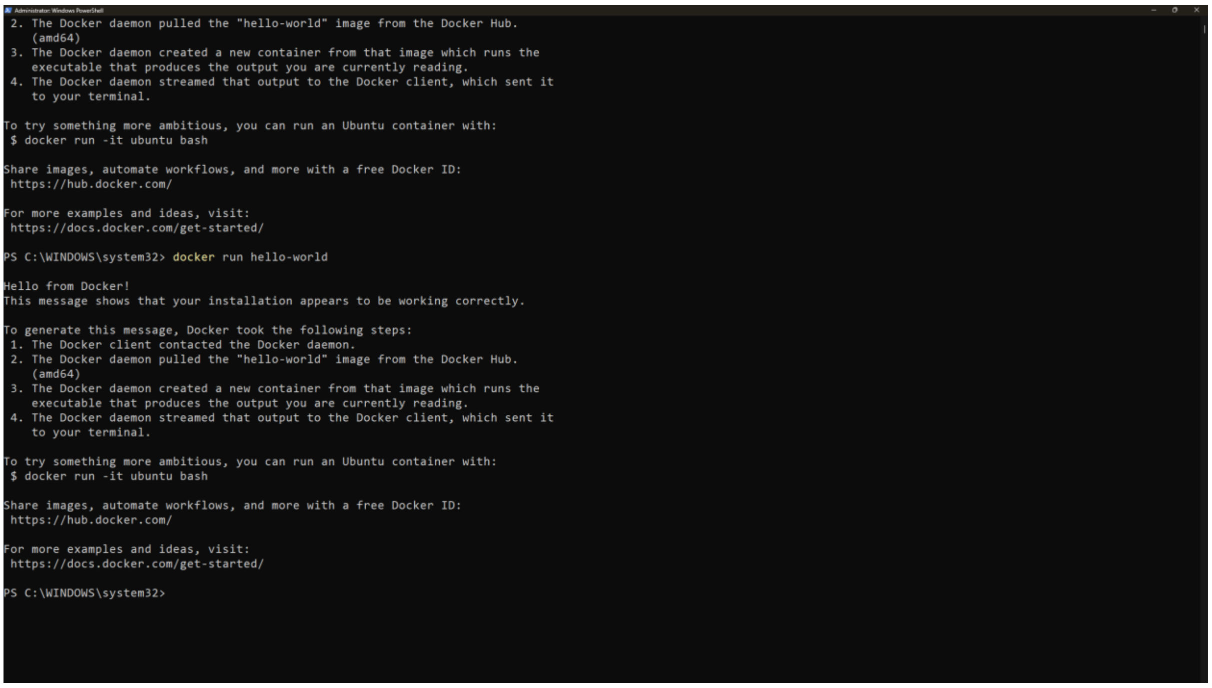
Github Link – Assignment5

1. Install the redis-py library

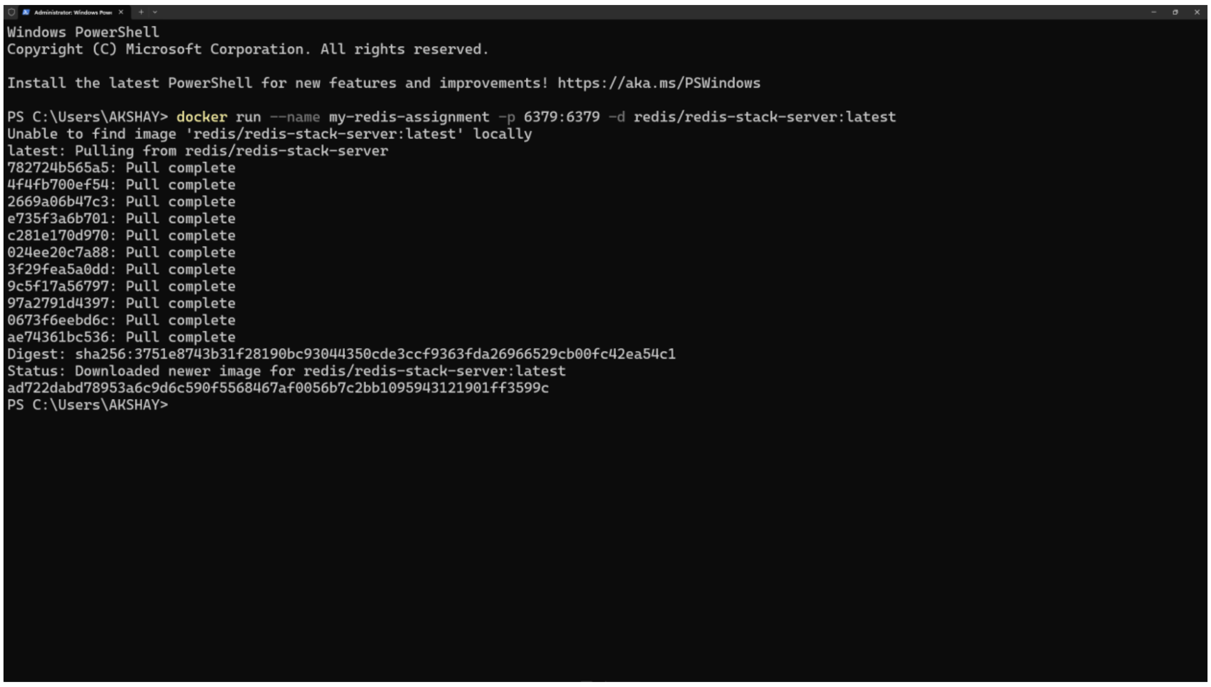


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\AKSHAY> pip install redis
Collecting redis
  Downloading redis-6.2.0-py3-none-any.whl (278 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 278.7/278.7 kB 1.9 MB/s eta 0:00:00
Installing collected packages: redis
Successfully installed redis-6.2.0

