

BILKENT UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING



CS353

DATABASE SYSTEMS

**TECHNICAL INTERVIEW AND CODING PLATFORM
FINAL REPORT**

Ahmet Ayrancıoğlu

21601206

Deniz Dalkılıç

21601896

Kaan Gönç

21602670

Description of the Application	3
Final E/R Diagram	4
Final List of Tables	5
Implementation Details	6
Sample Reports	7
User's Manual	8
Login/Signup Type	8
Login	8
Sign In	9
Problems	9
Solving A Problem	10
Looking At The Solution	10
Looking At Past Submissions	11
Looking At A Past Submissions	11
Questions	12
Answering a Question	12
Contests	13
Join A Contest	13
Interviews	14
Conducting An Interview	14
Coding Bets	15
Past Interviews	15
Code Produced	16

1. Description of the Application

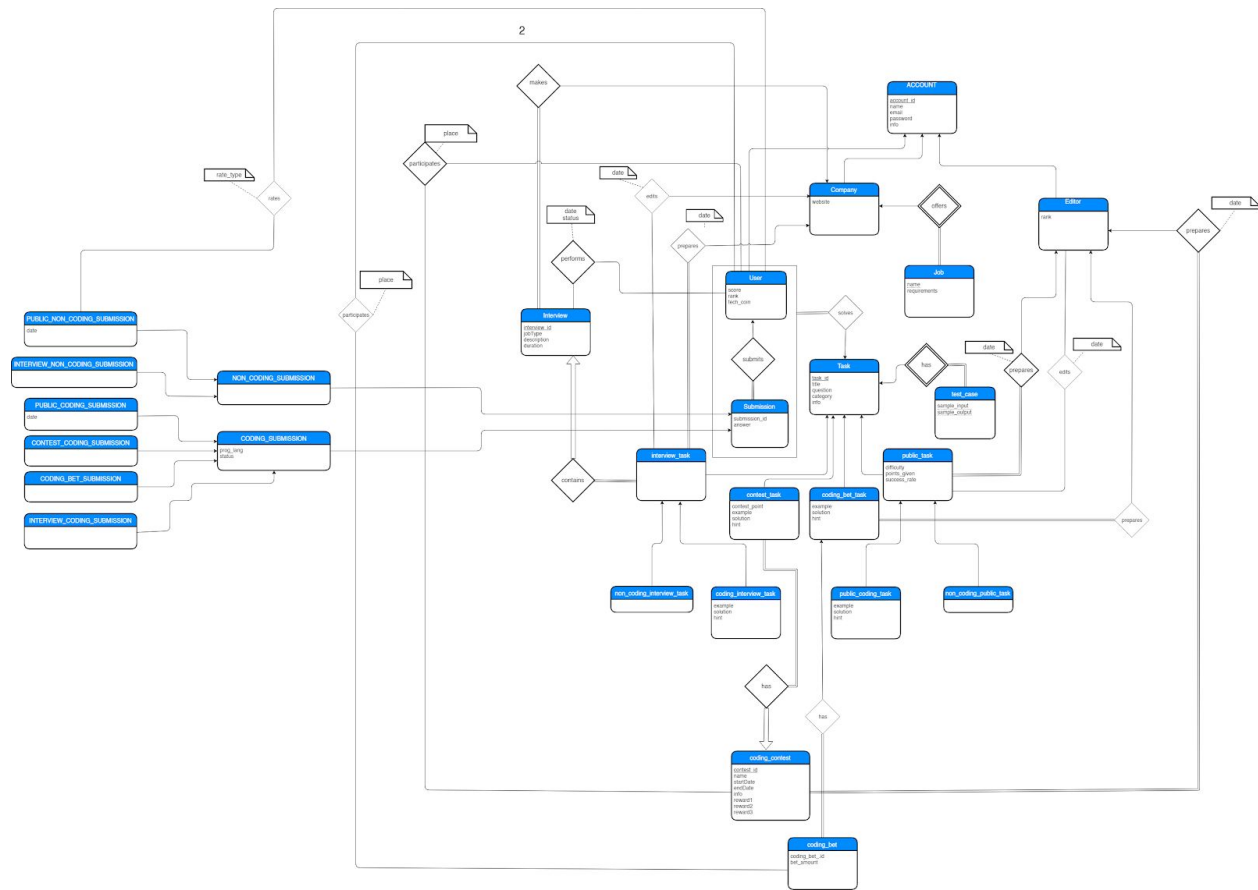
Online Technical Interview Preparation and Coding Platform Database System is a web platform, intended to be used by potential employees and employing companies, as well as editors who post content for other users. Essentially, TechCode is a site with a variety of tool focused on programming interview questions. One of the main parts of the site is dedicated to a list of problems. Each problem is tagged with a different computer science term indicating the nature of the problem such as "object-oriented programming", "string", "operating systems" and so forth. Another part of the site is a series of coding competitions.

A user can solve different tasks that are either created by an editor or by a company. These tasks can be categorized into two, public tasks that are available for everyone to complete, and interview tasks that are created by companies to find potential employees and test their programming skills. Public tasks can be found on the main page of TechCode. However, interview tasks are only available for users that have an ongoing interview.

These tasks can also be categories in a different way, coding, and non-coding. Coding tasks involve coding questions, while non-coding questions involve general interview questions and other non-coding technical questions. The user can also view their task and interview result from TechCode without needing any other external tools and websites. Addition to all of this as mentioned before, there is also coding contests that everyone can participate in. These events happen at different times and are open for application for a limited amount of time. In these contests, various users compete with each other to solve the same coding tasks, and the winner gets points that show they were better than other participants and possibly attract the attention of possible employers.

Finally, there are coding bets that two people can participate in. A coding bet is created by a user and posted publicly with everyone. The user that creates the bet put a wager on the bet. When another user accepts the open bet matching the wager, the two users start to compete with each other to solve a randomly assigned coding problem. The user that correctly answers the coding problem first wins the wager that was placed on the bet.

2. Final E/R Diagram



3. Final List of Tables

ACCOUNT

Relational Model:

ACCOUNT(account_id, name, email, password, info)

Foreign Keys:

None

USER

Relational Model:

USER(user_id, score, tech_coin)

Foreign Keys:

user_id references ACCOUNT.id

EDITOR

Relational Model:

EDITOR(editor_id, rank)

Foreign Keys:

editor_id references ACCOUNT.id

COMPANY

Relational Model:

COMPANY(company_id, website)

Foreign Keys:

company_id references ACCOUNT.id

JOB

Relational Model:

JOB(company_id, name, requirements)

Foreign Keys:

company_id references COMPANY.company_id

INTERVIEW

Relational Model:

INTERVIEW(interview_id, job_type, description, duration)

Foreign Keys:

None

CODING_CONTEST

Relational Model:

CODING_CONTEST(contest_id, name, start_date, end_date, reward1, reward2, reward3)

Foreign Keys:

None

CODING_BET

Relational Model:

CODING_BET(coding_bet_id, bet_amount, status, owner_id)

Foreign Keys:

None

TASK

Relational Model:

TASK(task_id, title, question, category, info)

Foreign Keys:

None

PUBLIC_TASK

Relational Model:

PUBLIC_TASK(task_id, difficulty, points_given, success_rate)

Foreign Keys:

task_id references TASK.task_id

PUBLIC_CODING_TASK

Relational Model:

PUBLIC_CODING_TASK(task_id, example, solution, hint)

Foreign Keys:

task_id references PUBLIC_TASK.id

PUBLIC_NON_CODING_TASK

Relational Model:

PUBLIC_NON_CODING_TASK(task_id)

Foreign Keys:

task_id references PUBLIC_TASK.task_id

INTERVIEW_TASK

Relational Model:

INTERVIEW_TASK(task_id, interview_id)

Foreign Keys:

task_id references TASK.task_id

Interview_id references INTERVIEW.interview_id

NON_CODING_INTERVIEW_TASK

Relational Model:

NON_CODING_INTERVIEW_TASK(task_id)

Foreign Keys:

task_id references INTERVIEW_TASK.task_id

CODING_INTERVIEW_TASK

Relational Model:

CODING_INTERVIEW_TASK(task_id, example, solution, hint)

Foreign Keys:

task_id references INTREVIEW_TASK.task_id

CONTEST_TASK

Relational Model:

CONTEST_TASK(task_id, contest_point, example, solution, hint, contest_id)

Foreign Keys:

task_id references TASK.task_id

contest_id references CODING_CONTEST.contest_id

CODING_BET_TASK

Relational Model:

CODING_BET_TASK(task_id, example, solution, hint)

Foreign Keys:

task_id references TASK.task_id

SUBMISSION

Relational Model:

SUBMISSION(submission_id, user_id, task_id, answer)

Foreign Keys:

user_id references USER.user_id

task_id references TASK.task_id

NON_CODING_SUBMISSION

Relational Model:

NON_CODING_SUBMISSION(submission_id)

Foreign Keys:

submission_id references SUBMISSION.submission_id

CODING_SUBMISSION

Relational Model:

CODING_SUBMISSION(submission_id, prog_lang, status)

Foreign Keys:

submission_id references SUBMISSION.submission_id

PUBLIC_NON_CODING_SUBMISSION

Relational Model:

PUBLIC_NON_CODING_SUBMISSION(submission_id, date)

Foreign Keys:

submission_id references NON_CODING_SUBMISSION.submission_id

INTERVIEW_NON_CODING_SUBMISSION

Relational Model:

