

BIG DATA MANAGEMENT

POST GRADUATE DIPLOMA IN DATA ENGINEERING

ASSIGNMENT - 5

SUBMITTED BY:

NIRAJ BHAGCHANDANI [G23AI2087]



SUBMISSION DATE: 15th December, 2024

**DEPARTMENT OF AIDE
INDIAN INSTITUTE OF TECHNOLOGY, JODHPUR**

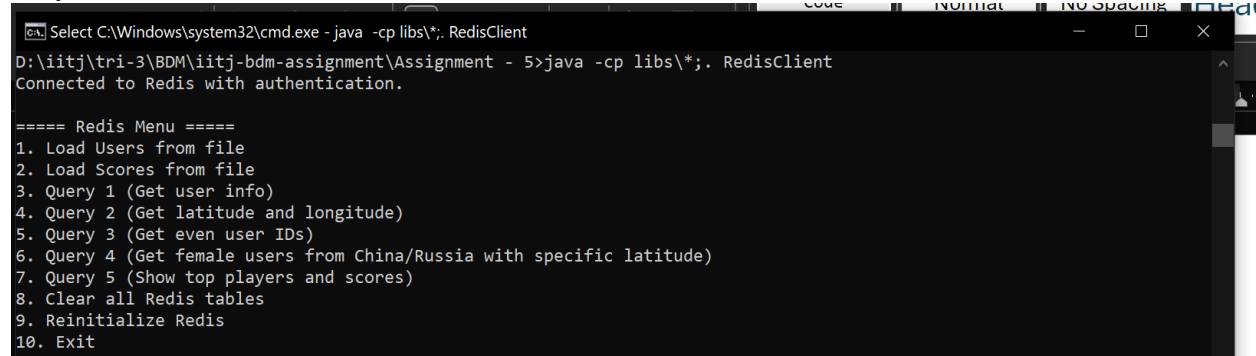
Read about redis-py and redis-cli to connect to your cloud redis database.

1. Write the method connect() to create a connection to Redis. [5]

Note: Instead of writing it in connect() I have created the constructor method to make my code life easier for better execution.

```
public RedisClient() {
    try {
        jedis = new Jedis("redis-16150.c16.us-east-1-3.ec2.redns.redis-
cloud.com", 16150);
        String password = "0ahVB0ixFG49ILZK5qvC6QG6BmujVgHC"; // Replace with
your Redis password
        jedis.auth(password);
        System.out.println("Connected to Redis with authentication.");
    } catch (JedisException e) {
        e.printStackTrace();
    }
}
```

Output:



```
Select C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient
D:\iitj\tri-3\BDM\iitj-bdm-assignment\Assignment - 5>java -cp libs\*. RedisClient
Connected to Redis with authentication.

===== Redis Menu =====
1. Load Users from file
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
```

As the constructor initialized by itself, and I have made call constructor instead of the connect() method.

2. Write the methods load_users() and load_scores() to load the data into the redis db.
Use appropriate data structures. Provide details of the Redis data structures that you are using. [5]

load_users() method:

```
public void loadUsers(String filePath) {
    try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
        String line;
        int totalLines = 0;
        int linesProcessed = 0;

        // Count total lines for progress tracking
        while ((line = br.readLine()) != null) {
            totalLines++;
        }

        br.close();
    }
```

```

// Reopen the file to process data
BufferedReader br2 = new BufferedReader(new FileReader(filePath));
while ((line = br2.readLine()) != null) {
    line = line.trim();
    if (line.isEmpty()) continue; // Skip empty lines

    String[] parts = line.split(" (?=(?:[^\"]*"\"[^\"]*"")*(?:\\\"))");
    // Split while preserving quoted text
    if (parts.length < 2) {
        System.out.println("Skipping malformed line: " + line);
        continue; // Skip lines that don't match the expected format
    }
    String key = parts[0].replace("\\\"", ""); // Remove quotes from
the key
    Map<String, String> user = new HashMap<>();
    for (int i = 1; i < parts.length; i += 2) {
        if (i + 1 < parts.length) {
            String attributeKey = parts[i].replace("\\\"", "");
            String attributeValue = parts[i + 1].replace("\\\"", "");
            user.put(attributeKey, attributeValue);
        } else {
            System.out.println("Skipping incomplete key-value pair in
line: " + line);
        }
    }
    jedis.hset(key, user);

    // Update progress
    linesProcessed++;
    System.out.printf("Progress: %d/%d lines uploaded (%.2f%%
complete)\n", linesProcessed, totalLines, (linesProcessed / (double) totalLines)
* 100);
}

br2.close();
System.out.println("Users loaded successfully.");
} catch (IOException e) {
    e.printStackTrace();
}
}