[notice] A new release of pip is available: 23.1.2 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\AKSHAY> python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\akshay\appdata\local\programs\python\python311\lib\site-packages (23.1.2)
Collecting pip
  Using cached pip-25.1.1-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.1.2
    Uninstalling pip-23.1.2:
      Successfully uninstalled pip-23.1.2
Successfully installed pip-25.1.1
PS C:\Users\AKSHAY>
```

2. Get your Redis Database Connection Details (Using Docker)

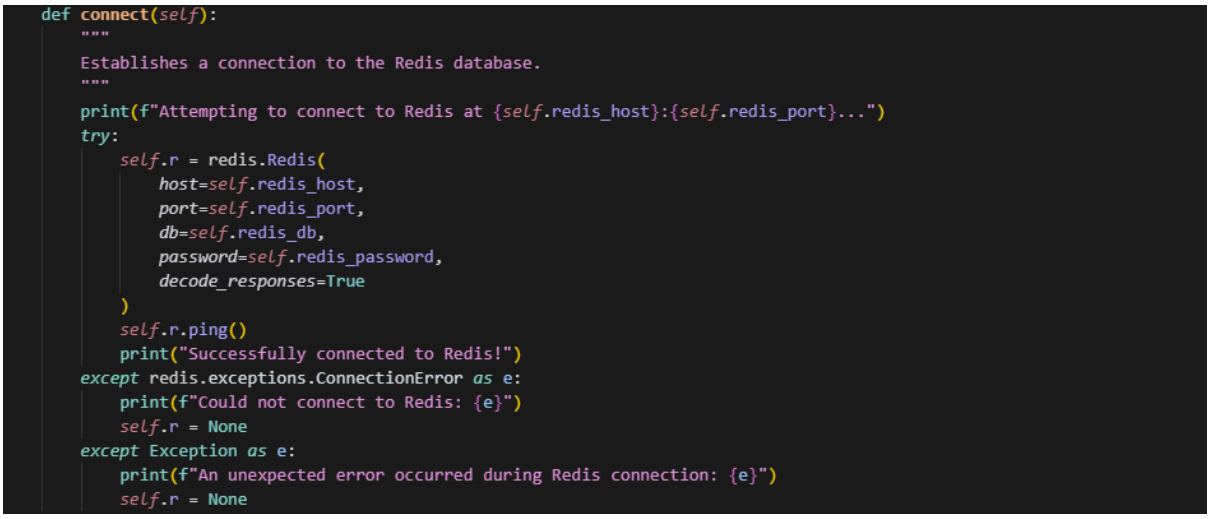


```
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

PS C:\WINDOWS\system32> docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

PS C:\WINDOWS\system32>
```

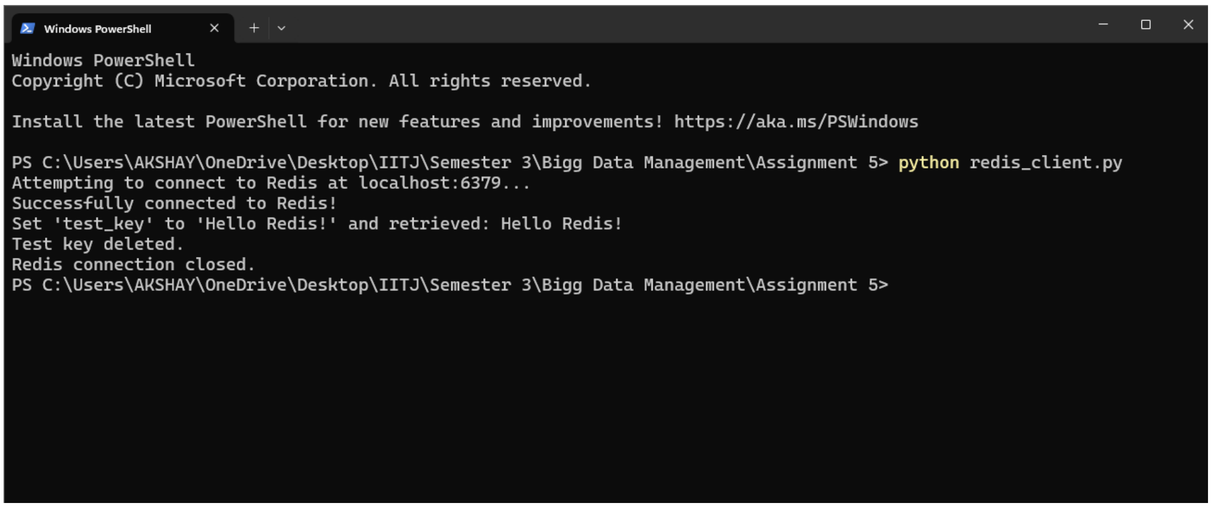


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\AKSHAY> docker run --name my-redis-assignment -p 6379:6379 -d redis/redis-stack-server:latest
Unable to find image 'redis/redis-stack-server:latest' locally
latest: Pulling from redis/redis-stack-server
782724b565a5: Pull complete
4f4fb700ef54: Pull complete
2669a06b47c3: Pull complete
e735f3a6b701: Pull complete
c281e170d970: Pull complete
024ee20c7a88: Pull complete
3f29fea5a0dd: Pull complete
9c5f17a56797: Pull complete
97a2791d4397: Pull complete
0673f6eebd6c: Pull complete
ae74361bc536: Pull complete
Digest: sha256:3751e8743b31f28190bc93044350cde3ccf9363fda26966529cb00fc42ea54c1
Status: Downloaded newer image for redis/redis-stack-server:latest
ad722dabd78953a6c9d6c590f5568467af0056b7c2bb1095943121901ff3599c
PS C:\Users\AKSHAY>
```

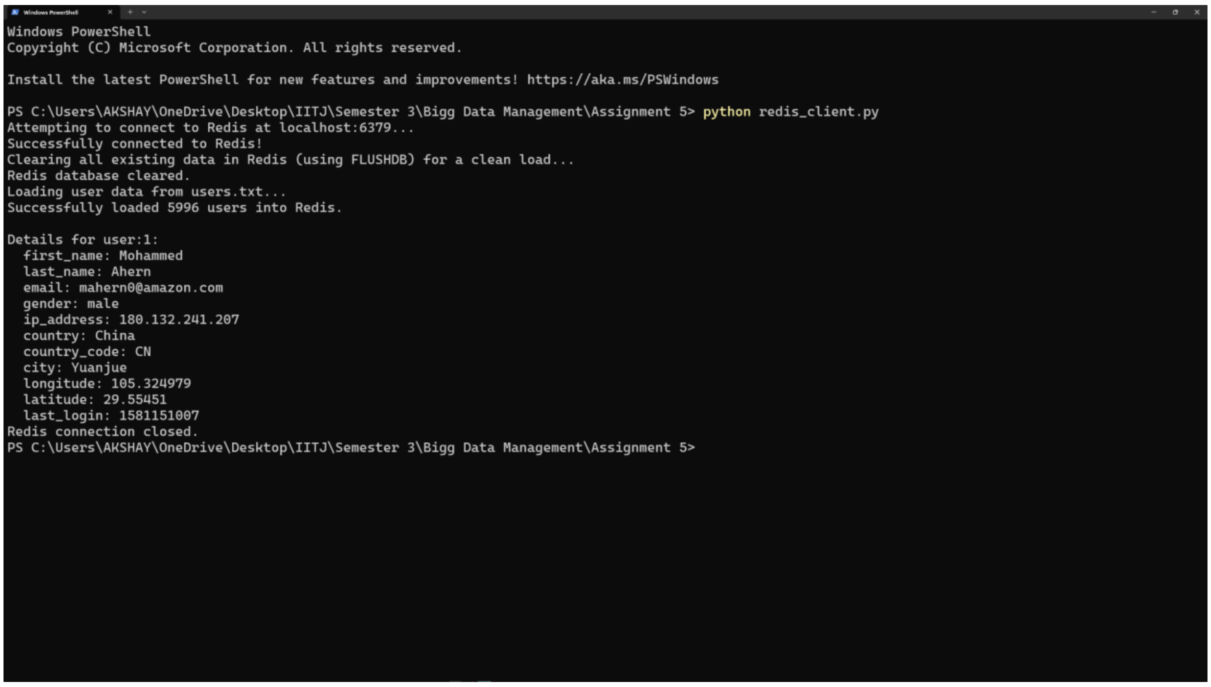
4. Adding the basic structure and connect() method



```python
    def connect(self):
        """
        Establishes a connection to the Redis database.
        """
        print(f"Attempting to connect to Redis at {self.redis_host}:{self.redis_port}...")
        try:
            self.r = redis.Redis(
                host=self.redis_host,
                port=self.redis_port,
                db=self.redis_db,
                password=self.redis_password,
                decode_responses=True
            )
            self.r.ping()
            print("Successfully connected to Redis!")
        except redis.exceptions.ConnectionError as e:
            print(f"Could not connect to Redis: {e}")
            self.r = None
        except Exception as e:
            print(f"An unexpected error occurred during Redis connection: {e}")
            self.r = None
```