INTERVIEW_NON_CODING_SUBMISSION(submission_id)

Foreign Keys:

submission_id references NON_CODING_SUBMISSION.submission_id

INTERVIEW_CODING_SUBMISSION

Relational Model:

INTERVIEW_CODING_SUBMISSION(submission_id)

Foreign Keys:

submission_id references INTERVIEW_CODING_SUBMISSION.submission_id

PUBLIC_CODING_SUBMISSION

Relational Model:

PUBLIC_CODING_SUBMISSION(submission_id, date)

Foreign Keys:

submission_id references CODING_SUBMISSION.submission_id

CONTEST_CODING_SUBMISSION

Relational Model:

CONTEST_CODING_SUBMISSION(submission_id)

Foreign Keys:

submission_id references CODING_SUBMISSION.submission_id

CODING_BET_SUBMISSION

Relational Model:

CODING_BET_SUBMISSION(submission_id)

Foreign Keys:

submission_id references CODING_SUBMISSION.submission_id

COMPANY_INTERVIEW_TASK_PREPARE

Relational Model:

COMPANY_INTERVIEW_TASK_PREPARE(company_id, task_id, date)

Foreign Keys:

company_id references COMPANY.company_id

task_id references INTERVIEW_TASK.task_id

EDITOR_PUBLIC_TASK_PREPARE

Relational Model:

EDITOR_PUBLIC_TASK_PREPARE(editor_id, task_id, date)

Foreign Keys:

editor_id references EDITOR.editor_id

task_id references PUBLIC_TASK.task_id

EDITOR_CODING_CONTEST_PREPARE

Relational Model:

EDITOR_CODING_CONTEST_PREPARE(editor_id, contest_id, date)

Foreign Keys:

editor_id references EDITOR.editor_id

contest_id references CODING_CONTEST.contest_id

COMPANY_INTERVIEW_PREPARE

Relational Model:

COMPANY_INTERVIEW_RELATION(company_id, interview_id)

Foreign Keys:

company_id references COMPANY.company_id

interview_id references INTERVIEW.interview_id

USER_CONTEST_PARTICIPATE

Relational Model:

USER_CONTEST_PARTICIPATE(user_id, contest_id, place)

Foreign Keys:

user_id references USER.user_id

contest_id references CODING_CONTEST.contest_id

USER_CODING_BET_PARTICIPATE

Relational Model:

USER_CONTEST_PARTICIPATE(user_id, coding_bet_id, place)

Foreign Keys:

user_id references USER.user_id

coding_bet_id references CODING_BET.coding_bet_id

USER_INTERVIEW_PERFORM

Relational Model:

USER_CONTEST_PARTICIPATE(user_id, interview_id, date, result)

Foreign Keys:

user_id references USER.user_id

interview_id references INTERVIEW.interview_id

TEST_CASE

Relational Model:

TEST_CASE(task_id, sample_input, sample_output)

Foreign Keys:

task_id references TASK.task_id

CODING_BET_TASK_RELATION

Relational Model:

CODING_BET_TASK_RELATION(coding_bet_id, task_id)

Foreign Keys:

Coding_bet_id references CODING_BET.coding_bet_id

task_id references CODING_BET_TASK.task_id

RATE_ANSWER

Relational Model:

COMPANY_INTERVIEW_TASK_PREPARE(user_id, answer_id, rate_type)

Foreign Keys:

user_id references USER.user_id

answer_id references PUBLIC_NON_CODING_SUBMISSION.submission_id

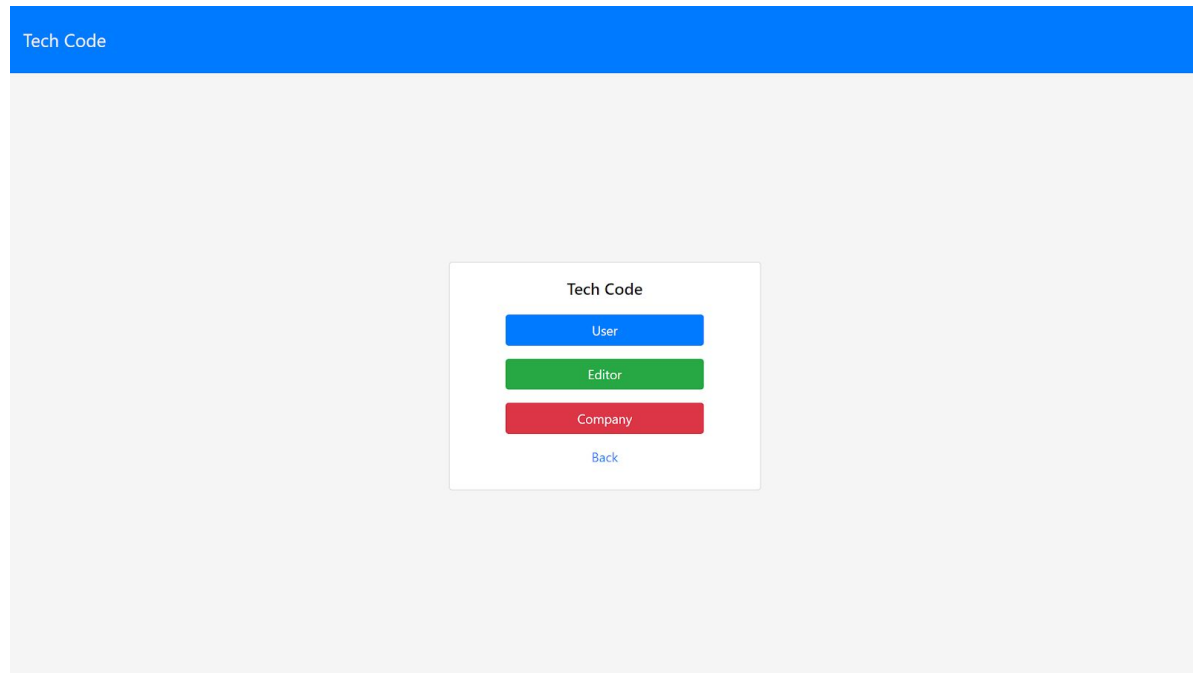
4. Implementation Details

In this project, PHP was used to develop the backend of the application. Accessing the database was done using PHP with the help of MySQL Improved Extension. In some of the pages, JavaScript was used to help with the internal logic of the page. For front-end development, HTML was used. The main layout of the pages was done using CSS grids. To improve the quality of the UI, Bootstrap 4 components was used. MySQL was used to create the database and for data retrieval and modification. MariaDB was used to allow referencing of tables. Java was used to create tables, reports, procedures and to add mock-up data to the database to help with testing.

To access our database, we created an SQL controller class where we stored our queries in logical functions so we could quickly access them in our pages. We also had a SQL displayer class where the results from the SQL controller were formatted with HTML tags to be correctly displayed in our pages.

5. User's Manual

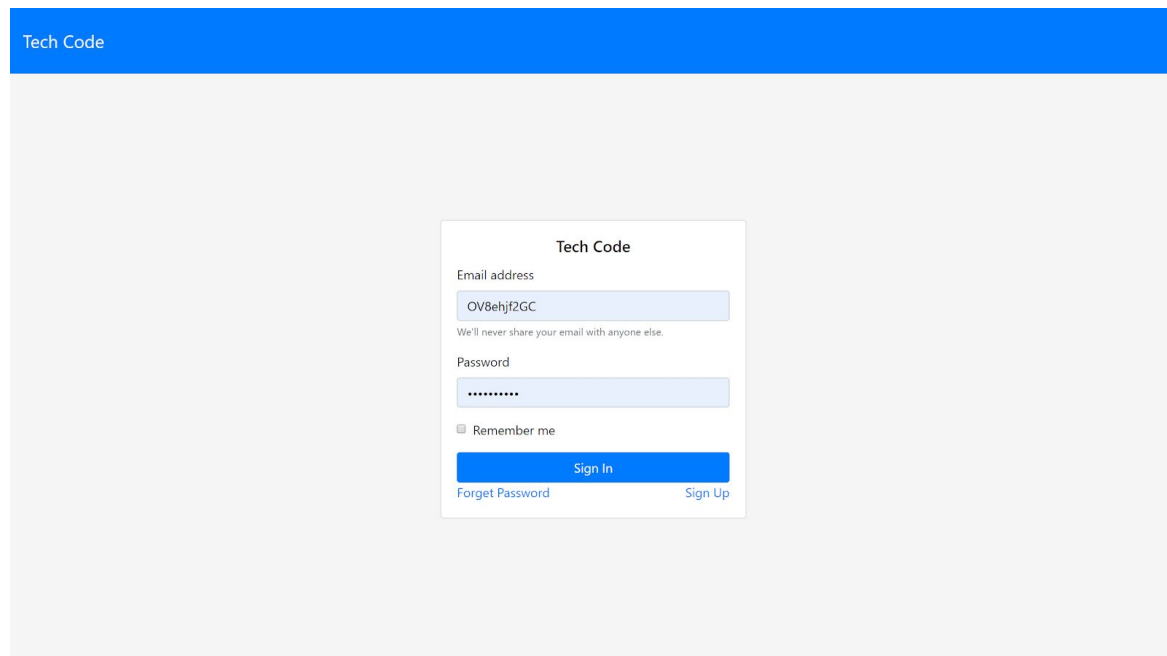
Login/Signup Type



The screenshot shows a web page with a blue header bar containing the text "Tech Code". The main content area is light gray and contains a white modal box titled "Tech Code". Inside the modal, there are three colored buttons stacked vertically: a blue button labeled "User", a green button labeled "Editor", and a red button labeled "Company". Below these buttons is a blue link labeled "Back".

