**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 = "OahVB0ixFG49ILZK5qvC6QG6BmujVgHC"; // Replace with your Redis password

            jedis.auth(password);

            System.out.println("Connected to Redis with authentication.");

        } catch (JedisException e) {

            e.printStackTrace();

        }

    }

**Output:**

A computer screen with text on it

Description automatically generated

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

1. 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) {

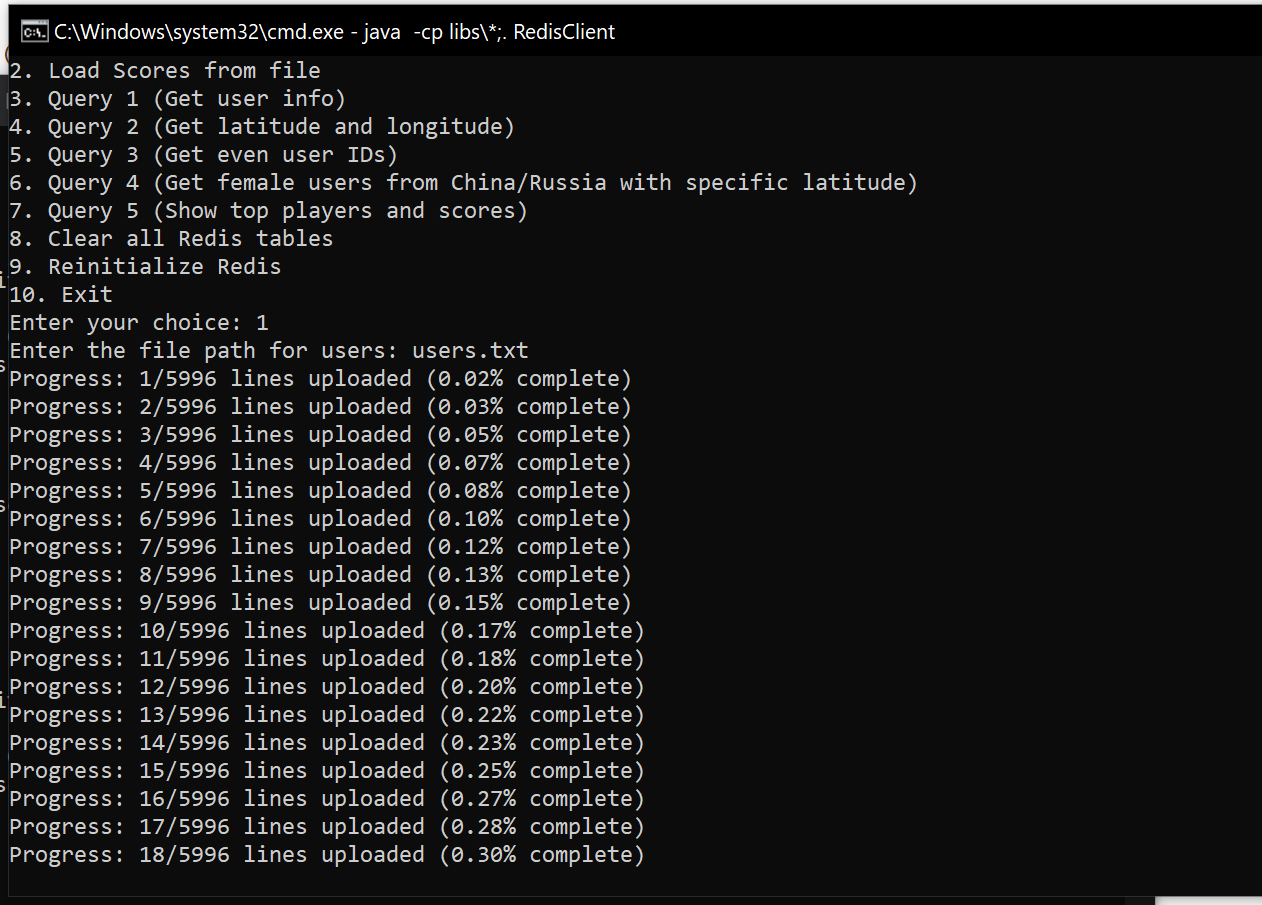
            e.printStackTrace();