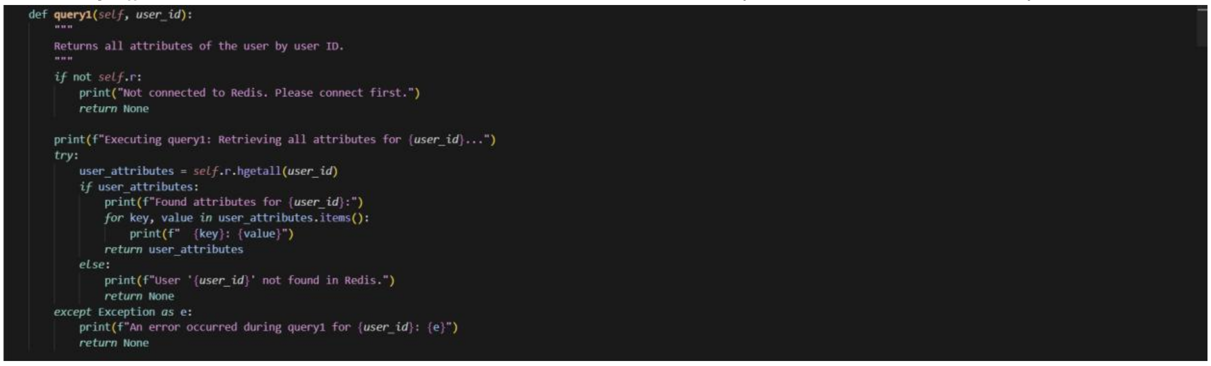


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\Bigg Data Management\Assignment 5> python redis_client.py
Attempting to connect to Redis at localhost:6379...
Successfully connected to Redis!
Set 'test_key' to 'Hello Redis!' and retrieved: Hello Redis!
Test key deleted.
Redis connection closed.
PS C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\Bigg Data Management\Assignment 5>
```

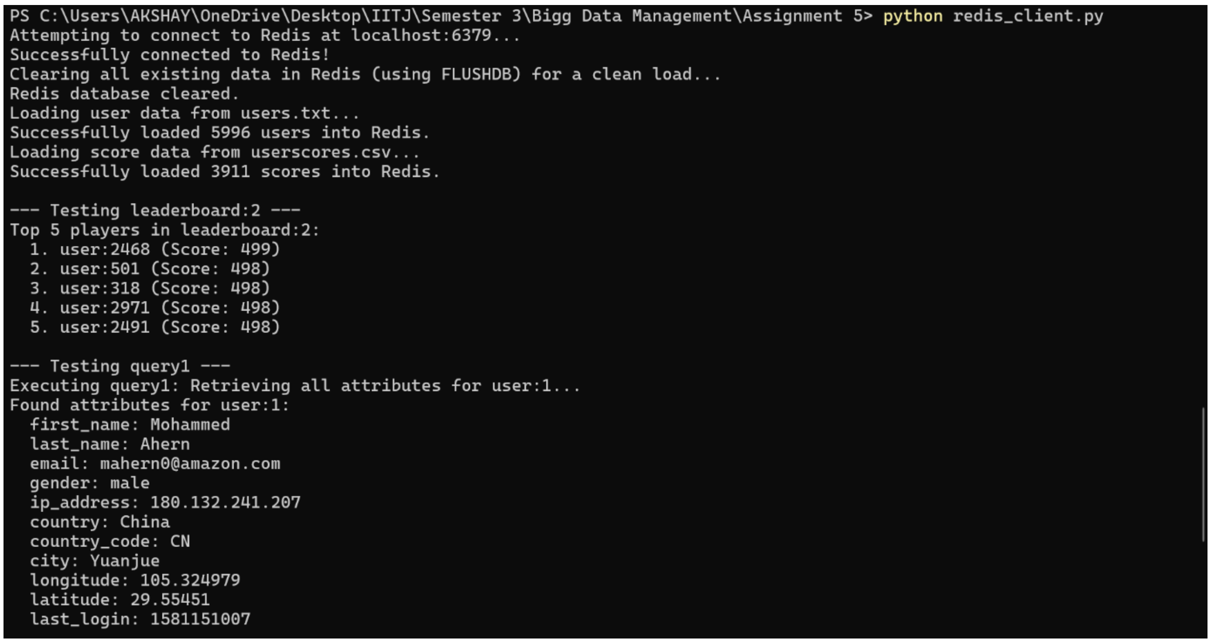


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\Bigg Data Management\Assignment 5> python redis_client.py
Attempting to connect to Redis at localhost:6379...
Successfully connected to Redis!
Clearing all existing data in Redis (using FLUSHDB) for a clean load...
Redis database cleared.
Loading user data from users.txt...
Successfully loaded 5996 users into Redis.

Details for user:1:
  first_name: Mohammed
  last_name: Ahern
  email: mahern0@amazon.com
  gender: male
  ip_address: 180.132.241.207
  country: China
  country_code: CN
  city: Yuanjue
  longitude: 105.324979
  latitude: 29.55451
  last_login: 1581151007
Redis connection closed.
PS C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\Bigg Data Management\Assignment 5>
```