```

load_scoares() method

```
public void loadScores(String filename) {
    try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
        String line;
        boolean firstLine = true; // To skip the header line
        while ((line = br.readLine()) != null) {
            if (firstLine) {
                firstLine = false;
                continue;
            }
        }
    }
}
```

```

        String[] values = line.split(",");
        if (values.length < 3) {
            System.out.println("Skipping malformed row: " + line);
            continue;
        }
        String userId = values[0].trim();
        try {
            double score = Double.parseDouble(values[1].trim());
            if (values[2].trim().isEmpty()) {
                System.out.println("Skipping row with missing leaderboard
value: " + line);
                continue;
            }
            int leaderboard = (int) Double.parseDouble(values[2].trim());

            users.put(userId, new User(userId, score, leaderboard));
        } catch (NumberFormatException e) {
            System.out.println("Error parsing score or leaderboard for "
+ userId + " in row: " + line);
            e.printStackTrace();
        }
    }
    System.out.println("Scores loaded successfully.");
} catch (IOException e) {
    e.printStackTrace();
}
}

```

Output:

User.txt

```

C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 1
Enter the file path for users: users.txt
Progress: 1/5996 lines uploaded (0.02% complete)
Progress: 2/5996 lines uploaded (0.03% complete)
Progress: 3/5996 lines uploaded (0.05% complete)
Progress: 4/5996 lines uploaded (0.07% complete)
Progress: 5/5996 lines uploaded (0.08% complete)
Progress: 6/5996 lines uploaded (0.10% complete)
Progress: 7/5996 lines uploaded (0.12% complete)
Progress: 8/5996 lines uploaded (0.13% complete)
Progress: 9/5996 lines uploaded (0.15% complete)
Progress: 10/5996 lines uploaded (0.17% complete)
Progress: 11/5996 lines uploaded (0.18% complete)
Progress: 12/5996 lines uploaded (0.20% complete)
Progress: 13/5996 lines uploaded (0.22% complete)
Progress: 14/5996 lines uploaded (0.23% complete)
Progress: 15/5996 lines uploaded (0.25% complete)
Progress: 16/5996 lines uploaded (0.27% complete)
Progress: 17/5996 lines uploaded (0.28% complete)
Progress: 18/5996 lines uploaded (0.30% complete)

```

```

C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient
Progress: 5969/5996 lines uploaded (99.55% complete)
Progress: 5970/5996 lines uploaded (99.57% complete)
Progress: 5971/5996 lines uploaded (99.58% complete)
Progress: 5972/5996 lines uploaded (99.60% complete)
Progress: 5973/5996 lines uploaded (99.62% complete)
Progress: 5974/5996 lines uploaded (99.63% complete)
Progress: 5975/5996 lines uploaded (99.65% complete)
Progress: 5976/5996 lines uploaded (99.67% complete)
Progress: 5977/5996 lines uploaded (99.68% complete)
Progress: 5978/5996 lines uploaded (99.70% complete)
Progress: 5979/5996 lines uploaded (99.72% complete)
Progress: 5980/5996 lines uploaded (99.73% complete)
Progress: 5981/5996 lines uploaded (99.75% complete)
Progress: 5982/5996 lines uploaded (99.77% complete)
Progress: 5983/5996 lines uploaded (99.78% complete)
Progress: 5984/5996 lines uploaded (99.80% complete)
Progress: 5985/5996 lines uploaded (99.82% complete)
Progress: 5986/5996 lines uploaded (99.83% complete)
Progress: 5987/5996 lines uploaded (99.85% complete)
Progress: 5988/5996 lines uploaded (99.87% complete)
Progress: 5989/5996 lines uploaded (99.88% complete)
Progress: 5990/5996 lines uploaded (99.90% complete)
Progress: 5991/5996 lines uploaded (99.92% complete)
Progress: 5992/5996 lines uploaded (99.93% complete)
Progress: 5993/5996 lines uploaded (99.95% complete)
Progress: 5994/5996 lines uploaded (99.97% complete)
Progress: 5995/5996 lines uploaded (99.98% complete)
Progress: 5996/5996 lines uploaded (100.00% complete)
Users loaded successfully.

```

UserScores.csv

```

C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient

===== Redis Menu =====
1. Load Users from file
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 2
Enter the file path for scores: userscores.csv
Scores loaded successfully.

