# CS209A Project Report

## Stack Overflow Q&A Data w.r.t. Java Programming

**Student Name and Student ID**

丁健乐 12111517

林俊杰 12111511

## 1 Contributions

| Member | Contributions | Ratio |
|---|---|---|
| 丁健乐 | 1. Data getting and storage | 50% |
| 林俊杰 | 1. Data analysis and visualization | 50% |

## 2 Problem Analysis

Number of answers (15 points)

1. Percentage of unanswered questions
2. For each question, the mean and maximum values of the answers
3. Distribution of the number of answers for each question

Acceptable answer (15 points)

1. Percentage of questions with acceptable answers
2. Distribution of problem resolution times (problem release times -- release times with acceptable answers)
3. The percentage of questions in which an answer that was not accepted received more likes than an answer that was acceptable

Label (15 points)

1. Tags that often appear with java tags
2. The hashtag or hashtag combination that gets the most likes

3. The TAB or combination of tabs that get the most views

Users (15 points)

1. The distribution of the total number of different users Posting questions, answering questions, or commenting on answers, per question
2. The most active users who regularly participate in discussions

## 3 The Description of the Collected Data

First of all, we obtained 500 basic information from https://api.stackexchange.com/2.3/questions, Including owner_id, question_id, is_answered answer_numbers, question_posting_time, tags, upvote, views, user_count, comment. At the same time, we also store every user who asks a question. Then we added from https://api.stackexchange.com/2.3/questions/ {ids} / answers to get accepted_answer_id, answer_posting_time,most_upvote_answer_id. We also store every user who answers the question

Finally, we go through https://api.stackexchange.com/2.3/questions/ {ids} / comments added to every question the user comments. So that's the combination of the data that we got.

The SQL statements below are the tables we built to store the answer.

```sql
CREATE TABLE question
(
    owner_id              VARCHAR,
    question_id           INTEGER PRIMARY KEY,
    is_answered           BOOLEAN,
    answer_numbers        INTEGER,
    accepted_answer_id    INTEGER,
    question_posting_time INTEGER,
    answer_posting_time   INTEGER,
    most_upvote_answer_id INTEGER,
    tags                  VARCHAR,
    upvote                INTEGER,
    views                 INTEGER,
    user_count            INTEGER,
    comment               INTEGER
);

CREATE TABLE owner
(
    owner_id         VARCHAR PRIMARY KEY,
    question_numbers INTEGER,
    answer_numbers   INTEGER,
    comment_numbers  INTEGER
);
```

## 4 The Architecture Design of Project

In terms of getting the data, I use Java's HttpURLConnection to send the HTTP request and process the JSON response data returned from the request. A BufferedReader in is then created to read the connected input stream. Determine whether to use GZIP decompression based on the content encoding in the response header. The code uses a BufferedReader to read the data line by line, appending each line to the response until the data read is empty. Next, the code converts the JSON data in response to the JSONObject object 'json'. Finally, we loop through each element of the items array to get the corresponding JSONObject object 'item' for each element.

```java
URL url = new URL(apiUrl + "?page=" + page + "&pagesize=" + pageSize +
apiParams);
HttpURLConnection connection1 = (HttpURLConnection) url.openConnection();
connection1.setRequestMethod("GET");
BufferedReader in;
if ("gzip".equals(connection1.getContentEncoding())) {
    in = new BufferedReader(
        new InputStreamReader(new
GZIPInputStream(connection1.getInputStream())));
} else {
    in = new BufferedReader(new
InputStreamReader(connection1.getInputStream()));
}
StringBuilder response = new StringBuilder();
String line;
while ((line = in.readLine()) != null) {
    response.append(line);
}
in.close();
JSONObject json = new JSONObject(response.toString());
JSONArray items = json.getJSONArray("items");
```

In terms of storing data, after I get the data from the website, I select the data I need and save it in the postgresql database. I choose data insertion as an example:

```java
private static void insertData(Connection connection, String ownerId, int
questionId,
    boolean isAnswered, int answerCount,
    int acceptedAnswerId, int questionPostingTime, int answerPostingTime,
    int most_upvote_answer_id, String tags,
    int upvote, int views, int userCount, int comments) throws SQLException, SQLException {
    String insertQuery = "INSERT INTO question (owner_id,question_id,is_answered," +
        "answer_numbers,accepted_answer_id,question_posting_time,answer_posting_time," +
        "most_upvote_answer_id,tags,upvote,views,user_count,comment) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?)";
    PreparedStatement statement = connection.prepareStatement(insertQuery);
    statement.setString(1, ownerId);
    statement.setInt(2, questionId);
    statement.setBoolean(3, isAnswered);
    statement.setInt(4, answerCount);
    statement.setInt(5, acceptedAnswerId);
    statement.setInt(6, questionPostingTime);
    statement.setInt(7, answerPostingTime);
    statement.setInt(8, most_upvote_answer_id);
    statement.setString(9, tags);
    statement.setInt(10, upvote);
    statement.setInt(11, views);
    statement.setInt(12, userCount);
    statement.setInt(13, comments);
    statement.executeUpdate();
    statement.close();
}
```

I use PreparedStatement to store the insertQuery, and add the required part into the statement and execute the statement.