6. Query1() : This method retrieves all stored attributes for a specific user, identified by their user ID.

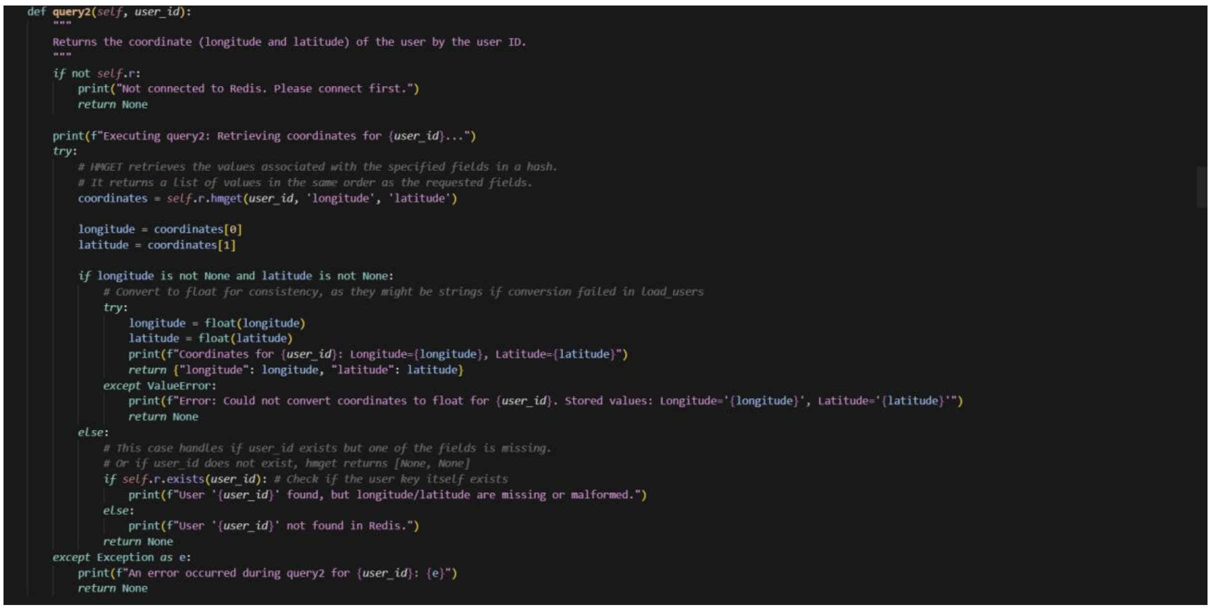


```python
    def query1(self, user_id):
        """
        Returns all attributes of the user by user ID.
        """
        if not self.r:
            print("Not connected to Redis. Please connect first.")
            return None

        print(f"Executing query1: Retrieving all attributes for {user_id}...")
        try:
            user_attributes = self.r.hgetall(user_id)
            if user_attributes:
                print(f"Found attributes for {user_id}:")
                for key, value in user_attributes.items():
                    print(f"  {key}: {value}")
                return user_attributes
            else:
                print(f"User '{user_id}' not found in Redis.")
                return None
        except Exception as e:
            print(f"An error occurred during query1 for {user_id}: {e}")
            return None
```

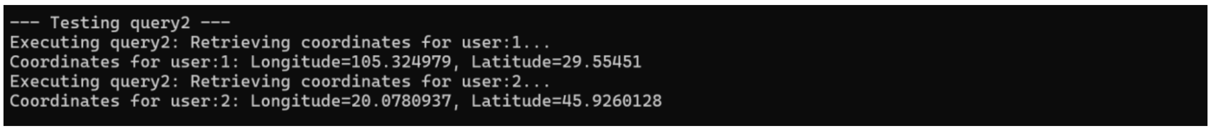


```
PS C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\Bigg Data Management\Assignment 5> python redis_client.py
Attempting to connect to Redis at localhost:6379...
Successfully connected to Redis!
Clearing all existing data in Redis (using FLUSHDB) for a clean load...
Redis database cleared.
Loading user data from users.txt...
Successfully loaded 5996 users into Redis.
Loading score data from userscores.csv...
Successfully loaded 3911 scores into Redis.

--- Testing leaderboard:2 ---
Top 5 players in leaderboard:2:
  1. user:2468 (Score: 499)
  2. user:501 (Score: 498)
  3. user:318 (Score: 498)
  4. user:2971 (Score: 498)
  5. user:2491 (Score: 498)

--- Testing query1 ---
Executing query1: Retrieving all attributes for user:1...
Found attributes for user:1:
  first_name: Mohammed
  last_name: Ahern
  email: mahern0@amazon.com
  gender: male
  ip_address: 180.132.241.207
  country: China
  country_code: CN
  city: Yuanjue
  longitude: 105.324979
  latitude: 29.55451
  last_login: 1581151007
```

7. Query2() : This method fetches the longitude and latitude coordinates for a specific user ID.

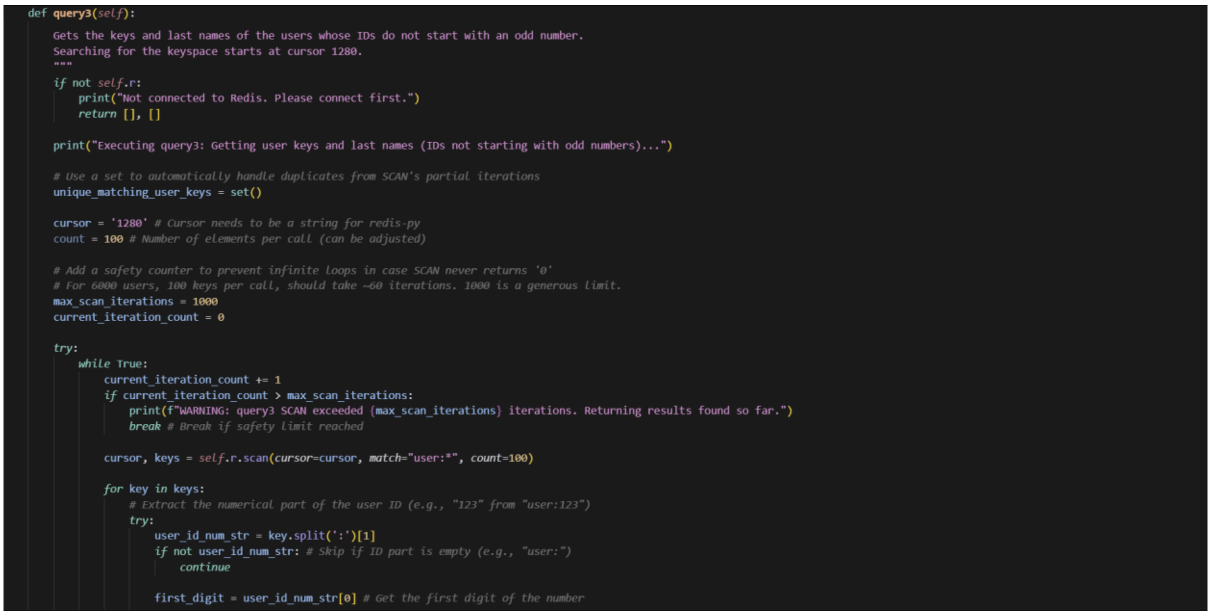


```python
def query2(self, user_id):
    """
    Returns the coordinate (longitude and latitude) of the user by the user ID.
    """
    if not self.r:
        print("Not connected to Redis. Please connect first.")
        return None

    print(f"Executing query2: Retrieving coordinates for {user_id}...")
    try:
        # HMGET retrieves the values associated with the specified fields in a hash.
        # It returns a list of values in the same order as the requested fields.
        coordinates = self.r.hmget(user_id, 'longitude', 'latitude')

        longitude = coordinates[0]
        latitude = coordinates[1]

        if longitude is not None and latitude is not None:
            # Convert to float for consistency, as they might be strings if conversion failed in load_users
            try:
                longitude = float(longitude)
                latitude = float(latitude)
                print(f"Coordinates for {user_id}: Longitude={longitude}, Latitude={latitude}")
                return {"longitude": longitude, "latitude": latitude}
            except ValueError:
                print(f"Error: Could not convert coordinates to float for {user_id}. Stored values: Longitude='{longitude}', Latitude='{latitude}'")
                return None
        else:
            # This case handles if user_id exists but one of the fields is missing.
            # Or if user_id does not exist, hmget returns [None, None]
            if self.r.exists(user_id): # Check if the user key itself exists
                print(f"User '{user_id}' found, but longitude/latitude are missing or malformed.")
            else:
                print(f"User '{user_id}' not found in Redis.")
            return None
    except Exception as e:
        print(f"An error occurred during query2 for {user_id}: {e}")
        return None
```