This page is the first page on the website. User can choose the type of account they want to login/signup with.

Login



The screenshot shows a web page with a blue header bar containing the text "Tech Code". The main content area is light gray and contains a white modal box titled "Tech Code". Inside the modal, there is a form with the following elements: a label "Email address" above a text input field containing "OV8ehjf2GC"; a small text note "We'll never share your email with anyone else."; a label "Password" above a password input field with masked characters "*****"; a checkbox labeled "Remember me"; a blue button labeled "Sign In"; a link labeled "Forgot Password" below the "Sign In" button; and a link labeled "Sign Up" to the right of the "Sign In" button.

After choosing the type of account the user has, they can log in with their emails.

Sign In

Tech Code

Tech Code

Email address

OV8ehjf2GC

Name

Enter Name

Info

Enter Info

Password

Confirm Password

Confirm Password

Continue

Back

After choosing the type of account, the user can sign up with their emails.

Problems

Tech Code Problems Questions Contests Interviews Coding Bets Profile

Category

All Data Strings Algorithms Databases Operating Systems OOP

Search Search

#	Title	Solution	Success Rate	Difficulty
100	Two Sum	***	85	Easy
101	Add Two Numbers	***	68	Medium
102	Longest Substring Without Repeating Characters	***	77	Medium
103	Median of Two Sorted Arrays	***	50	Hard
104	Longest Palindromic Substring	***	30	Medium

Your Progress

20.00%

4
Todo

1/5
Solved

1
Attempted

100 Gold

This page is the main page of the website. The user can choose a coding problem to try and solve. The users can filter problems with categories or they can search for keywords. In addition to that, the user can see their progress on the right.

Solving A Problem

The screenshot shows the 'Solving A Problem' interface. The top navigation bar is blue with links for 'Tech Code', 'Problems', 'Questions', 'Contests', 'Interviews', 'Coding Bets', and a 'Profile' button. Below the navigation bar, there are tabs for 'Description', 'Solution', and 'Submissions'. The 'Description' tab is active, showing the problem details for '1.Two Sum'. The problem is marked as 'Easy'. The description states: 'Given an array of integers, return indices of the two numbers such that they add up to a specific target. You may assume that each input would have exactly one solution, and you may not use the same element twice.' An example is provided: 'Given nums = [2, 7, 11, 15], target = 9, Because nums[0] + nums[1] = 2 + 7 = 9, return [0, 1]. Accepted 1 Submissions 4'. On the right side, there is a code editor with a language dropdown set to 'C++'. The code editor contains a partial solution:

```
class Solution {
public:
    vector< int > twoSum(vector< int > &nums, int target) {
    }
};
```

 Below the code editor is a 'Test' section. At the bottom, there are buttons for 'Problems', 'Run Code', and 'Submit'.

After choosing a problem from the problems page, the user can try to solve the problem. The description and details of the problem are displayed on the left and the users can write their code on the right. The test cases are shown at the bottom right. The user can submit their solution by clicking the submit button.

Looking At The Solution

The screenshot shows the 'Looking At The Solution' interface. The top navigation bar is blue with links for 'Tech Code', 'Problems', 'Questions', 'Contests', 'Interviews', 'Coding Bets', and a 'Profile' button. Below the navigation bar, there are tabs for 'Description', 'Solution', and 'Submissions'. The 'Solution' tab is active, showing the solution for '1.Two Sum'. The problem is marked as 'Easy'. The description is the same as in the previous screenshot. The solution is displayed in the code editor:

```
class Solution { public String solution(int solution) { //solution } }
```

 At the bottom, there is a 'Problems' button.

The users can look at the solution of the problem if they fail to solve it by themselves by clicking the solution button.

Looking At Past Submissions

Time Submitted	Status	Language	Code
2019-02-01 00:00:00	solved	C	***
2019-03-03 00:00:00	solved	C++	***
2018-08-11 00:00:00	failed	JAVA	***
2019-07-10 00:00:00	solved	PYTHON	***

The users can see their previous attempts on the problem, by clicking the Submissions button. The code they are writing remains on the right side and the previous submissions are shown on the left. The users can click on their submissions to see further details.

Looking At A Past Submissions

Submission Detail

Status: **solved**

Submitted: 2019-02-01 00:00:00

Language: C

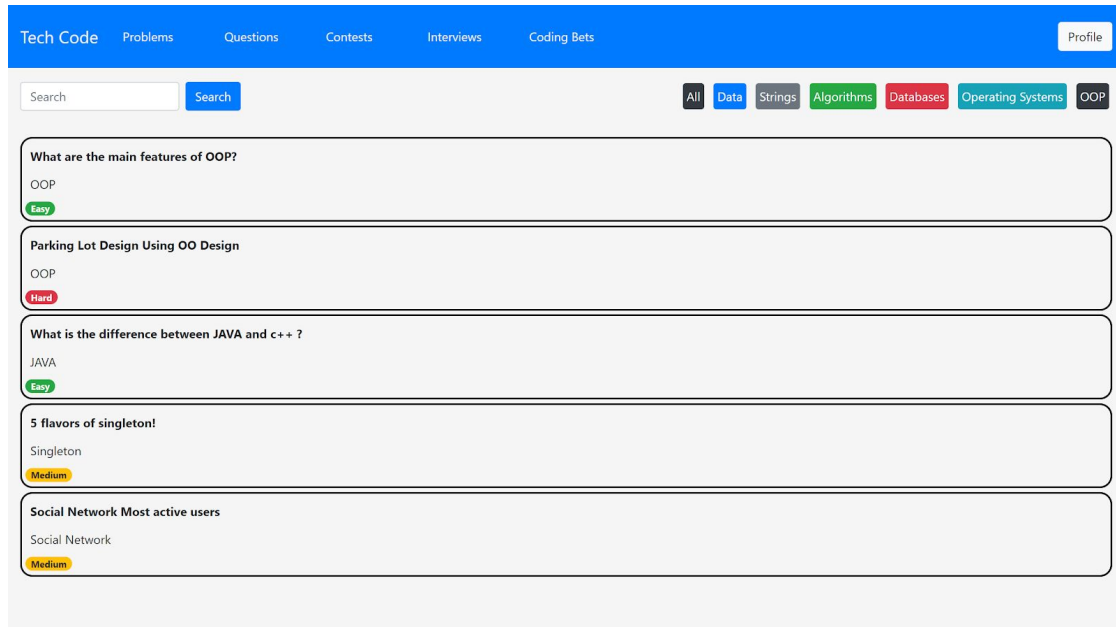
```
class Solution { public String solution(int solution) { //solution } public int searchInsert(int[] nums, int target) { Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output: 2));}}
```

Test

[Back To Problem](#)

The users can see their previously submitted code from this page.

Questions



Tech Code Problems Questions Contests Interviews Coding Bets Profile

Search Search

All Data Strings Algorithms Databases Operating Systems OOP

What are the main features of OOP?
OOP
Easy

Parking Lot Design Using OO Design
OOP
Hard

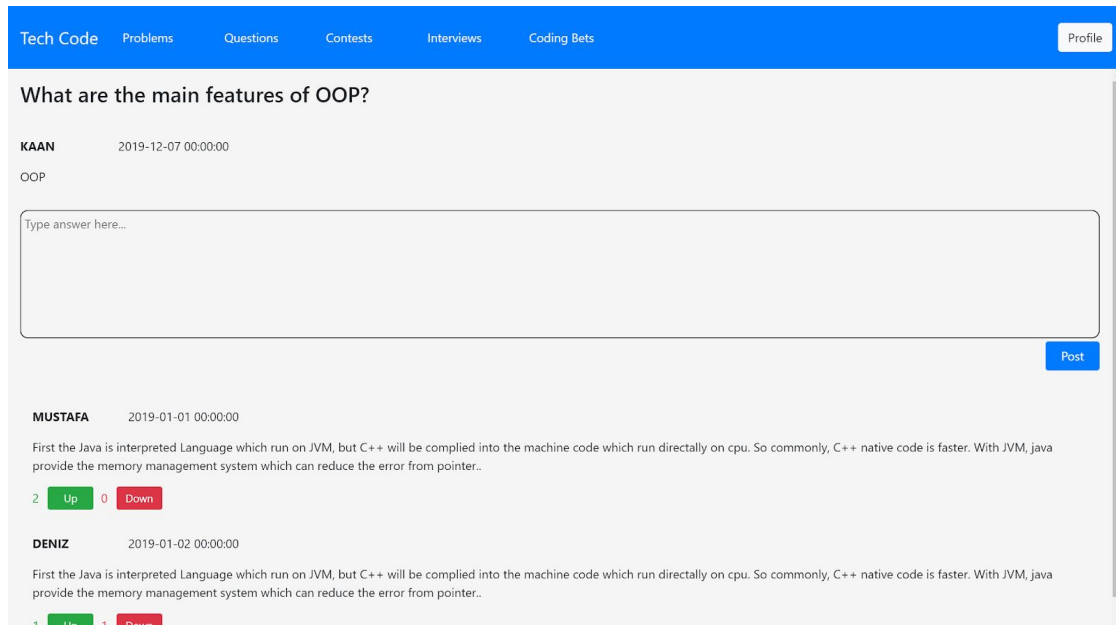
What is the difference between JAVA and c++ ?
JAVA
Easy

5 flavors of singleton!
Singleton
Medium

Social Network Most active users
Social Network
Medium

This page is the secondary page of the website. The user can choose a non-coding problem to try and solve. The users can filter problems with categories or they can search for keywords.