About data analysis, we use SQL statements to search for the required data from the database, and store it into JavaScript files and Sync to visualization side. The code above is an example of the whole process:

```java
public static void ParticipationDistribution(String str) throws SQLException {
    String filePath = "src/main/resources/static/js/Users/ParticipationDistribut

    Statement stmt = con.createStatement();
    ResultSet rs0 = stmt.executeQuery("select count(*)\n"
        + "from (select (answer_numbers + comment + 1) as sum\n"
        + "       from question) l\n"
        + "where sum >= 1\n"
        + "  and sum <= 10;");
    rs0.next();
    int first = rs0.getInt(1);

    ResultSet rs1 = stmt.executeQuery("select count(*)\n"
        + "from (select (answer_numbers + comment + 1) as sum\n"
        + "      from question) l\n"
        + "where sum > 10\n"
        + "  and sum <= 20;");
    rs1.next();
    int second = Math.round(rs1.getFloat(1));

    ResultSet rs2 = stmt.executeQuery("select count(*)\n"
        + "from (select (answer_numbers + comment + 1) as sum\n"
        + "      from question) l\n"
        + "where sum > 20\n"
        + "  and sum <= 30;");
    rs2.next();
    int third = Math.round(rs2.getFloat(1));

    ResultSet rs3 = stmt.executeQuery("select count(*)\n"
        + "from (select (answer_numbers + comment + 1) as sum\n"
        + "      from question) l\n"
        + "where sum > 30\n"
        + "  and sum <= 40;");
    rs3.next();
    int fourth = Math.round(rs3.getFloat(1));

    ResultSet rs4 = stmt.executeQuery("select count(*)\n"
        + "from (select (answer_numbers + comment + 1) as sum\n"
        + "      from question) l\n"
        + "where sum > 40;");
    rs4.next();
    int fifth = Math.round(rs4.getFloat(1));

    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
        StringBuilder content = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            if (line.contains("let " + str + " =")) {
                Map<String, Integer> values = extractValues(line);
                if (values.containsKey(">= 1 & <= 10")) {
                    values.put(">= 1 & <= 10", first);
                }
                if (values.containsKey("> 10 & <= 20")) {
                    values.put("> 10 & <= 20", second);
                }
                if (values.containsKey("> 20 & <= 30")) {
                    values.put("> 20 & <= 30", third);
                }
                if (values.containsKey("> 30 & <= 40")) {
                    values.put("> 30 & <= 40", fourth);
                }
                if (values.containsKey("> 40")) {
                    values.put("> 40", fifth);
                }
                String modifiedLine = generateModifiedLine(str, values);
                line = line.replaceFirst("let " + str + " =.*", modifiedLine);
            }
            content.append(line).append("\n");
        }
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath)
            writer.write(content.toString());
        }
        System.out.println("JavaScript file modified successfully.");
    } catch (IOException e) {
        System.out.println(
            "An error occurred while modifying the JavaScript file: " + e.getMes
    }
}
```

## 5 The insights from the Data Analysis Results

After conducting the data analysis on Java programming, several insights have emerged. Here are some of the key findings:

1. What percentage of questions don't have any answer?

   Almost two-thirds of the questions asked were not answered.

   "No answer": 332, "Has answer": 168

2. What is the average and maximum number of answers?

   The maximum number of answers to answered questions was 40 and the mean was 1.832.

   "Maximum": 40, "Average": 2

3. What is the distribution of the number of answers?

   ">= 4 & <= 6": 27, ">= 10": 19, ">= 1 & <= 3": 238, "= 0": 212, ">= 7 & <= 9": 4

4. What percentage of questions have accepted answers (one question could only have one accepted answer)?

   "Has accepted answer": 76, "No accepted answer": 424

5. What is the distribution of question resolution time (i.e., the duration between the question posting time and the posting time of the accepted answer)?

   ">= 0, <= 1000": 15, "> 100000": 17, "> 1000, <= 10000": 24, "> 10000, <= 100000": 20

   The unit of time is seconds.

6. What percentage of questions have non-accepted answers (i.e., answers that are not marked as accepted) that have received more upvotes than the accepted answers?

   "More Upvotes Non-accepted Answer": 26, "Less Upvotes Non-accepted Answer": 50

7. Which tags frequently appear together with the java tag?

   Most Frequently Asked Topics: The analysis revealed that certain topics of Java programming are asked more frequently than others. The top five most asked topics are:

   A. Spring-boot  B. Spring  C. Android  D. Spring-boot-actuator  E. Maven

8. Which tags or tag combinations receive the most upvotes?

   A. List  B. Jar  C. Arraylist  D. Java-stream  E. Java8

9. Which tags or tag combinations receive the most views?

   A. Spring-boot  B. Spring  C. Android  D. Spring-boot-actuator  E. Maven

10. Many users could participate in a thread discussion. What is the distribution of such participation (i.e., the number of distinct users who post the question, answers, or comments in a thread)?

    "> 40": 1, "> 20 & <= 30": 5, "> 10 & <= 20": 33, ">= 1 & <= 10": 457, "> 30 & <= 40": 4

11. Which are the most active users who frequently participate in thread discussions?

    "6140624", "28659413", "7079914", "19118721", "28657233"...

## 6 Code

https://github.com/isLinHehea/CS209A_Project