```

3. Write the method query1() that returns all the attributes of the user by usr. [5]

```

public Map<String, String> query1(String userId) {
    return jedis.hgetAll(userId);
}

```

Output:

```

C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient
Connected to Redis with authentication.

===== Redis Menu =====
1. Load Users from file
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 3
Enter user ID: user:1
{country_code=CN, country=China, gender=male, city=Yuanjue, last_login=1581151007, latitude=29.55451, last_name=Ahern, i
p_address=180.132.241.207, first_name=Mohammed, email=mahern0@amazon.com, longitude=105.324979}

```

4. Write the method query2() that the coordinate (longitude and latitude) of the user by the user. [5]

```

public String query2(String userId) {
    String latitude = jedis.hget(userId, "latitude");
    String longitude = jedis.hget(userId, "longitude");
    return "Latitude: " + latitude + ", Longitude: " + longitude;
}

```

Output:

```

C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 4
Enter user ID: user:2
Latitude: 45.9260128, Longitude: 20.0780937

```

5. Write the method query3() that get the keys and last names of the users whose ids do not start with an odd number. [5]

```

public Map<String, String> query3() {
    Map<String, String> result = new HashMap<>();
    String cursor = "0"; // Initial cursor value
    int acceptedCount = 0; // Counter for accepted keys
    int rejectedCount = 0; // Counter for rejected keys
    int totalScanned = 0; // Counter for total scanned keys
    int batchSize = 500; // Number of records to scan in each iteration

    do {
        // Scan for keys matching "user:*" with higher batch size
        ScanResult<String> scanResult = jedis.scan(cursor, new
        ScanParams().match("user:*").count(batchSize));
        cursor = scanResult.getCursor(); // Update cursor
        List<String> keys = scanResult.getResult();
        totalScanned += keys.size();
    }
}

```

```

        // Use pipelining to fetch `last_name` attributes for all keys in the
batch
        List<Object> lastNames;
        try (var pipeline = jedis.pipelined()) { // Open a pipeline session
            for (String key : keys) {
                pipeline.hget(key, "last_name");
            }
            lastNames = pipeline.syncAndReturnAll(); // Execute all commands
in the pipeline
        }

        // Process each key and corresponding last name
        for (int i = 0; i < keys.size(); i++) {
            String key = keys.get(i);
            String lastName = (String) lastNames.get(i); // Cast pipeline
result to String

            // Extract the user ID from the key (assumes key format is
"user:<id>")
            String[] keyParts = key.split(":");
            if (keyParts.length < 2) {
                rejectedCount++;
                continue; // Invalid key format
            }

            String userIdStr = keyParts[1];
            if (!userIdStr.isEmpty() &&
Character.isDigit(userIdStr.charAt(0))) {
                int firstDigit =
Character.getNumericValue(userIdStr.charAt(0));
                if (firstDigit % 2 == 0) { // Check if the first digit is
even
                    if (lastName != null) {
                        result.put(key, lastName);
                        acceptedCount++;
                    } else {
                        rejectedCount++; // Last name is null
                    }
                } else {
                    rejectedCount++; // First digit is odd
                }
            } else {
                rejectedCount++; // First character not a digit or ID is
empty
            }
        }

        // Progress report after every 500 records
        if (totalScanned % 500 == 0) {
            System.out.println("Processed: " + totalScanned + " records so
far.");
            System.out.println("Accepted: " + acceptedCount + ", Rejected: "
+ rejectedCount);
        }

```

```

        } while (!cursor.equals("0")); // Continue scanning until the cursor
        loops back to "0"

        // Final summary
        System.out.println("Query complete.");
        System.out.println("Total scanned: " + totalScanned + " records.");
        System.out.println("Accepted keys: " + acceptedCount + ", Rejected keys:
" + rejectedCount);

        return result;
    }

```

The screenshot shows a Java application window titled "C:\Windows\system32\cmd.exe - java -cp libs*. RedisClient". The application displays a menu with 10 options. The user has entered choice 4, then choice 5. The application shows the results of processing records, including total scanned, accepted, and rejected counts.

```

C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 4
Enter user ID: user:2
Latitude: 45.9260128, Longitude: 20.0780937

==== Redis Menu ====
1. Load Users from file
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 5
Processed: 500 records so far.
Accepted: 196, Rejected: 304
Processed: 1000 records so far.
Accepted: 405, Rejected: 595
Processed: 1500 records so far.
Accepted: 599, Rejected: 901