Answering a Question



Tech Code Problems Questions Contests Interviews Coding Bets Profile

What are the main features of OOP?

KAAN 2019-12-07 00:00:00
OOP

Type answer here...

Post

MUSTAFA 2019-01-01 00:00:00
First the Java is interpreted Language which run on JVM, but C++ will be compiled into the machine code which run directly on cpu. So commonly, C++ native code is faster. With JVM, java provide the memory management system which can reduce the error from pointer..
2 Up 0 Down

DENIZ 2019-01-02 00:00:00
First the Java is interpreted Language which run on JVM, but C++ will be compiled into the machine code which run directly on cpu. So commonly, C++ native code is faster. With JVM, java provide the memory management system which can reduce the error from pointer..
1 Up 1 Down

The users can answer the question from this page. In addition to that, the users can see other peoples results and upvote/downvote them.

Contests

[Tech Code](#) [Problems](#) [Questions](#) [Contests](#) [Interviews](#) [Coding Bets](#) [Profile](#)

Contests

Weekly Contest 1

Weekly Contest 2

Weekly Contest 3

Weekly Contest 4

Weekly Contest 5

New Contest2

Weekly Contest 6

The user can see available contests from this page.

Join A Contest

[Tech Code](#) [Problems](#) [Questions](#) [Contests](#) [Interviews](#) [Coding Bets](#) [Profile](#)

Weekly Contest 1

Start Date: 2019-01-01 00:00:00 End Date: 2020-01-02 00:00:00

Welcome to the 131th Weekly Contest

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Augue interdum velit euismod in pellentesque massa placerat duis ultricies. Aliquam purus sit amet luctus venenatis lectus magna. Vel elit scelerisque mauris pellentesque pulvinar pellentesque habitant. Faucibus turpis in eu mi bibendum neque egestas. Lacinia at quis risus sed. Et ultrices neque ornare aenean euismod elementum. Eu ultrices vitae auctor eu augue ut lectus arcu. Lectus proin nibh nisl condimentum id. Tristique sollicitudin nibh sit amet commodo nulla facilisi. Ultrices mi tempus imperdiet nulla malesuada. Pellentesque eu tincidunt tortor aliquam nulla facilisi cras fermentum odio. Et odio pellentesque diam volutpat commodo sed egestas egestas fringilla. Euismod lacinia at quis risus. Natoque penatibus et magnis dis parturient montes nascetur ridiculus.

Prize

1st

100

2nd

50

3rd

10

Join Contest

The user can join a contest from here and see the possible prizes on the right.

Interviews

The screenshot shows the 'Interviews' page with a blue navigation bar at the top containing links for Tech Code, Problems, Questions, Contests, Interviews, and Coding Bets. A 'Profile' button is located on the right side of the navigation bar. Below the navigation bar, the section is titled 'Available Interviews'. It displays six interview slots in a grid. Each slot includes a role name, a job type (internship or full time), and a 'Go' button.

Role	Job Type	Action
Game Developer	internship	Go
Data Miner	internship	Go
AI Expert	full time	Go
Full Stack Developer	full time	Go
Python Developer	part time	Go
Unity Developer.	part time	Go

The users can see available interviews on this page. The user can click go to start the interview.

Conducting An Interview

The screenshot shows the 'Conducting An Interview' page. It features a blue navigation bar at the top with links for Tech Code, Problems, Questions, Contests, Interviews, and Coding Bets. A 'Profile' button is on the right. Below the navigation bar, the section is titled 'Company Name'. Underneath, there is a 'Job Position' label. The main content area contains a table with two columns: 'Problem List' and 'Status'.

Problem List	Status
ZigZag Conversion	solved
Reverse Integer	solved

The user can conduct the interview from this page.

Coding Bets

The screenshot shows the 'Coding Bets' section of a web application. The top navigation bar is blue with links for 'Tech Code', 'Problems', 'Questions', 'Contests', 'Interviews', and 'Coding Bets'. A 'Profile' button is on the right. The main content area is divided into five columns: 'Open Bets', 'Ongoing Bets', 'Created Bets', 'Closed Bets', and 'Create Bet'. 'Open Bets' shows two cards for 'DENIZ' with values 400 and 500. 'Ongoing Bets' shows one card for 'DENIZ' with value 200. 'Created Bets' shows one card for 'MUSTAFA' with value 100. 'Closed Bets' shows one card with value 300. 'Create Bet' shows a form with a 'Wager' input field and a 'Create Bet' button.

The user can see the open bets that are created by other users and ready to be accepted. The user can also see its own ongoing bets. The user can see the bets that he created. The user can see his past bets. Finally, the user can create a new bet.

Past Interviews

The screenshot shows the 'Past Interviews' section of a web application. The top navigation bar is blue with links for 'Tech Code', 'Problems', 'Questions', 'Contests', 'Interviews', and 'Coding Bets'. A 'Profile' button is on the right. The main content area is titled 'Past Interviews' and shows the name 'MUSTAFA'. Below the name is a table with three rows of interview data.

2019-12-07 00:00:00	Google	waiting
2019-12-07 00:00:00	Google	accepted
2019-12-07 00:00:00	Google	rejected

The user can see his past interview performances.

6. Code Produced

<?php

```
define('DB_SERVER', 'remotemysql.com');
define('DB_USERNAME', 'OV8ehjf2GC');
define('DB_PASSWORD', 'FzxAUMbrqV');
define('DB_NAME', 'OV8ehjf2GC');
/*
define('DB_SERVER', 'localhost:2720');
define('DB_USERNAME', 'user1');
define('DB_PASSWORD', 'abcd');
define('DB_NAME', 'techcode');
*/
class SQLController
{
    public static $link;

    public static function connectSQL()
    {
        self::$link = new mysqli(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);

        // Check connection
        if (self::$link->connect_error) {
            die("ERROR: Could not connect. " . mysqli_connect_error());
        }
        //echo "Connected successfully" . "<br>";
    }

    public static function loginCheck($email, $password, $type)
    {
        if ($type == "user") {
            $q = "SELECT *
            FROM ACCOUNT JOIN USER ON account_id = user_id
            WHERE email = '$email' and password = '$password'";
        } else if ($type == "editor") {
            $q = "SELECT *
            FROM ACCOUNT JOIN EDITOR ON account_id = editor_id
            WHERE email = '$email' and password = '$password'";
        } else if ($type == "company") {
            $q = "SELECT *
            FROM ACCOUNT JOIN COMPANY ON account_id = company_id
            WHERE email = '$email' and password = '$password'";
        }

        $result = self::$link->query($q);

        return $result;
    }

    public static function getCompanyInfo($int_id)
    {
        $q = "SELECT *
        FROM INTERVIEW NATURAL JOIN COMPANY_INTERVIEW_PREPARE JOIN ACCOUNT ON account_id =
company_id
        WHERE interview_id = '$int_id'";
```

```

        $result = self::$link->query($q);

        return $result;
    }

    public static function acceptCodingBet($coding_bet_id, $user_id)
    {
        $q = "UPDATE CODING_BET
        SET status = 'ongoing'
        WHERE coding_bet_id = '$coding_bet_id'";

        self::$link->query($q);

        $q = "INSERT USER_CODING_BET_PARTICIPATE
        VALUES('$user_id', '$coding_bet_id', 0)";

        self::$link->query($q);
    }

    public static function getAccountInfo($id)
    {
        $q = "SELECT *
        FROM ACCOUNT
        WHERE account_id = '$id'";

        $result = self::$link->query($q);

        return $result;
    }

    public static function getProblemIdByCodingBet($coding_bet_id)
    {
        $q = "SELECT *
        FROM CODING_BET NATURAL JOIN CODING_BET_TASK_RELATION
        WHERE coding_bet_id = '$coding_bet_id'";

        $result = self::$link->query($q);

        return $result;
    }

    public static function getPublicCodingTasks($search_input, $category)
    {
        if ($category == "") {

            if ($search_input == "") {
                $q = "SELECT *
                FROM PUBLIC_CODING_TASK NATURAL JOIN PUBLIC_TASK NATURAL JOIN TASK
                ORDER BY task_id ASC";
            } else {
                $q = "SELECT *
                FROM PUBLIC_CODING_TASK NATURAL JOIN PUBLIC_TASK NATURAL JOIN TASK
                where title LIKE '%$search_input%'
                ORDER BY task_id ASC";
            }
        } else {