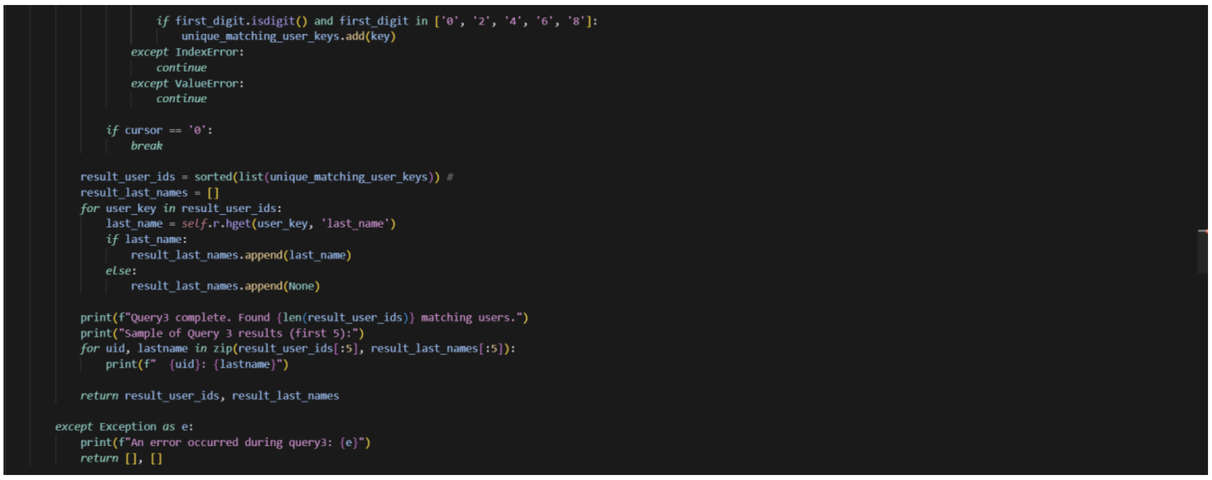


```
--- Testing query2 ---
Executing query2: Retrieving coordinates for user:1...
Coordinates for user:1: Longitude=105.324979, Latitude=29.55451
Executing query2: Retrieving coordinates for user:2...
Coordinates for user:2: Longitude=20.0780937, Latitude=45.9260128
```

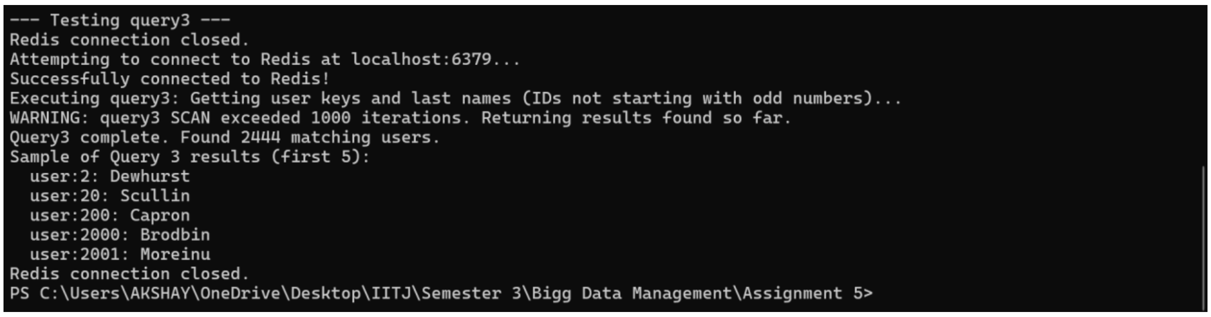
8. Query3() : This method scans the user keyspace, filters for users whose IDs do not start with an odd number, and returns their keys and last names.



```python
def query3(self):
    Gets the keys and last names of the users whose IDs do not start with an odd number.
    Searching for the keyspace starts at cursor 1280.
    """
    if not self.r:
        print("Not connected to Redis. Please connect first.")
        return [], []

    print("Executing query3: Getting user keys and last names (IDs not starting with odd numbers)...")

    # Use a set to automatically handle duplicates from SCAN's partial iterations
    unique_matching_user_keys = set()

    cursor = '1280' # Cursor needs to be a string for redis-py
    count = 100 # Number of elements per call (can be adjusted)

    # Add a safety counter to prevent infinite loops in case SCAN never returns '0'
    # For 6000 users, 100 keys per call, should take ~60 iterations. 1000 is a generous limit.
    max_scan_iterations = 1000
    current_iteration_count = 0

    try:
        while True:
            current_iteration_count += 1
            if current_iteration_count > max_scan_iterations:
                print(f"WARNING: query3 SCAN exceeded {max_scan_iterations} iterations. Returning results found so far.")
                break # Break if safety limit reached

            cursor, keys = self.r.scan(cursor=cursor, match="user:*", count=100)

            for key in keys:
                # Extract the numerical part of the user ID (e.g., "123" from "user:123")
                try:
                    user_id_num_str = key.split(':')[1]
                    if not user_id_num_str: # Skip if ID part is empty (e.g., "user:")
                        continue

                    first_digit = user_id_num_str[0] # Get the first digit of the number
```