```



```

C:\Windows\system32\cmd.exe - java -cp libs*; RedisClient
Enter your choice: 5
Processed: 500 records so far.
Accepted: 196, Rejected: 304
Processed: 1000 records so far.
Accepted: 405, Rejected: 595
Processed: 1500 records so far.
Accepted: 599, Rejected: 901
Query complete.
Total scanned: 5996 records.
Accepted keys: 2444, Rejected keys: 3552
{user:2403=Stracey, user:2887=Penlington, user:2888=Cartledge, user:2404=Megahey, user:2405=Wyld, user:2889=Paulsen, user:2406=McFeat, user:2883=Cromwell, user:2400=Heartfield, user:2884=Bechley, user:2401=Winspurr, user:2885=Harrap, user:2402=Nerney, user:2886=Creasy, user:2407=Alpin, user:2408=MacMaykin, user:2409=Tosspell, user:2880=Cacacie, user:2881=Doohan, user:2882=Jiru, user:2898=Way, user:2414=Dunguy, user:2899=Osichev, user:2415=Gritland, user:2416=Guilloux, user:2417=Veldman, user:2894=Kehir, user:2410=Spriggs, user:2895=Spafford, user:2411=Pelcheur, user:2896=Orrobin, user:2412=McKerron, user:2413=Holworth, user:2897=Shippey, user:2418=Gallamore, user:2419=Piggens, user:2890=Ennever, user:2891=Varfolomeev, user:2892=Jaulme, user:2893=Harms, user:2425=Gatecliffe, user:4604=Coggles, user:4603=Tort, user:2426=Clowes, user:4606=Hinnerk, user:2427=Yushankin, user:2428=Arkill, user:4605=Stuck, user:2421=Joye, user:4600=Bouldon, user:2422=Ponde, user:2423=Gurner, user:4602=Jacob, user:2424=Choppin, user:4601=Rosenbush, user:2429=Byram, user:4608=Leckie, user:4607=Ponter, user:4609=Paradis, user:2420=Canfer, user:2436=McVeighy, user:4615=Dougharty, user:4614=Sibson, user:2437=Whittam, user:4617=Triggol, user:2438=Auletta, user:2439=Upstell, user:4616=D'Aguanno, user:2432=Widd, user:4611=Collington, user:2433=Strange, user:4610=Langworthy, user:2434=Harmer, user:4613=Locket, user:4612=Bradberry, user:2435=Rahill, user:4619=Bingell, user:4618=Gilding, user:2430=Willcott, user:2431=Tinklin, user:4626=Skokoe, user:2447=D'Alesco, user:2448=Thams, user:4625=Dronsfield, user:2449=Petworth, user:4628=St. Louis, user:4627=Fyldes, user:2443=Luter, user:4622=Lundy, user:4621=Biddy, user:2444=Ortsmann, user:4624=Rowles, user:2445=Dybell, user:4623=Gilcrist, user:2446=Krystek, user:4629=Whitland, user:2440=Durand, user:2441=Weddup, user:4620=Hinkens, user:2442=Shimman, user:2458=Buye, user:4637=Filochov, user:2459=Pischoff, user:4636=Major, user:4639=Schwand, user:4638=Lermit, user:4633=Tester, user:2454=McKissack, user:4632=Lodemann, user:2455=Bywaters, user:2456=Heinonen, user:4635=Osmant, user:2457=Glaisner, user:4634=Worsnop, user:2450=Westman, user:2451=Somersett, user:4631=Astlett, user:2452=Cannop, user:2453=Ebbage, user:4630=Singh, user:4648=Sellors, user:2469=Pervoe, user:4647=Conquest, user:4649=Fitzackerley, user:4644=Symons, user:2465=Higgoe, user:4643=

```

6. Write the method query4() that returns the female in China or Russia with a latitude between 40 and 46. [5]

```

public List<String> query4() {
    List<String> result = new ArrayList<>();
    String cursor = "0";
    int acceptedCount = 0; // Counter for accepted keys
    int rejectedCount = 0; // Counter for rejected keys

    do {
        // Scan for keys matching "user:*"
        ScanResult<String> scanResult = jedis.scan(cursor, new
        ScanParams().match("user:*").count(100));
        cursor = scanResult.getCursor();

        System.out.println("Current cursor: " + cursor); // Debugging output
        to track cursor progress
        System.out.println("Scanning " + scanResult.getResult().size() + "
        keys...");

        for (String key : scanResult.getResult()) {
            try {
                // Fetch necessary attributes
                String gender = jedis.hget(key, "gender");
                String country = jedis.hget(key, "country");
                String latitudeStr = jedis.hget(key, "latitude");

                if (latitudeStr != null) {
                    double latitude = Double.parseDouble(latitudeStr);