        }

    }

Output:

**User.txt**



A screen shot of a computer

Description automatically generated

**UserScores.csv**

A screenshot of a computer

Description automatically generated

1. 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:**

A computer screen with white text

Description automatically generated

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

public String query2(String userId) {

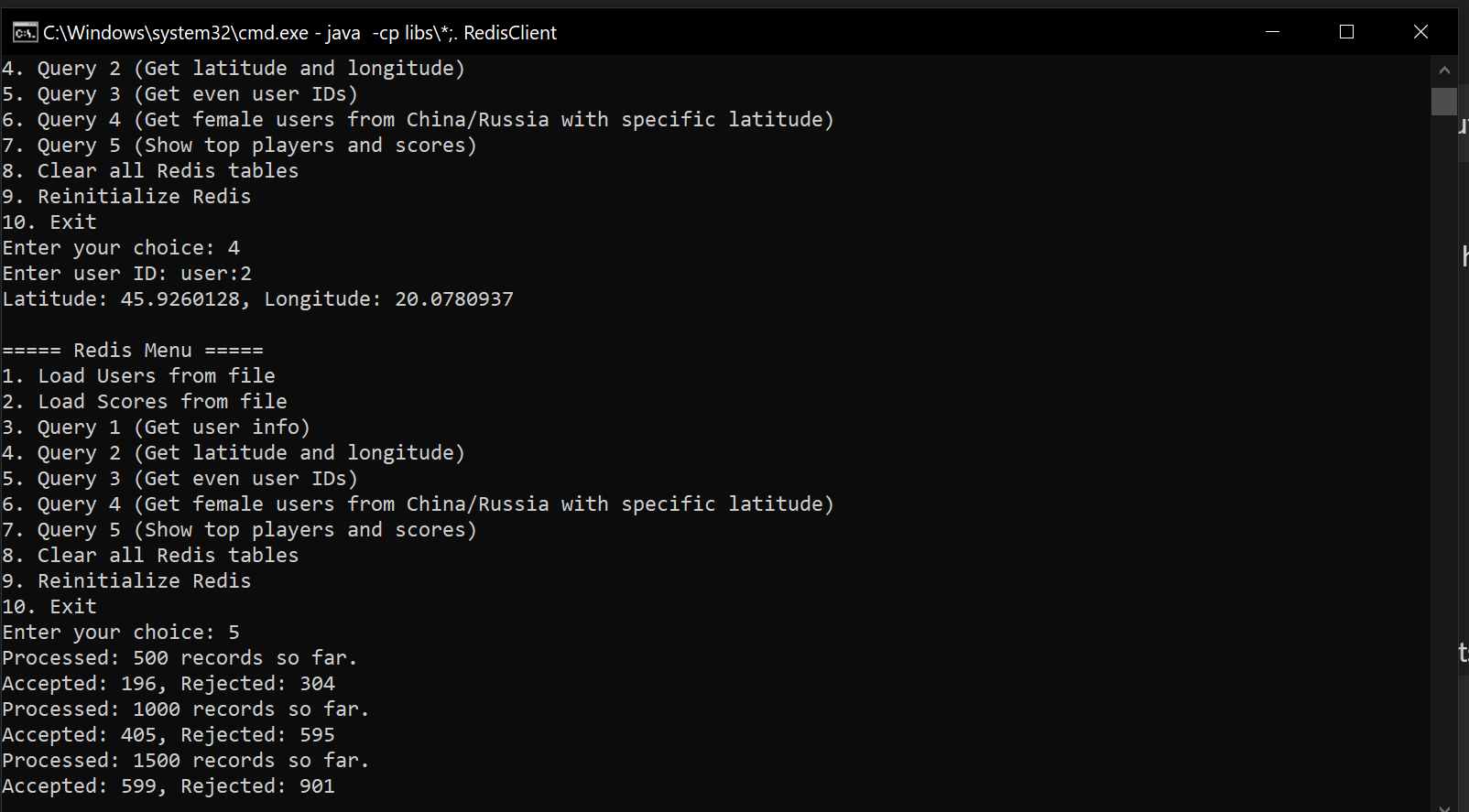
        String latitude = jedis.hget(userId, "latitude");