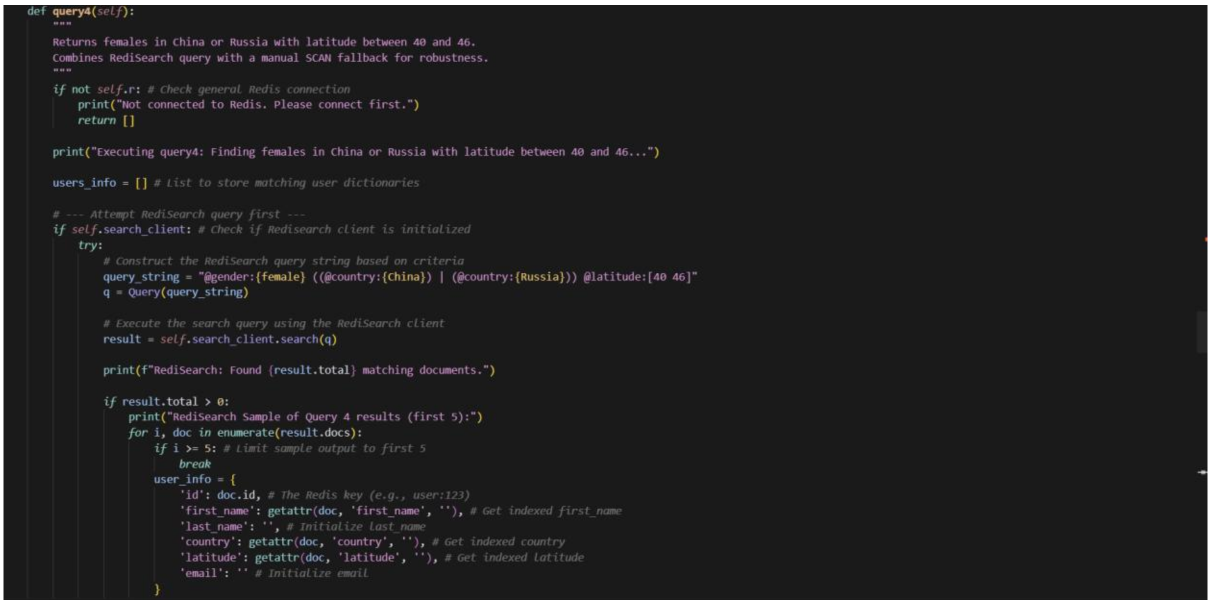


```python
                    if first_digit.isdigit() and first_digit in ['0', '2', '4', '6', '8']:
                        unique_matching_user_keys.add(key)
                except IndexError:
                    continue
                except ValueError:
                    continue

            if cursor == '0':
                break

        result_user_ids = sorted(list(unique_matching_user_keys)) #
        result_last_names = []
        for user_key in result_user_ids:
            last_name = self.r.hget(user_key, 'last_name')
            if last_name:
                result_last_names.append(last_name)
            else:
                result_last_names.append(None)

        print(f"Query3 complete. Found {len(result_user_ids)} matching users.")
        print("Sample of Query 3 results (first 5):")
        for uid, lastname in zip(result_user_ids[:5], result_last_names[:5]):
            print(f"  {uid}: {lastname}")

        return result_user_ids, result_last_names

    except Exception as e:
        print(f"An error occurred during query3: {e}")
        return [], []
```

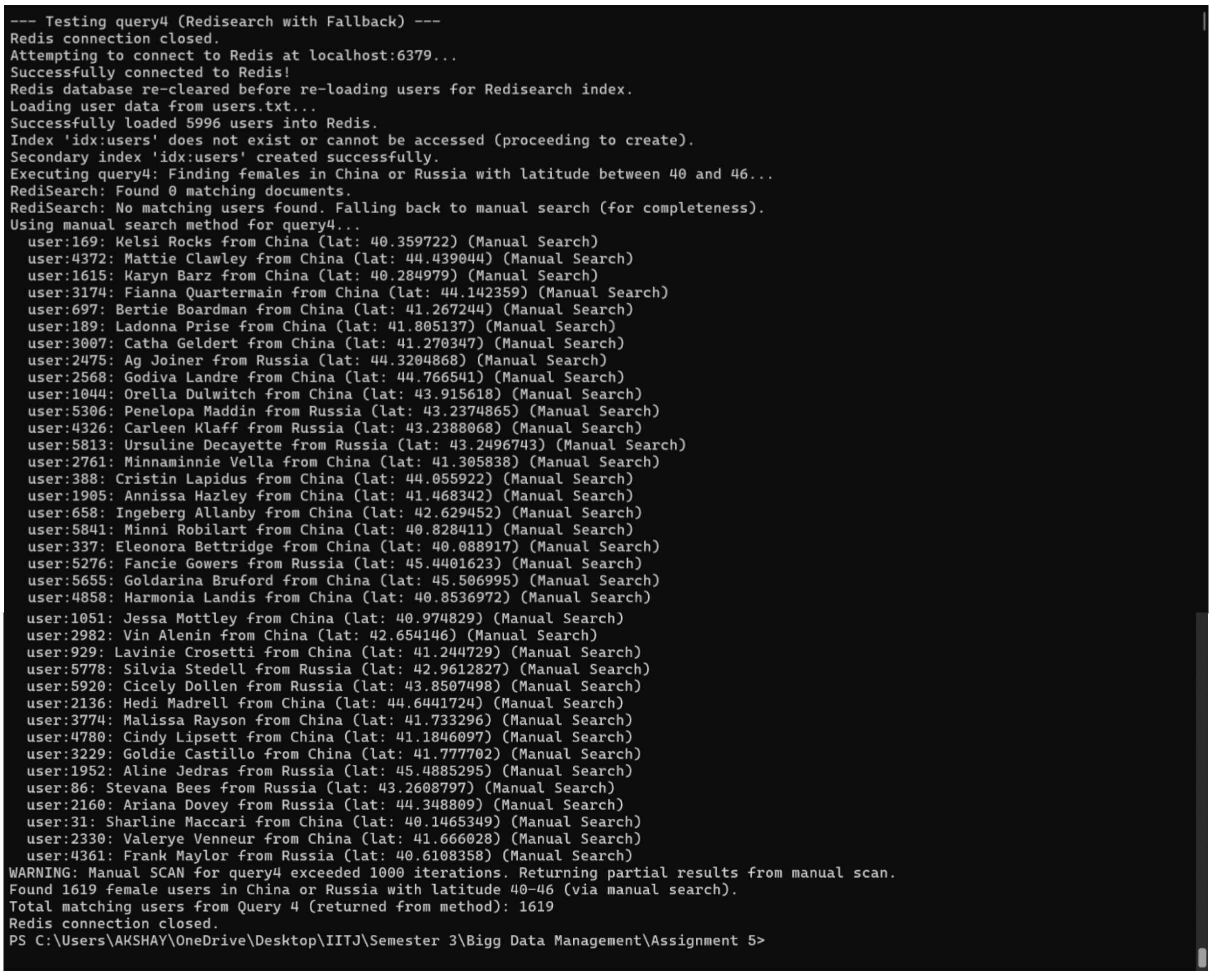


```
--- Testing query3 ---
Redis connection closed.
Attempting to connect to Redis at localhost:6379...
Successfully connected to Redis!
Executing query3: Getting user keys and last names (IDs not starting with odd numbers)...
WARNING: query3 SCAN exceeded 1000 iterations. Returning results found so far.
Query3 complete. Found 2444 matching users.
Sample of Query 3 results (first 5):
  user:2: Dewhurst
  user:20: Scullin
  user:200: Capron
  user:2000: Brodbin
  user:2001: Moreinu
Redis connection closed.
PS C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\Bigg Data Management\Assignment 5>
```