```

```

        // Check conditions
        if ("female".equalsIgnoreCase(gender) &&
            ("China".equalsIgnoreCase(country) ||
"Russia".equalsIgnoreCase(country)) &&
            (latitude >= 40 && latitude <= 46)) {
            result.add(key);
            System.out.println("Accepted key: " + key +
                " (Gender: " + gender + ", Country: " + country +
", Latitude: " + latitude + ")");
            acceptedCount++;
        } else {
            //System.out.println("Rejected key: " + key +
            //    " (Gender: " + gender + ", Country: " + country
+ ", Latitude: " + latitude + ")");
            rejectedCount++;
        }
    } else {
        //System.out.println("Rejected key: " + key + " (Latitude
is null)");
        rejectedCount++;
    }
} catch (NumberFormatException e) {
    //System.out.println("Rejected key: " + key + " (Latitude
parsing error)");
    rejectedCount++;
} catch (NullPointerException e) {
    //System.out.println("Rejected key: " + key + " (Missing
required fields)");
    rejectedCount++;
}
}
} while (!cursor.equals("0")); // Continue scanning until the cursor
loops back to "0"

// Summary
System.out.println("Query complete. Found " + result.size() + " matching
users.");
System.out.println("Accepted keys: " + acceptedCount + ", Rejected keys:
" + rejectedCount);

return result;
}

```

Output:

```

C:\Windows\system32\cmd.exe - java -cp libs\*; RedisClient

===== Redis Menu =====
1. Load Users from file
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 6
Current cursor: 5152
Scanning 100 keys...
Accepted key: user:1576 (Gender: female, Country: Russia, Latitude: 45.8697083)
Accepted key: user:3649 (Gender: female, Country: China, Latitude: 41.234822)
Accepted key: user:5655 (Gender: female, Country: China, Latitude: 45.506995)
Current cursor: 6160
Scanning 100 keys...
Accepted key: user:5875 (Gender: female, Country: China, Latitude: 44.34072)
Accepted key: user:337 (Gender: female, Country: China, Latitude: 40.088917)
Accepted key: user:4652 (Gender: female, Country: China, Latitude: 40.608793)
Accepted key: user:4579 (Gender: female, Country: Russia, Latitude: 43.2374865)
Current cursor: 3376
Scanning 100 keys...
Accepted key: user:1708 (Gender: female, Country: Russia, Latitude: 45.1242818)
Current cursor: 6792
Scanning 101 keys...
Current cursor: 1448
Scanning 100 keys...

```

```

C:\Windows\system32\cmd.exe - java -cp libs\*; RedisClient

Scanning 100 keys...
Accepted key: user:5251 (Gender: female, Country: China, Latitude: 43.9912932)
Current cursor: 6391
Scanning 100 keys...
Accepted key: user:4645 (Gender: female, Country: Russia, Latitude: 43.4265165)
Current cursor: 1487
Scanning 100 keys...
Accepted key: user:2160 (Gender: female, Country: Russia, Latitude: 44.348809)
Accepted key: user:929 (Gender: female, Country: China, Latitude: 41.244729)
Current cursor: 2591
Scanning 101 keys...
Accepted key: user:1698 (Gender: female, Country: Russia, Latitude: 45.05426)
Accepted key: user:4038 (Gender: female, Country: Russia, Latitude: 43.57344)
Current cursor: 1855
Scanning 100 keys...
Accepted key: user:1642 (Gender: female, Country: China, Latitude: 42.697778)
Current cursor: 0
Scanning 73 keys...
Accepted key: user:2489 (Gender: female, Country: Russia, Latitude: 43.5007512)
Query complete. Found 97 matching users.
Accepted keys: 97, Rejected keys: 5899
[user:1576, user:3649, user:5655, user:5875, user:337, user:4652, user:4579, user:1708, user:5775, user:5306, user:4588,
user:1677, user:2761, user:4148, user:3774, user:3944, user:2554, user:2289, user:1051, user:5162, user:4402, user:4929
, user:5698, user:86, user:2136, user:3007, user:2330, user:1375, user:3213, user:3174, user:5280, user:4945, user:189,
user:5868, user:2627, user:1952, user:3300, user:4780, user:4781, user:5920, user:1905, user:5238, user:897, user:2568,
user:2392, user:1615, user:4991, user:5778, user:509, user:4858, user:1143, user:4451, user:116, user:4364, user:2982, u
ser:4326, user:494, user:3343, user:1044, user:3099, user:4197, user:5977, user:2106, user:3229, user:824, user:388, use
r:2475, user:2771, user:697, user:5841, user:3542, user:4785, user:1163, user:3550, user:169, user:31, user:5276, user:4
085, user:2266, user:3568, user:5813, user:673, user:492, user:2851, user:4361, user:5597, user:1662, user:4372, user:65
8, user:5251, user:4645, user:2160, user:929, user:1698, user:4038, user:1642, user:2489]

```

7. Write the method query5() that gets the email ids of the top 10 players(in terms of score) in leaderboard:2. [5]

```

public void query5() {
    if (!jedis.exists("leaderboard:2")) {