```

```

        if ($search_input == "") {
            $q = "SELECT *
                FROM PUBLIC_CODING_TASK NATURAL JOIN PUBLIC_TASK
                NATURAL JOIN TASK
                WHERE category = '$category'
                ORDER BY task_id ASC";
        } else {
            $q = "SELECT *
                FROM PUBLIC_CODING_TASK NATURAL JOIN PUBLIC_TASK
                NATURAL JOIN TASK
                WHERE category = '$category' and title LIKE %search_input%
                ORDER BY task_id ASC";
        }
    }

    $result = self::$link->query($q);

    return $result;
}

public static function getPublicCodingTasksInInterview($int_id)
{
    $q = "SELECT *
        FROM INTERVIEW_TASK NATURAL JOIN CODING_INTERVIEW_TASK
        NATURAL JOIN TASK
        WHERE interview_id = '$int_id'
        ORDER BY task_id ASC";

    $result = self::$link->query($q);

    return $result;
}

public static function getPublicCodingTasksInContest($contest_id)
{
    $q = "SELECT *
        FROM CONTEST_TASK
        NATURAL JOIN TASK
        WHERE contest_id = '$contest_id'
        ORDER BY task_id ASC";

    $result = self::$link->query($q);

    return $result;
}

public static function getStatusForInterviewProblem($problem_id, $user_id)
{
    $q = "SELECT *
        FROM INTERVIEW_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION NATURAL JOIN
SUBMISSION
        WHERE task_id = '$problem_id' AND user_id = '$user_id'";

```

```

        $result = self::$link->query($q);

        return $result;
    }

    public static function getStatusForContestProblem($problem_id, $user_id)
    {
        $q = "SELECT *
                FROM CONTEST_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION NATURAL JOIN
SUBMISSION
                WHERE task_id = '$problem_id' AND user_id = '$user_id'";

        $result = self::$link->query($q);

        return $result;
    }

    public static function getPublicNonCodingTasks($search_input, $category)
    {
        if ($category == "") {

            if ($search_input == "") {
                $q = "SELECT *
                    FROM PUBLIC_NON_CODING_TASK NATURAL JOIN PUBLIC_TASK NATURAL JOIN TASK
                    ORDER BY task_id ASC";
            } else {
                $q = "SELECT *
                    FROM PUBLIC_NON_CODING_TASK NATURAL JOIN PUBLIC_TASK NATURAL JOIN TASK
                    where title LIKE '%$search_input%'
                    ORDER BY task_id ASC";
            }
        } else {

            if ($search_input == "") {
                $q = "SELECT *
                    FROM PUBLIC_NON_CODING_TASK NATURAL JOIN PUBLIC_TASK
                    NATURAL JOIN TASK
                    WHERE category = '$category'
                    ORDER BY task_id ASC";
            } else {
                $q = "SELECT *
                    FROM PUBLIC_NON_CODING_TASK NATURAL JOIN PUBLIC_TASK
                    NATURAL JOIN TASK
                    WHERE category = '$category' and title LIKE %search_input%
                    ORDER BY task_id ASC";
            }
        }

        $result = self::$link->query($q);

        return $result;
    }

    public static function createUser($id, $name, $email, $password, $info)

```

```

{
    echo $name;
    echo $id;
    $query1 = "INSERT INTO ACCOUNT VALUES ('$id','$name','$email','$password','$info')";
    $query2 = "INSERT INTO USER VALUES ('$id',0,0)";

    $result1 = self::$link->query($query1);
    $result2 = self::$link->query($query2);
    return $result1 && $result2;
}

public static function createEditor($id, $name, $email, $password, $info)
{
    $query1 = "INSERT INTO ACCOUNT VALUES ('$id','$name','$email','$password','$info')";
    $query2 = "INSERT INTO EDITOR VALUES ('$id',1)";

    $result1 = self::$link->query($query1);
    $result2 = self::$link->query($query2);
    return $result1 && $result2;
}

public static function createCompany($id, $name, $email, $password, $info, $website)
{
    $query1 = "INSERT INTO ACCOUNT VALUES ('$id','$name','$email','$password','$info')";
    $query2 = "INSERT INTO COMPANY VALUES ('$id','$website')";

    $result1 = self::$link->query($query1);
    $result2 = self::$link->query($query2);
    return $result1 && $result2;
}

public static function getUserInfo($email, $type)
{
    if ($type == "user") {
        $query = "SELECT *
        FROM ACCOUNT JOIN USER ON account_id = user_id
        WHERE email = '$email'";
    } else if ($type == "editor") {
        $query = "SELECT *
        FROM ACCOUNT JOIN EDITOR ON account_id = editor_id
        WHERE email = '$email'";
    } else if ($type == "company") {
        $query = "SELECT *
        FROM ACCOUNT JOIN COMPANY ON account_id = company_id
        WHERE email = '$email'";
    }

    $result = self::$link->query($query);

    return $result;
}

public static function getCodingQuestionInfo($task_id)
{
    $query = "SELECT *
    FROM TASK NATURAL JOIN PUBLIC_TASK NATURAL JOIN PUBLIC_CODING_TASK

```



```

        WHERE task_id = '$task_id';

        $result = self::$link->query($query);

        return $result;
    }

    public static function getInterviewCodingQuestionInfo($task_id)
    {

        $query = "SELECT *
        FROM TASK NATURAL JOIN INTERVIEW_TASK NATURAL JOIN CODING_INTERVIEW_TASK
        WHERE task_id = '$task_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getContestCodingQuestionInfo($task_id)
    {

        $query = "SELECT *
        FROM TASK NATURAL JOIN CONTEST_TASK
        WHERE task_id = '$task_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getCodingBetQuestionInfo($task_id)
    {

        $query = "SELECT *
        FROM TASK NATURAL JOIN CODING_BET_TASK
        WHERE task_id = '$task_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function makeSubmission($sub_type, $sub_id, $prog_lang, $status, $task_id,
    $answer, $user_id)
    {
        echo $user_id;
        $date = "";
        if ($sub_type == "PUBLIC_CODING_SUBMISSION") {
            $date = ", '" . date('Y-m-d H:i:s') . "'";
        }

        $query1 = "INSERT INTO SUBMISSION
        VALUES('$sub_id', '$user_id', '$task_id', '$answer')";

        $query2 = "INSERT INTO CODING_SUBMISSION
        VALUES('$sub_id', '$prog_lang', '$status')";
    }

```

```

$query3 = "INSERT INTO $sub_type
VALUES('$sub_id'$date)";

$result1 = self::$link->query($query1);
$result2 = self::$link->query($query2);
$result3 = self::$link->query($query3);

return $result1 && $result2 && $result3;
}

public static function makeNonCodingSubmission($sub_type, $sub_id, $task_id, $answer,
$user_id)
{
    echo $user_id;
    $date = "";
    if ($sub_type == "PUBLIC_NON_CODING_SUBMISSION") {
        $date = ", '" . date('Y-m-d H:i:s') . "'";
    }

    $query1 = "INSERT INTO SUBMISSION
VALUES('$sub_id', '$user_id', '$task_id', '$answer')";

    $query2 = "INSERT INTO NON_CODING_SUBMISSION
VALUES('$sub_id')";

    $query3 = "INSERT INTO $sub_type
VALUES('$sub_id'$date)";

    $result1 = self::$link->query($query1);
    $result2 = self::$link->query($query2);
    $result3 = self::$link->query($query3);

    return $result1 && $result2 && $result3;
}

public static function getTestCases($task_id)
{
    $query = "SELECT *
FROM test_case
WHERE task_id = '$task_id'";

    $result = self::$link->query($query);

    return $result;
}

public static function getPreviousSubmissions($task_id, $user_id)
{
    //User submission view ama bu baÄŸlantÄ± nerde yapÄ±lacak
    $query = "SELECT *
FROM SUBMISSION NATURAL JOIN PUBLIC_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION
WHERE user_id = '$user_id' and task_id = '$task_id'";

    $result = self::$link->query($query);

    return $result;
}

```

```

    }

    public static function getSubmissionDetails($sub_id)
    {
        //User submission detail view ama bu baÄŸlantÄ± nerde yapÄ±lacak
        $query = "SELECT *
        FROM SUBMISSION NATURAL JOIN PUBLIC_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION
        WHERE submission_id = '$sub_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getNonCodingQuestionInfo($task_id)
    {
        $query = "SELECT *
        FROM PUBLIC_NON_CODING_TASK NATURAL JOIN PUBLIC_TASK NATURAL JOIN TASK NATURAL JOIN
EDITOR_PUBLIC_TASK_PREPARE
        WHERE task_id = '$task_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getAnswersForNonCodingQuestion($task_id)
    {
        $query = "SELECT DISTINCT *
        FROM PUBLIC_NON_CODING_SUBMISSION NATURAL JOIN SUBMISSION JOIN ACCOUNT ON account_id =
user_id
        WHERE task_id = '$task_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function giveAnswerToNonCodingQuestion($task_id, $date, $sub_id, $user_id,
$answer)
    {
        $query = "INSERT INTO public_non_coding_submission
VALUES ('$sub_id', '$date');
INSERT INTO submission
VALUES('$sub_id', '$user_id', '$task_id', '$answer')";

        self::$link->query($query);
    }

    public static function rateAnswer($user_id, $answer_id, $rate_type)
    {
        $query = "INSERT INTO rate_answer
VALUES ($user_id, '$answer_id', $rate_type)";
    }

```

```

        self::$link->query($query);
    }

    public static function deleteRateFromAnswer($user_id, $answer_id)
    {
        $query = "DELETE FROM TABLE rate_answer
        WHERE user_id = '$user_id' and answer_id = '$answer_id'";

        self::$link->query($query);
    }

    public static function getContestDetails($contest_id)
    {
        $query = "SELECT *
        FROM CONTEST
        WHERE contest_id = '$contest_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function joinContest($user_id, $contest_id)
    {
        $query = "INSERT INTO user_contest_participate
        VALUES ('$user_id', '$contest_id', 0)";

        self::$link->query($query);
    }

    public static function getContestProblems($contest_id)
    {
        $query = "SELECT *
        FROM contest_task NATURAL JOIN task
        WHERE contest_id = '$contest_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getContests()
    {
        $query = "SELECT *
        FROM CONTEST ";

        $result = self::$link->query($query);

        return $result;
    }
}