9. Query4() : This method creates a secondary index in Redisearch to enable efficient querying on specific user attributes like gender, country, latitude, and first name.

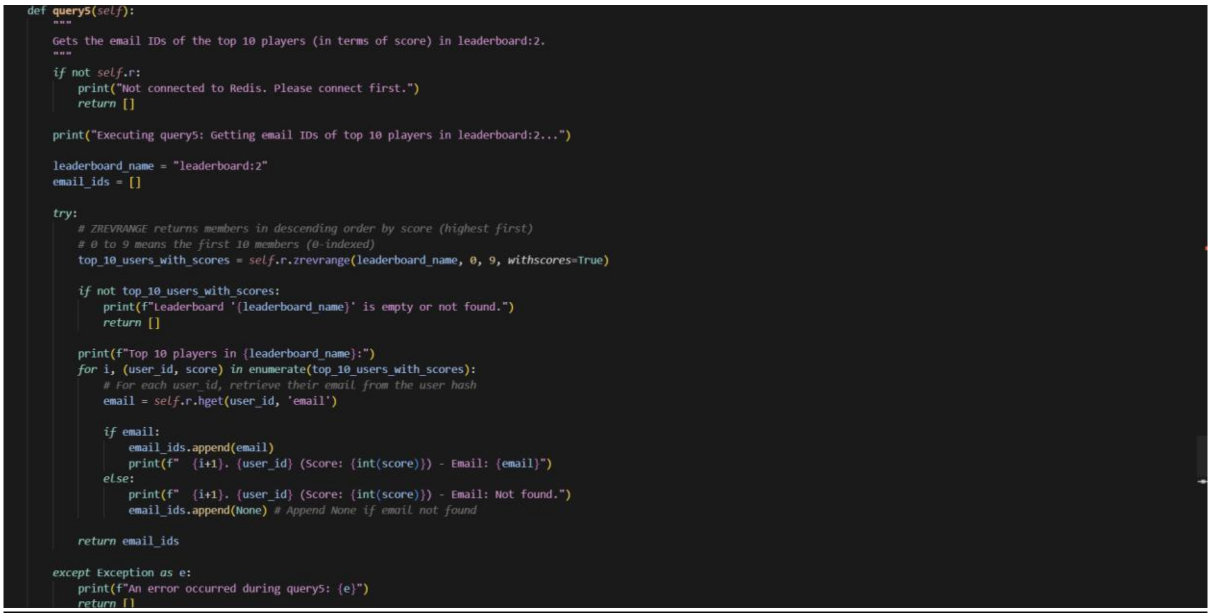


```python
def query4(self):
    """
    Returns females in China or Russia with latitude between 40 and 46.
    Combines RediSearch query with a manual SCAN fallback for robustness.
    """
    if not self.r: # Check general Redis connection
        print("Not connected to Redis. Please connect first.")
        return []

    print("Executing query4: Finding females in China or Russia with latitude between 40 and 46...")

    users_info = [] # List to store matching user dictionaries

    # --- Attempt RediSearch query first ---
    if self.search_client: # Check if Redisearch client is initialized
        try:
            # Construct the RediSearch query string based on criteria
            query_string = "@gender:{female} ((@country:{China}) | (@country:{Russia})) @latitude:[40 46]"
            q = Query(query_string)

            # Execute the search query using the RediSearch client
            result = self.search_client.search(q)

            print(f"RediSearch: Found {result.total} matching documents.")

            if result.total > 0:
                print("RediSearch Sample of Query 4 results (first 5):")
                for i, doc in enumerate(result.docs):
                    if i >= 5: # Limit sample output to first 5
                        break
                    user_info = {
                        'id': doc.id, # The Redis key (e.g., user:123)
                        'first_name': getattr(doc, 'first_name', ''), # Get indexed first_name
                        'last_name': '', # Initialize last_name
                        'country': getattr(doc, 'country', ''), # Get indexed country
                        'latitude': getattr(doc, 'latitude', ''), # Get indexed latitude
                        'email': '' # Initialize email
                    }
```