```

```

        System.out.println("Leaderboard data does not exist.");
        return;
    }

    // Fetch the top 10 players based on their scores
    Set<String> topPlayers = jedis.zrevrange("leaderboard:2", 0, 9);
    if (topPlayers.isEmpty()) {
        System.out.println("No players found in leaderboard:2.");
        return;
    }

    // Use pipelining to fetch scores and emails in bulk
    try (var pipeline = jedis.pipelined()) {
        List<Response<Double>> scores = new ArrayList<>();
        List<Response<String>> emails = new ArrayList<>();

        // Queue the commands in the pipeline
        for (String playerName : topPlayers) {
            scores.add(pipeline.zscore("leaderboard:2", playerName));
            emails.add(pipeline.hget(playerName, "email"));
        }

        // Execute all commands
        pipeline.sync();

        // Process the results
        int rank = 1;
        Iterator<String> playerIterator = topPlayers.iterator();
        for (int i = 0; i < topPlayers.size(); i++) {
            String playerName = playerIterator.next();
            Double score = scores.get(i).get();
            String email = emails.get(i).get();

            System.out.println("Rank: " + rank + " | Player: " + playerName +
                " | Score: " + score + " | Email: " + email);
            rank++;
        }
    }
}

```

Output:

```

C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient

D:\iitj\tri-3\BDM\iitj-bdm-assignment\Assignment - 5>java -cp libs\*. RedisClient
Connected to Redis with authentication.

===== Redis Menu =====
1. Load Users from file
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 7
Leaderboard data does not exist.

```

Full Code:

```

import redis.clients.jedis.Jedis;
import redis.clients.jedis.exceptions.JedisException;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.Set;
import redis.clients.jedis.Response;
import redis.clients.jedis.Pipeline;
import java.util.Iterator; // For iterators
import redis.clients.jedis.ScanResult;
import redis.clients.jedis.ScanParams;

public class RedisClient {
    private Jedis jedis;
    private Map<String, User> users = new HashMap<>();

    public RedisClient() {
        try {
            jedis = new Jedis("redis-10554.c257.us-east-1-3.ec2.redns.redis-
cloud.com", 10554);
            String password = "QYG2W3RqFDbYAGs1SeUr30iQCvTFpGYt"; // Replace with
your Redis password
            jedis.auth(password);
            System.out.println("Connected to Redis with authentication.");
        } catch (JedisException e) {
            e.printStackTrace();
        }
    }

    public void loadUsers(String filePath) {
        try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
            String line;

```

```

        int totalLines = 0;
        int linesProcessed = 0;

        // Count total lines for progress tracking
        while ((line = br.readLine()) != null) {
            totalLines++;
        }

        br.close();

        BufferedReader br2 = new BufferedReader(new FileReader(filePath));
        while ((line = br2.readLine()) != null) {
            line = line.trim();
            if (line.isEmpty()) continue; // Skip empty lines

            String[] parts = line.split(" (?=(?:[^\"]*" * "[^\"]*" * "[^\"]*" * "$)");
            if (parts.length < 2) {
                System.out.println("Skipping malformed line: " + line);
                continue; // Skip lines that don't match the expected format
            }
            String key = parts[0].replace("\"", ""); // Remove quotes from
the key
            Map<String, String> user = new HashMap<>();
            for (int i = 1; i < parts.length; i += 2) {
                if (i + 1 < parts.length) {
                    String attributeKey = parts[i].replace("\"", "");
                    String attributeValue = parts[i + 1].replace("\"", "");
                    user.put(attributeKey, attributeValue);
                } else {
                    System.out.println("Skipping incomplete key-value pair in
line: " + line);
                }
            }
            jedis.hset(key, user);
            linesProcessed++;
            System.out.printf("Progress: %d/%d lines uploaded (%.2f%%
complete)\n", linesProcessed, totalLines, (linesProcessed / (double) totalLines)
* 100);
        }

        br2.close();
        System.out.println("Users loaded successfully.");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void loadScores(String filename) {
    try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
        String line;
        boolean firstLine = true; // To skip the header line
        while ((line = br.readLine()) != null) {
            if (firstLine) {
                firstLine = false;
            }
        }
    }
}