```

```

        public static function createContest($contest_id, $name, $start_date, $end_date, $info,
        $reward1, $reward2, $reward3, $editor_id)
        {

            $query1 = "INSERT INTO CONTEST
            VALUES
                ('$contest_id', '$name', '$start_date', '$end_date', '$reward1', '$reward2',
'$reward3')";

            $result1 = self::$link->query($query1);

            if ($result1) {
                echo "New record created successfully";
            } else {
                echo "Error: " . self::$link->error;
            }

            $date = date('Y-m-d H:i:s');

            $query2 = "INSERT INTO EDITOR_CODING_CONTEST_PREPARE
            VALUES
                ('$editor_id', '$contest_id', '$date')";
            $result2 = self::$link->query($query2);

            if ($result2) {
                echo "New record created successfully";
            } else {
                echo "Error: " . self::$link->error;
            }
            return $result1 && $result2;
        }

        public static function createPublicCodingTask($id, $title, $question, $category, $info,
        $difficulty, $example, $solution, $hint, $editor_id, $sample_inputs, $sample_outputs)
        {
            $date = date('Y-m-d H:i:s');

            $query1 = "INSERT INTO TASK
            VALUES('$id', '$title', '$question', '$category', '$info')";

            $query2 = "INSERT INTO PUBLIC_TASK
            VALUES('$id', '$difficulty', 0, 0)";

            $query3 = "INSERT INTO PUBLIC_CODING_TASK
            VALUES('$id', '$example', '$solution', '$hint')";

            $query4 = "INSERT INTO EDITOR_PUBLIC_TASK_PREPARE
            VALUES('$editor_id', '$id', '$date')";

            $result1 = self::$link->query($query1);
            if ($result1) {
                echo "New record created successfully";
            } else {
                echo "Error: " . self::$link->error;
            }
            $result2 = self::$link->query($query2);
            if ($result2) {

```

```

        echo "New record created successfully";
    } else {
        echo "Error: " . self::$link->error;
    }
}
$result3 = self::$link->query($query3);
if ($result3) {
    echo "New record created successfully";
} else {
    echo "Error: " . self::$link->error;
}
}
$result4 = self::$link->query($query4);
if ($result4) {
    echo "New record created successfully";
} else {
    echo "Error: " . self::$link->error;
}
}

for ($i = 0; $i < count($sample_inputs); $i++) {
    $query5 = "INSERT INTO TEST_CASE " .
        "VALUES('$id', ' " . $sample_inputs[$i] . "', ' " . $sample_outputs[$i] . "')";
    $result5 = self::$link->query($query5);
    if ($result5) {
        echo "New record created successfully";
    } else {
        echo "Error: " . self::$link->error;
    }
    if ($result5 === false) {
        return false;
    }
}

return $result1 && $result2 && $result3 && $result4;
}

public static function createContestTask($id, $title, $question, $category, $info,
$contest_point, $example, $solution, $hint, $contest_id, $editor_id, $sample_inputs,
$sample_outputs)
{
    $date = date('Y-m-d H:i:s');
    echo $contest_id;

    $query1 = "INSERT INTO TASK
VALUES('$id','$title', '$question', '$category', '$info')";

    $query2 = "INSERT INTO CONTEST_TASK
VALUES('$id', '$contest_point', '$example', '$solution', '$hint', '$contest_id')";

    $result1 = self::$link->query($query1);
    if ($result1) {
        echo "New record created successfully";
    } else {
        echo "Error: " . self::$link->error;
    }
}
$result2 = self::$link->query($query2);
if ($result2) {
    echo "New record created successfully";
} else {

```

```

        echo "Error: " . self::$link->error;
    }

    for ($i = 0; $i < count($sample_inputs); $i++) {
        $query4 = "INSERT INTO TEST_CASE " .
            "VALUES('$id', '" . $sample_inputs[$i] . "', '" . $sample_outputs[$i] . "')";
        $result4 = self::$link->query($query4);
        if ($result4) {
            echo "New record created successfully";
        } else {
            echo "Error: " . self::$link->error;
        }
        if ($result4 === false)
        {
            return false;
        }
    }

    return $result1 && $result2;
}

    public static function createProblemForContest($contest_id, $task_id, $contest_point,
$example, $solution, $hint, $title, $question, $category, $info)
    {

        $query = "INSERT INTO contest_task
VALUES ('$task_id', '$contest_point', '$example', '$solution', '$hint', '$contest_id');
INSERT INTO task
VALUES ('$task_id', '$title', '$question', '$category', '$info')";

        self::$link->query($query);
    }

    public static function createProblemForPublic($task_id, $points_given, $difficulty, $example,
$solution, $hint, $title, $question, $category, $info)
    {

        $query = "INSERT INTO public_coding_task
VALUES ('$task_id', '$example', '$solution', '$hint');
INSERT INTO public_task
VALUES ('$task_id', '$difficulty', '$points_given', 0);
INSERT INTO task
VALUES ('$task_id', '$title', '$question', '$category', '$info')";

        self::$link->query($query);
    }

    public static function createInterview($int_id, $job_type, $description, $duration,
$company_id)
    {

        $query = "INSERT INTO interview
VALUES ('$int_id', '$job_type', '$description', '$duration');
INSERT INTO company_interview_task_prepare
VALUES ('$company_id', '$int_id')";

        self::$link->query($query);
    }

```

```

    }

    public static function getInterviewQuestions($int_id)
    {

        $query = "SELECT *
        FROM interview_task NATURAL JOIN task
        WHERE interview_id = '$int_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getInterviews()
    {

        $query = "SELECT *
        FROM INTERVIEW";

        $result = self::$link->query($query);

        return $result;
    }

    public static function createProblemForInterview($task_id, $int_id, $date, $example,
    $solution, $hint, $title, $company_id, $category, $info)
    {

        $query = "INSERT INTO task
        VALUES ('$task_id', '$title', '$info', '$category', '$info');
        INSERT INTO interview_task
        VALUES ('$task_id', '$int_id');
        INSERT INTO coding_interview_task
        VALUES ('$task_id', '$example', '$solution', '$hint');
        INSERT INTO non_coding_interview_task
        VALUES ('$task_id');
        INSERT INTO company_interview_task_prepare
        VALUES ('$company_id', '$task_id', '$date')";

        self::$link->query($query);
    }

    public static function createPublicNonCodingTask($id, $title, $question, $category, $info,
    $difficulty, $editor_id)
    {

        $date = date('Y-m-d H:i:s');

        $query1 = "INSERT INTO TASK
        VALUES('$id', '$title', '$question', '$category', '$info')";

        $query2 = "INSERT INTO PUBLIC_TASK
        VALUES('$id', '$difficulty', 0, 0)";

        $query3 = "INSERT INTO PUBLIC_NON_CODING_TASK
        VALUES('$id')";
    }

```



```

$query4 = "INSERT INTO EDITOR_PUBLIC_TASK_PREPARE
VALUES('$editor_id', '$id', '$date');"

$result1 = self::$link->query($query1);
if ($result1) {
    echo "New record created successfully";
} else {
    echo "Error: " . self::$link->error;
}
$result2 = self::$link->query($query2);
if ($result2) {
    echo "New record created successfully";
} else {
    echo "Error: " . self::$link->error;
}
$result3 = self::$link->query($query3);
if ($result3) {
    echo "New record created successfully";
} else {
    echo "Error: " . self::$link->error;
}
$result4 = self::$link->query($query4);
if ($result4) {
    echo "New record created successfully";
} else {
    echo "Error: " . self::$link->error;
}

return $result1 && $result2 && $result3 && $result4;
}

public static function getPreviousSubmissionsInterviewCodingTask($user_id, $int_id)
{
    $query = " SELECT *
FROM interview_coding_submission NATURAL JOIN coding_submission NATURAL JOIN
submission
WHERE interview_id = '$int_id' and user_id = '$user_id'";

    $result = self::$link->query($query);

    return $result;
}

public static function getPreviousSubmissionsInterviewNonCodingTask($user_id, $int_id)
{
    $query = " SELECT *
FROM interview_non_coding_submission NATURAL JOIN non_coding_submission NATURAL JOIN
submission
WHERE interview_id = '$int_id' and user_id = '$user_id'";

    $result = self::$link->query($query);

    return $result;
}

```

```

    }

    public static function determineInterviewResult($user_id, $int_id, $result)
    {

        $query = "UPDATE user_interview_perform
        SET result = '$result'
        WHERE user_id = '$user_id' and interview_id = '$int_id'";

        self::$link->query($query);
    }

    public static function getPreviousInterviewResultsByUser($user_id)
    {

        $query = "SELECT *
        FROM USER_INTERVIEW_PERFORM NATURAL JOIN INTERVIEW NATURAL JOIN
        COMPANY_INTERVIEW_PREPARE
        WHERE company_id = '$user_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getPreviousInterviewResultsByCompany($company_id)
    {

        $query = "SELECT distinct *
        FROM USER_INTERVIEW_PERFORM NATURAL JOIN COMPANY_INTERVIEW_PREPARE NATURAL JOIN INTERVIEW
        WHERE company_id = '$company_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function createCodingBetChallenge($user_id, $coding_bet_id, $bet_amount,
    $task_id)
    {

        $query = "INSERT INTO coding_bet
        VALUES ('$coding_bet_id', '$bet_amount');
        INSERT INTO user_coding_bet_participate
        VALUES ('$user_id', '$coding_bet_id');
        INSERT INTO coding_bet_task_relation
        VALUES ('$coding_bet_id', '$task_id')";

        self::$link->query($query);
    }

    public static function getOpenCodingBetChallenges($user_id)
    {

        $q = "SELECT *
        FROM CODING_BET NATURAL JOIN USER_CODING_BET_PARTICIPATE
        WHERE owner_id != $user_id AND status = 'open'";