        String longitude = jedis.hget(userId, "longitude");

        return "Latitude: " + latitude + ", Longitude: " + longitude;

    }

**Output:**

****

1. 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

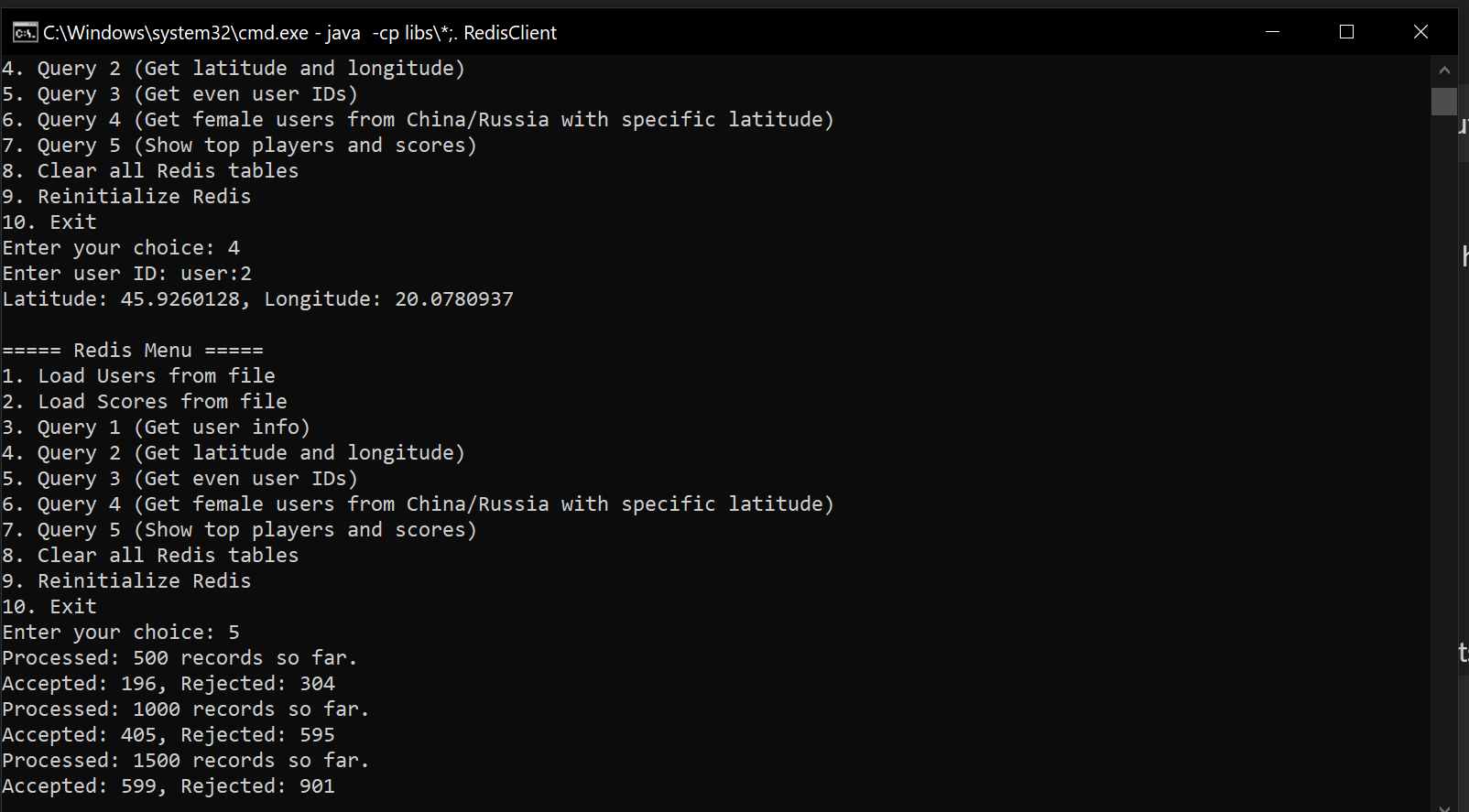
        System.out.println("Query complete.");