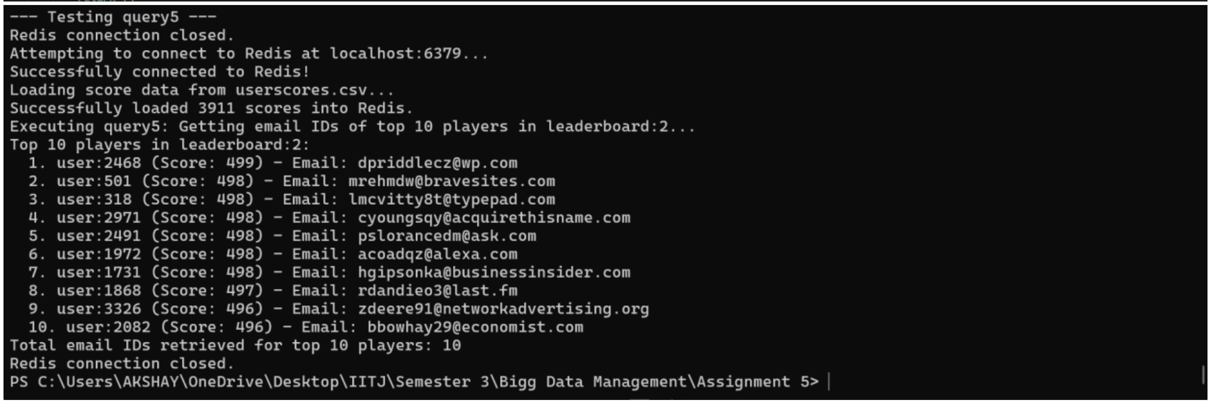


```
--- Testing query4 (Redisearch with Fallback) ---
Redis connection closed.
Attempting to connect to Redis at localhost:6379...
Successfully connected to Redis!
Redis database re-cleared before re-loading users for Redisearch index.
Loading user data from users.txt...
Successfully loaded 5996 users into Redis.
Index 'idx:users' does not exist or cannot be accessed (proceeding to create).
Secondary index 'idx:users' created successfully.
Executing query4: Finding females in China or Russia with latitude between 40 and 46...
RediSearch: Found 0 matching documents.
RediSearch: No matching users found. Falling back to manual search (for completeness).
Using manual search method for query4...
  user:169: Kelsi Rocks from China (lat: 40.359722) (Manual Search)
  user:4372: Mattie Clawley from China (lat: 44.439044) (Manual Search)
  user:1615: Karyn Barz from China (lat: 40.284979) (Manual Search)
  user:3174: Fianna Quartermain from China (lat: 44.142359) (Manual Search)
  user:697: Bertie Boardman from China (lat: 41.267244) (Manual Search)
  user:189: Ladonna Prise from China (lat: 41.805137) (Manual Search)
  user:3007: Catha Geldert from China (lat: 41.270347) (Manual Search)
  user:2475: Ag Joiner from Russia (lat: 44.3204868) (Manual Search)
  user:2568: Godiva Landre from China (lat: 44.766541) (Manual Search)
  user:1044: Orella Dulwitch from China (lat: 43.915618) (Manual Search)
  user:5306: Penelopa Maddin from Russia (lat: 43.2374865) (Manual Search)
  user:4326: Carleen Klaff from Russia (lat: 43.2388068) (Manual Search)
  user:5813: Ursuline Decayette from Russia (lat: 43.2496743) (Manual Search)
  user:2761: Minnaminnie Vella from China (lat: 41.305838) (Manual Search)
  user:388: Cristin Lapidus from China (lat: 44.055922) (Manual Search)
  user:1905: Annissa Hazley from China (lat: 41.468342) (Manual Search)
  user:658: Ingeberg Allanby from China (lat: 42.629452) (Manual Search)
  user:5841: Minni Robilart from China (lat: 40.828411) (Manual Search)
  user:337: Eleonora Bettridge from China (lat: 40.088917) (Manual Search)
  user:5276: Fancie Gowers from Russia (lat: 45.4401623) (Manual Search)
  user:5655: Goldarina Bruford from China (lat: 45.506995) (Manual Search)
  user:4858: Harmonia Landis from China (lat: 40.8536972) (Manual Search)
  user:1051: Jessa Mottley from China (lat: 40.974829) (Manual Search)
  user:2982: Vin Alenin from China (lat: 42.654146) (Manual Search)
  user:929: Lavinie Crosetti from China (lat: 41.244729) (Manual Search)
  user:5778: Silvia Stedell from Russia (lat: 42.9612827) (Manual Search)
  user:5920: Cicely Dollen from Russia (lat: 43.8507498) (Manual Search)
  user:2136: Hedi Madrell from China (lat: 44.6441724) (Manual Search)
  user:3774: Malissa Rayson from China (lat: 41.733296) (Manual Search)
  user:4780: Cindy Lipsett from China (lat: 41.1846097) (Manual Search)
  user:3229: Goldie Castillo from China (lat: 41.777702) (Manual Search)
  user:1952: Aline Jedras from Russia (lat: 45.4885295) (Manual Search)
  user:86: Stevana Bees from Russia (lat: 43.2608797) (Manual Search)
  user:2160: Ariana Dovey from Russia (lat: 44.348809) (Manual Search)
  user:31: Sharline Maccari from China (lat: 40.1465349) (Manual Search)
  user:2330: Valerye Venneur from China (lat: 41.666028) (Manual Search)
  user:4361: Frank Maylor from Russia (lat: 40.6108358) (Manual Search)
WARNING: Manual SCAN for query4 exceeded 1000 iterations. Returning partial results from manual scan.
Found 1619 female users in China or Russia with latitude 40-46 (via manual search).
Total matching users from Query 4 (returned from method): 1619
Redis connection closed.
PS C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\Bigg Data Management\Assignment 5>
```

10. Query5() : This method finds female users in China or Russia with a specific latitude range (40 to 46), first attempting a RediSearch query and falling back to a manual scan if RediSearch fails or yields no results.



```python
def query5(self):
    """
    Gets the email IDs of the top 10 players (in terms of score) in leaderboard:2.
    """
    if not self.r:
        print("Not connected to Redis. Please connect first.")
        return []

    print("Executing query5: Getting email IDs of top 10 players in leaderboard:2...")

    leaderboard_name = "leaderboard:2"
    email_ids = []

    try:
        # ZREVRANGE returns members in descending order by score (highest first)
        # 0 to 9 means the first 10 members (0-indexed)
        top_10_users_with_scores = self.r.zrevrange(leaderboard_name, 0, 9, withscores=True)

        if not top_10_users_with_scores:
            print(f"Leaderboard '{leaderboard_name}' is empty or not found.")
            return []

        print(f"Top 10 players in {leaderboard_name}:")
        for i, (user_id, score) in enumerate(top_10_users_with_scores):
            # For each user_id, retrieve their email from the user hash
            email = self.r.hget(user_id, 'email')

            if email:
                email_ids.append(email)
                print(f"  {i+1}. {user_id} (Score: {int(score)}) - Email: {email}")
            else:
                print(f"  {i+1}. {user_id} (Score: {int(score)}) - Email: Not found.")
                email_ids.append(None) # Append None if email not found

        return email_ids

    except Exception as e:
        print(f"An error occurred during query5: {e}")
        return []
```



```
--- Testing query5 ---
Redis connection closed.
Attempting to connect to Redis at localhost:6379...
Successfully connected to Redis!
Loading score data from userscores.csv...
Successfully loaded 3911 scores into Redis.
Executing query5: Getting email IDs of top 10 players in leaderboard:2...
Top 10 players in leaderboard:2:
  1. user:2468 (Score: 499) - Email: dpriddlecz@wp.com
  2. user:501 (Score: 498) - Email: mrehmdw@bravesites.com
  3. user:318 (Score: 498) - Email: lmcvitty8t@typepad.com
  4. user:2971 (Score: 498) - Email: cyoungsqy@acquirethisname.com
  5. user:2491 (Score: 498) - Email: pslorancedm@ask.com
  6. user:1972 (Score: 498) - Email: acoadqz@alexa.com
  7. user:1731 (Score: 498) - Email: hgipsonka@businessinsider.com
  8. user:1868 (Score: 497) - Email: rdandieo3@last.fm
  9. user:3326 (Score: 496) - Email: zdeere91@networkadvertising.org
  10. user:2082 (Score: 496) - Email: bbowhay29@economist.com
Total email IDs retrieved for top 10 players: 10
Redis connection closed.
PS C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\Bigg Data Management\Assignment 5>
```