```

```

        $result = self::$link->query($q);

        return $result;
    }

    public static function getOnGoingCodingBetChallenges($user_id)
    {
        $q = "SELECT *
        FROM CODING_BET NATURAL JOIN USER_CODING_BET_PARTICIPATE
        WHERE user_id = $user_id AND status = 'ongoing'";

        $result = self::$link->query($q);

        return $result;
    }

    public static function getCodingBetOpponent($coding_bet_id, $user_id)
    {
        $q = "SELECT *
        FROM CODING_BET NATURAL JOIN USER_CODING_BET_PARTICIPATE
        WHERE user_id != $user_id AND coding_bet_id = $coding_bet_id";

        $result = self::$link->query($q);

        $result2 = SQLController::getAccountInfo($result->fetch_assoc()['user_id']);

        return $result2;
    }

    public static function getClosedCodingBetChallenges($user_id)
    {
        $q = "SELECT *
        FROM CODING_BET NATURAL JOIN USER_CODING_BET_PARTICIPATE
        WHERE user_id = $user_id AND status = 'closed'";

        $result = self::$link->query($q);

        return $result;
    }

    public static function getOpenCodingBetChallengesByUser($user_id)
    {
        $q = "SELECT *
        FROM CODING_BET NATURAL JOIN USER_CODING_BET_PARTICIPATE
        WHERE owner_id = $user_id AND status = 'open'";

        $result = self::$link->query($q);

        return $result;
    }

    public static function getNumberOfPublicCodingTasks()

```

```

{
    $query = "SELECT count(distinct task_id) as count
FROM PUBLIC_CODING_TASK";

    $result = self::$link->query($query);

    if (!$result) {
        return 0;
    }

    return $result->fetch_assoc()["count"];
}

public static function getNumberOfSolvedPublicCodingTasks($user_id)
{
    $query = "SELECT count(distinct task_id) as count
FROM PUBLIC_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION NATURAL JOIN SUBMISSION
WHERE user_id = '$user_id' and status= 'solved'";

    $result = self::$link->query($query);

    if (!$result) {
        return 0;
    }

    return $result->fetch_assoc()['count'];
}

public static function getNumberOfAttemptedPublicCodingTasks($user_id)
{
    $query = "SELECT count(distinct task_id) as count
FROM PUBLIC_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION NATURAL JOIN SUBMISSION
WHERE user_id = '$user_id' and status <> 'solved'";

    $result = self::$link->query($query);

    if (!$result) {
        return 0;
    }

    return $result->fetch_assoc()['count'];
}

public static function getNumberOfSubmissionsOnPublicCodingQuestion($user_id, $task_id)
{
    $query = "SELECT count(distinct submission_id) as count
FROM PUBLIC_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION
NATURAL JOIN SUBMISSION
WHERE user_id = '$user_id' and task_id = '$task_id'";

    $result = self::$link->query($query);

    return $result;
}

```

```

public static function getNumberOfPeopleSolvedPublicCodingQuestion($task_id)
{
    $query = "SELECT count(distinct user_id) as count
FROM PUBLIC_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION
NATURAL JOIN SUBMISSION
WHERE task_id = '$task_id' and status = 'solved'";

    $result = self::$link->query($query);

    if (!$result) {
        return 0;
    }

    return $result->fetch_assoc()['count'];
}

public static function getNumberOfAllSubmissionsOnPublicCodingQuestion($task_id)
{
    $query = "SELECT count(distinct submission_id) as count
FROM PUBLIC_CODING_SUBMISSION NATURAL JOIN CODING_SUBMISSION
NATURAL JOIN SUBMISSION
WHERE task_id = '$task_id'";

    $result = self::$link->query($query);

    if (!$result) {
        return 0;
    }

    return $result->fetch_assoc()['count'];
}

public static function getNumberOfPublicNonCodingTasks()
{
    $query = "SELECT count(distinct *) as count
FROM non_coding_public_task";

    $result = self::$link->query($query);

    return $result;
}

public static function getNumberOfUpVotesOfAnswer($answer_id)
{
    $query = "SELECT count(distinct user_id) as count
FROM RATE_ANSWER
WHERE answer_id = '$answer_id' and rate_answer = 'Upvote'";

    $result = self::$link->query($query);

    return $result;
}

```

```

    }

    public static function getNumberOfDownVotesOfAnswer($answer_id)
    {

        $query = "SELECT count(distinct user_id) as count
        FROM RATE_ANSWER
        WHERE answer_id = '$answer_id' and rate_answer = 'Downvote'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getNumberOfCoins($user_id)
    {

        $query = "SELECT tech_coin
        FROM USER
        WHERE user_id = '$user_id'";

        $result = self::$link->query($query);

        if (!$result) {
            return 0;
        }

        return $result->fetch_assoc()['tech_coin'];
    }

    public static function getNumberOfPeopleInContest($contest_id)
    {

        $query = "SELECT count(distinct user_id)
        FROM user_contest_participate
        WHERE contest_id = '$contest_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getNumberOfTasksByEditor($editor_id)
    {

        $query = "SELECT count(distinct task_id)
        FROM editor_public_task_prepare
        WHERE editor_id = '$editor_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function getNumberOfInterviewsByUser($user_id)
    {

```

```

        $query = "SELECT count(distinct interview_id)
        FROM user_interview_perform
        WHERE user_id = '$user_id'";

        $result = self::$link->query($query);

        return $result;
    }

    public static function checkIfUpvoted($user_id, $submission_id)
    {
        $query = "SELECT count(distinct user_id) as count
        FROM RATE_ANSWER
        WHERE user_id = '$user_id' and answer_id = '$submission_id' and rate_answer = 'Upvote'";

        $result = self::$link->query($query);
        if ($result === false) {
            return false;
        }
        return true;
    }

    public static function checkIfDownvoted($user_id, $submission_id)
    {
        $query = "SELECT count(distinct user_id) as count
        FROM RATE_ANSWER
        WHERE user_id = '$user_id' and answer_id = '$submission_id' and rate_answer =
'Downvote'";

        $result = self::$link->query($query);
        if ($result === false) {
            return false;
        }
        return true;
    }

    public static function upvoteSubmission($user_id, $submission_id)
    {
        $query = "INSERT INTO RATE_ANSWER
        VALUES('$user_id', '$submission_id', 'Upvote')";

        $result = self::$link->query($query);
        return $result;
    }

    public static function downvoteSubmission($user_id, $submission_id)
    {
        $query = "INSERT INTO RATE_ANSWER
        VALUES('$user_id', '$submission_id', 'Downvote')";

        $result = self::$link->query($query);
        return $result;
    }

    public static function deleteUpvote($user_id, $submission_id)
    {

```

```

        $query = "DELETE FROM RATE_ANSWER
        WHERE user_id = '$user_id' and answer_id = '$submission_id' and rate_answer = 'Upvote'";
        $result = self::$link->query($query);
        return $result;
    }

    public static function deleteDownvote($user_id, $submission_id)
    {
        $query = "DELETE FROM RATE_ANSWER
        WHERE user_id = '$user_id' and answer_id = '$submission_id' and rate_answer =
'Downvote'";
        $result = self::$link->query($query);
        return $result;
    }

    public static function participateInContest($user_id, $contest_id)
    {
        $query = "INSERT INTO USER_CONTEST_PARTICIPATE
        VALUES('$user_id', '$contest_id', 0)";
        $result = self::$link->query($query);
        return $result;
    }
}

<?php

class SQLDisplayer
{

    public static function loginCheck($email, $password, $type)
    {

        $result = SQLController::loginCheck($email, $password, $type);

        if ($result) {

            if ($result->num_rows > 0) {
                $_SESSION['account_id'] = $result->fetch_assoc()['account_id'];
                return true;
            } else {
                return false;
            }
        } else {
            echo "empty";
        }
    }

    //Problems Page
    public static function displayProblems($search_input, $category)
    {

        $result = SQLController::getPublicCodingTasks($search_input, $category);
        $size = 0;
        if ($result != NULL) {
            $size = $result->num_rows;
        }
    }
}

```



```

if ($size > 0) {
    while ($row = $result->fetch_assoc()) {
        $problem_id = $row["task_id"];
        $problem_name = $row["title"];
        $problem_percentage = $row["success_rate"];
        $problem_difficulty = $row["difficulty"];
        $pill_color = "success";
        if ($problem_difficulty == "Medium") {
            $pill_color = "warning";
        } else if ($problem_difficulty == "Hard") {
            $pill_color = "danger";
        }
        echo '<div class="problem">';
        echo '<div>' . $problem_id . '</div>';
        echo '<div> <a href="problem.php?problem_id=' . $problem_id . '&type=public">' .
$problem_name . '</a> </div>';
        echo '<div> <a href="solution.php?problem_id=' . $problem_id .
'&type=public">****</a> </div>';
        echo '<div>' . $problem_percentage . '</div>';
        echo '<div> <span class="badge badge-pill badge-' . $pill_color . '">' .
$problem_difficulty . '</span> </div>';
        echo '</div>';
    }
} else {
    echo "0 results";
}
}