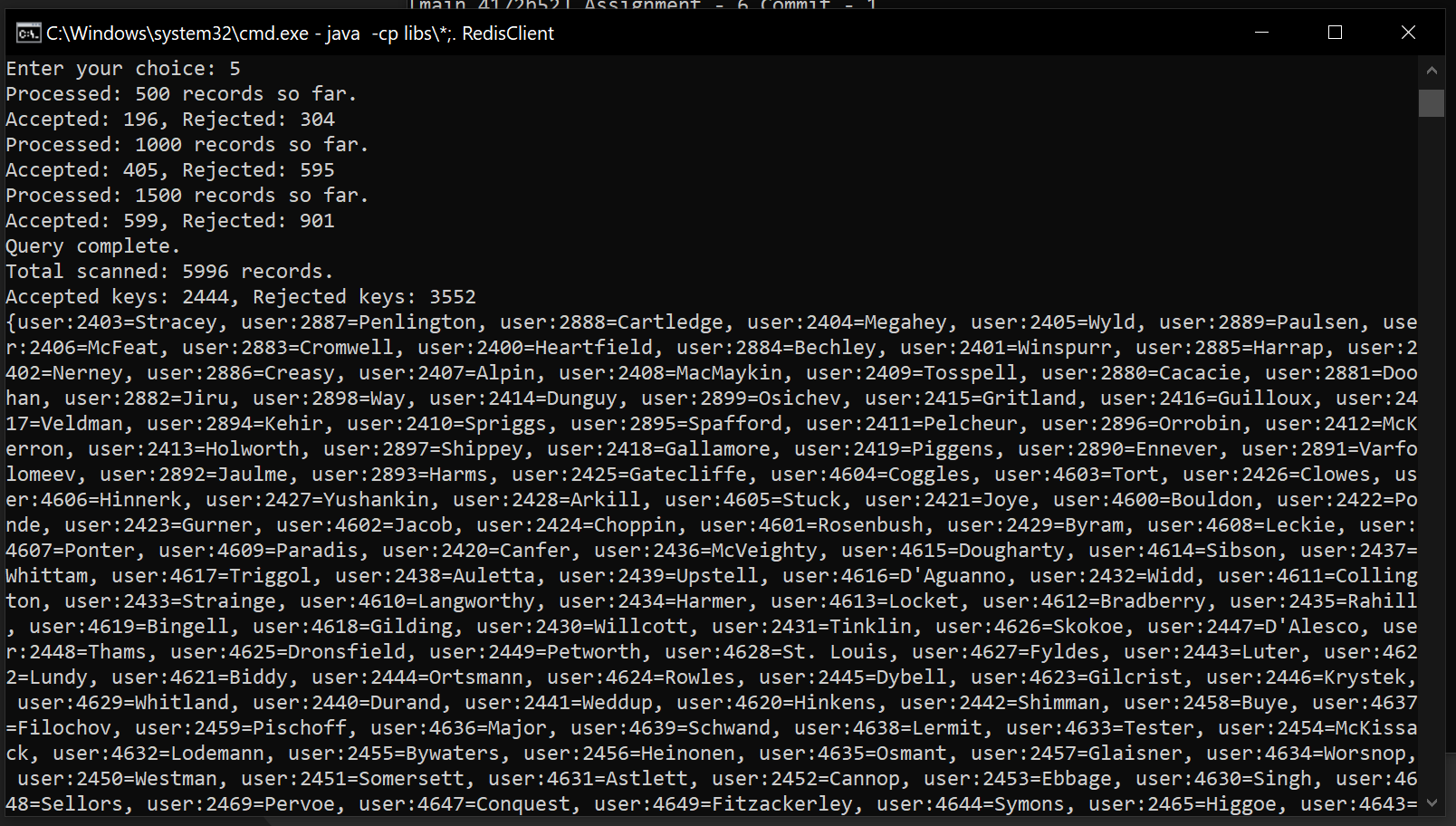
        System.out.println("Total scanned: " + totalScanned + " records.");

        System.out.println("Accepted keys: " + acceptedCount + ", Rejected keys: " + rejectedCount);

        return result;

    }

****



1. 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:**

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. 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:**

**A screen shot of a computer

Description automatically generated**

**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**

**A computer screen shot of a black screen

Description automatically generated**

**A screen shot of a computer

Description automatically generated**