```

```

        continue;
    }
    String[] values = line.split(",");
    if (values.length < 3) {
        System.out.println("Skipping malformed row: " + line);
        continue;
    }
    String userId = values[0].trim();
    try {
        double score = Double.parseDouble(values[1].trim());
        if (values[2].trim().isEmpty()) {
            System.out.println("Skipping row with missing leaderboard
value: " + line);
            continue;
        }
        int leaderboard = (int) Double.parseDouble(values[2].trim());

        users.put(userId, new User(userId, score, leaderboard));
    } catch (NumberFormatException e) {
        System.out.println("Error parsing score or leaderboard for "
+ userId + " in row: " + line);
        e.printStackTrace();
    }
}
System.out.println("Scores loaded successfully.");
} catch (IOException e) {
    e.printStackTrace();
}
}

public Map<String, String> query1(String userId) {
    return jedis.hgetAll(userId);
}

public String query2(String userId) {
    String latitude = jedis.hget(userId, "latitude");
    String longitude = jedis.hget(userId, "longitude");
    return "Latitude: " + latitude + ", Longitude: " + longitude;
}

public Map<String, String> query3() {
    Map<String, String> result = new HashMap<>();
    String cursor = "0";
    int acceptedCount = 0;
    int rejectedCount = 0;
    int totalScanned = 0;
    int batchSize = 500;

    do {
        ScanResult<String> scanResult = jedis.scan(cursor, new
ScanParams().match("user:*").count(batchSize));
        cursor = scanResult.getCursor(); // Update cursor
        List<String> keys = scanResult.getResult();
        totalScanned += keys.size();
    }
}

```

```

        List<Object> lastNames;
        try (var pipeline = jedis.pipelined()) { // Open a pipeline session
            for (String key : keys) {
                pipeline.hget(key, "last_name");
            }
            lastNames = pipeline.syncAndReturnAll(); // Execute all commands
in the pipeline
        }

        for (int i = 0; i < keys.size(); i++) {
            String key = keys.get(i);
            String lastName = (String) lastNames.get(i); // Cast pipeline
result to String

            String[] keyParts = key.split(":");
            if (keyParts.length < 2) {
                rejectedCount++;
                continue; // Invalid key format
            }

            String userIdStr = keyParts[1];
            if (!userIdStr.isEmpty() &&
Character.isDigit(userIdStr.charAt(0))) {
                int firstDigit =
Character.getNumericValue(userIdStr.charAt(0));
                if (firstDigit % 2 == 0) {
                    if (lastName != null) {
                        result.put(key, lastName);
                        acceptedCount++;
                    } else {
                        rejectedCount++;
                    }
                } else {
                    rejectedCount++;
                }
            } else {
                rejectedCount++;
            }
        }
        if (totalScanned % 500 == 0) {
            System.out.println("Processed: " + totalScanned + " records so
far.");
            System.out.println("Accepted: " + acceptedCount + ", Rejected: "
+ rejectedCount);
        }

    } while (!cursor.equals("0"));

    System.out.println("Query complete.");
    System.out.println("Total scanned: " + totalScanned + " records.");
    System.out.println("Accepted keys: " + acceptedCount + ", Rejected keys:
" + rejectedCount);

    return result;

```



```

    }

    public List<String> query4() {
        List<String> result = new ArrayList<>();
        String cursor = "0";
        int acceptedCount = 0;
        int rejectedCount = 0;

        do {
            // Scan for keys matching "user:*"
            ScanResult<String> scanResult = jedis.scan(cursor, new
ScanParams().match("user:*").count(100));
            cursor = scanResult.getCursor();

            System.out.println("Current cursor: " + cursor); // Debugging output
to track cursor progress
            System.out.println("Scanning " + scanResult.getResult().size() + "
keys...");

            Pipeline pipeline = jedis.pipelined();
            Map<String, Response<String>> genderResponses = new HashMap<>();
            Map<String, Response<String>> countryResponses = new HashMap<>();
            Map<String, Response<String>> latitudeResponses = new HashMap<>();

            for (String key : scanResult.getResult()) {

                genderResponses.put(key, pipeline.hget(key, "gender"));
                countryResponses.put(key, pipeline.hget(key, "country"));
                latitudeResponses.put(key, pipeline.hget(key, "latitude"));
            }

            pipeline.sync();

            for (String key : scanResult.getResult()) {
                try {
                    String gender = genderResponses.get(key).get();
                    String country = countryResponses.get(key).get();
                    String latitudeStr = latitudeResponses.get(key).get();

                    if (latitudeStr != null) {
                        double latitude = Double.parseDouble(latitudeStr);

                        // Check conditions
                        if ("female".equalsIgnoreCase(gender) &&
                            ("China".equalsIgnoreCase(country) ||
"Russia".equalsIgnoreCase(country)) &&
                            (latitude >= 40 && latitude <= 46)) {
                            result.add(key);
                            System.out.println("Accepted key: " + key +
                                " (Gender: " + gender + ", Country: " + country +
", Latitude: " + latitude + ")");
                            acceptedCount++;
                        } else {

```

```

        rejectedCount++;
    }
    } else {
        rejectedCount++;
    }
    } catch (NumberFormatException e) {
        rejectedCount++;
    } catch (NullPointerException e) {
        rejectedCount++;
    }
    }
} while (!cursor.equals("0"));

System.out.println("Query complete. Found " + result.size() + " matching
users.");
System.out.println("Accepted keys: " + acceptedCount + ", Rejected keys:
" + rejectedCount);

return result;
}

public void query5() {
    if (!jedis.exists("leaderboard:2")) {
        System.out.println("Leaderboard data does not exist.");
        return;
    }
    Set<String> topPlayers = jedis.zrevrange("leaderboard:2", 0, 9);
    if (topPlayers.isEmpty()) {
        System.out.println("No players found in leaderboard:2.");
        return;
    }
    try (var pipeline = jedis.pipelined()) {
        List<Response<Double>> scores = new ArrayList<>();
        List<Response<String>> emails = new ArrayList<>();

        for (String playerName : topPlayers) {
            scores.add(pipeline.zscore("leaderboard:2", playerName));
            emails.add(pipeline.hget(playerName, "email"));
        }
        pipeline.sync();
        int rank = 1;
        Iterator<String> playerIterator = topPlayers.iterator();
        for (int i = 0; i < topPlayers.size(); i++) {
            String playerName = playerIterator.next();
            Double score = scores.get(i).get();
            String email = emails.get(i).get();

            System.out.println("Rank: " + rank + " | Player: " + playerName +
" | Score: " + score + " | Email: " + email);
            rank++;
        }
    }
}
}

```

```

public void clearRedis() {
    jedis.flushAll();
    System.out.println("All Redis tables have been cleared.");
}

public void reinitializeRedis() {
    clearRedis();
    System.out.println("Redis has been cleared and reinitialized.");
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    RedisClient client = new RedisClient();

    boolean running = true;
    while (running) {
        System.out.println("\n===== Redis Menu =====");
        System.out.println("1. Load Users from file");
        System.out.println("2. Load Scores from file");
        System.out.println("3. Query 1 (Get user info)");
        System.out.println("4. Query 2 (Get latitude and longitude)");
        System.out.println("5. Query 3 (Get even user IDs)");
        System.out.println("6. Query 4 (Get female users from China/Russia with specific latitude)");
        System.out.println("7. Query 5 (Show top players and scores)");
        System.out.println("8. Clear all Redis tables");
        System.out.println("9. Reinitialize Redis");
        System.out.println("10. Exit");
        System.out.print("Enter your choice: ");

        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                System.out.print("Enter the file path for users: ");
                String userFile = scanner.nextLine();
                client.loadUsers(userFile);
                break;
            case 2:
                System.out.print("Enter the file path for scores: ");
                String scoreFile = scanner.nextLine();
                client.loadScores(scoreFile);
                break;
            case 3:
                System.out.print("Enter user ID: ");
                String userId1 = scanner.nextLine();
                System.out.println(client.query1(userId1));
                break;
            case 4:
                System.out.print("Enter user ID: ");
                String userId2 = scanner.nextLine();
                System.out.println(client.query2(userId2));
                break;
        }
    }
}

```

```

        case 5:
            System.out.println(client.query3());
            break;
        case 6:
            System.out.println(client.query4());
            break;
        case 7:
            client.query5();
            break;
        case 8:
            client.clearRedis();
            break;
        case 9:
            client.reinitializeRedis();
            break;
        case 10:
            System.out.println("Exiting...");
            running = false;
            break;
        default:
            System.out.println("Invalid choice, please try again.");
    }
}
scanner.close();
}
}

class User {
    private String userId;
    private double score;
    private int leaderboard;

    public User(String userId, double score, int leaderboard) {
        this.userId = userId;
        this.score = score;
        this.leaderboard = leaderboard;
    }

    public String getUserId() {
        return userId;
    }

    public double getScore() {
        return score;
    }

    public int getLeaderboard() {
        return leaderboard;
    }

    @Override
    public String toString() {
        return "User{id='" + userId + "', score=" + score + ", leaderboard=" +
        leaderboard + "'}";
    }
}

```

```
}
```

Some Additional Outputs

```
C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient
^C
D:\iitj\tri-3\BDM\iitj-bdm-assignment\Assignment - 5>java -cp libs\*. RedisClient
Connected to Redis with authentication.

===== Redis Menu =====
1. Load Users from file
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 8
All Redis tables have been cleared.
```

```
C:\Windows\system32\cmd.exe - java -cp libs\*. RedisClient
Enter your choice: 19
Invalid choice, please try again.

===== Redis Menu =====
1. Load Users from file
2. Load Scores from file
3. Query 1 (Get user info)
4. Query 2 (Get latitude and longitude)
5. Query 3 (Get even user IDs)
6. Query 4 (Get female users from China/Russia with specific latitude)
7. Query 5 (Show top players and scores)
8. Clear all Redis tables
9. Reinitialize Redis
10. Exit
Enter your choice: 9
All Redis tables have been cleared.
Redis has been cleared and reinitialized.
```