```

```

public static function displayAnswersForNonCodingQuestion($task_id)
{
    $result = SQLController::getAnswersForNonCodingQuestion($task_id);

    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $user_id = $row["user_id"];
            $user_name = $row["name"];
            $answer = $row["answer"];
            $date = $row["date"];
            $submission_id = $row["submission_id"];

            $supvotes_result = SQLController::getNumberOfUpVotesOfAnswer($submission_id);
            $downvotes_result = SQLController::getNumberOfDownVotesOfAnswer($submission_id);

            if ($downvotes_result) {
                $downvotes = $downvotes_result->fetch_assoc()['count'];
            } else {
                $downvotes = 0;
            }
        }
    }
}

```

```

        if ($supvotes_result) {
            $supvotes = $supvotes_result->fetch_assoc()['count'];
        } else {
            $supvotes = 0;
        }
        echo '<!-- Comment -->';
        <div class="comment">

            <!-- Details -->
            <div class="details">

                <!-- Username -->
                <div>
                    <b>' . $user_name . ' </b>
                </div>

                <!-- Date -->
                <div> ' . $date . ' </div>
            </div>

            <!-- Content -->
            <div class="content">
                ' . $answer . '
            </div>

            <!-- Buttons -->
            <div class="buttons">
                <div class="text-success">' . $supvotes . '</div>
                <button name="upvote" value="" . $submission_id . ' type="submit"
class="btn btn-success btn-sm">Up</button>
                </div>
                <div class="text-danger">' . $downvotes . '</div>
                <button name="downvote" value="" . $submission_id . ' type="submit"
class="btn btn-danger btn-sm">Down</button>
            </div>

        </div>';
    }
} else {
    echo "0 results";
}
}

public static function displayPreviousSubmissions($task_id, $user_id)
{
    $result = SQLController::getPreviousSubmissions($task_id, $user_id);
    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }
    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $problem_prog_lang = $row["prog_lang"];
            $problem_status = $row["status"];
            $problem_date = $row["date"];
            $problem_sub_id = $row['submission_id'];

```

```

        echo '<div class="submission-item">
        <div>' . $problem_date . '</div>
        <div class="text-success">' . $problem_status . '</div>
        <div>' . $problem_prog_lang . '</div>
        <a href="submission.php?submission_id=' . $problem_sub_id . '&problem_id=' .
$task_id . '">***</a>
        </div>';
    }
    } else {
        echo "0 results";
    }
}

public static function displayPastInterviewsForCompany($company_id)
{
    $result = SQLController::getPreviousInterviewResultsByCompany($company_id);
    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }
    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            //$name = $row["prog_lang"];
            $status = $row["result"];
            $date = $row["date"];
            $description = $row['description'];
            $user_id = $row["user_id"];
            $participating_user_name =
SQLController::getAccountInfo($user_id)->fetch_assoc()['name'];
            echo ' <!-- Interview -->
            <div class="interview position-relative">
                <div> ' . $participating_user_name . ' </div>
                <div> ' . $description . ' </div>
                <div> ' . $date . ' </div>
                <div class="text-info"> ' . $status . ' </div>
                <a href="interview-result.php" class="stretched-link"></a>
            </div>';
        }
    } else {
        echo "0 results";
    }
}

public static function displayOpenBets($user_id)
{
    $result = SQLController::getOpenCodingBetChallenges($user_id);
    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }
    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $challenging_user_id = $row["user_id"];
            $coding_bet_id = $row["coding_bet_id"];

```

```

                                $challenging_user_name =
SQLController::getAccountInfo($challenging_user_id)->fetch_assoc()['name'];
                                $problem_id =
SQLController::getProblemIdByCodingBet($coding_bet_id)->fetch_assoc()['task_id'];
                                $amount = $row["bet_amount"];

                                echo " <div class='bet position-relative card'>
                                <h2>" . $challenging_user_name . "</h2>
                                <div>" . $amount . "</div>
                                <a href='problem.php?problem_id=" . $problem_id . "&type=coding_bet' class='btn
btn-primary mt-2' role='button'>Accept</a>
                                </div>";
                                }
                                } else {
                                    echo "0 results";
                                }
                            }
}

```

```

public static function displayOngoingBets($user_id)
{

    $result = SQLController::getOngoingCodingBetChallenges($user_id);
    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }
    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $amount = $row["bet_amount"];
            $coding_bet_id = $row["coding_bet_id"];
            $challenging_user_name = SQLController::getCodingBetOpponent($coding_bet_id,
$user_id)->fetch_assoc()['name'];
            echo " <div class='bet position-relative card'>
            <h2>" . $challenging_user_name . "</h2>
            <div>" . $amount . "</div>
            <a href='coding-bet.php?coding_bet_id=" . $coding_bet_id . "' class='btn btn-primary
mt-2' role='button'>Continue</a>
            </div>";
            }
        } else {
            echo "0 results";
        }
    }
}

```

```

public static function displayOpenBetsByUser($user_id)
{

    $result = SQLController::getOpenCodingBetChallengesByUser($user_id);
    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {

```

```

        $challenging_user_id = $row["user_id"];
        $challenging_user_name =
SQLController::getAccountInfo($challenging_user_id)->fetch_assoc()['name'];
        $amount = $row["bet_amount"];
        $coding_bet_id = $row["coding_bet_id"];

        echo " <div class='bet position-relative card'>
        <h2>" . $challenging_user_name . "</h2>
        <div>" . $amount . "</div>
    </div>";
    }
    } else {
        echo "0 results";
    }
}

public static function displayClosedBets($user_id)
{
    $result = SQLController::getClosedCodingBetChallenges($user_id);
    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $amount = $row["bet_amount"];
            $coding_bet_id = $row["coding_bet_id"];
            $challenging_user_name = SQLController::getCodingBetOpponent($coding_bet_id,
$user_id)->fetch_assoc()['name'];

            echo " <div class='bet position-relative card'>
            <h2>" . $challenging_user_name . "</h2>
            <div>" . $amount . "</div>
        </div>";
        }
    } else {
        echo "0 results";
    }
}

public static function displayContests()
{
    $result = SQLController::getContests();
    $size = $result->num_rows;

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $title = $row['name'];
            $id = $row['contest_id'];
            echo "<div class='contest position-relative card'>
            <h2>" . $title . "</h2>
            <a href='contest.php?contest_id=" . $id . "' class='stretched-link'></a>
        </div>";
        }
    }
}

```

```

    } else {
        echo "0 results";
    }
}

public static function displayInterviewsByUser($user_id)
{
    $result = SQLController::getPreviousInterviewResultsByUser($user_id);
    $size = $result->num_rows;

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $date = $row['date'];
            $status = $row['result'];
            $company_id = $row['company_id'];
            $company_name = SQLController::getAccountInfo($company_id->fetch_assoc()['name']);

            echo " <div class='interview border-bottom border-secondary'>
            <h5>" . $date . "</h5>
            <h5>" . $company_name . "</h5>
            <h5>" . $status . "</h5>
            </div>";
        }
    } else {
        echo "0 results";
    }
}

public static function displayInterviews()
{
    $result = SQLController::getInterviews();
    $size = $result->num_rows;

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $title = $row['description'];
            $job_type = $row['job_type'];
            $id = $row['interview_id'];
            echo "<div class='interview card'>
            <h2>" . $title . "</h2>
            <div> " . $job_type . " </div>
            <a class='btn btn-primary mt-2 mb-2' href='interview-started.php?int_id=" . $id .
            "' role='button'>Go</a>
            </div>";
        }
    } else {
        echo "0 results";
    }
}

public static function displayContestDetails($contest_id)
{

```

```

$result = SQLController::getContestDetails($contest_id);
$size = $result->num_rows;

if ($size > 0) {
    while ($row = $result->fetch_assoc()) {
        $title = $row['name'];
        $s_date = $row['start_date'];
        $e_date = $row['end_date'];
        $rew1 = $row['reward1'];
        $rew2 = $row['reward2'];
        $rew3 = $row['reward3'];
        $id = $row['contest_id'];
        echo " <div class='title'>
            <h3> " . $title . " </h3>
        </div>

        <div class='date'>
            Start Date: " . $s_date . "                End Date: " . $e_date . "
        </div>

        <div class='details'>
            <h5> Welcome to the 131th Weekly Contest </h5>
            <div>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit,
                sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
                Augue interdum velit euismod in pellentesque massa placerat duis ultricies.
                Aliquam purus sit amet luctus venenatis lectus magna.
                Vel elit scelerisque mauris pellentesque pulvinar pellentesque habitant.
                Faucibus turpis in eu mi bibendum neque egestas. Lacinia at quis risus sed.
                Et ultrices neque ornare aenean euismod elementum. Eu ultrices vitae auctor
eu augue ut lectus arcu.
                Lectus proin nibh nisl condimentum id. Tristique sollicitudin nibh sit amet
commodo nulla facilisi.
                Ultrices mi tempus imperdiet nulla malesuada.
                Pellentesque eu tincidunt tortor aliquam nulla facilisi cras fermentum odio.
                Et odio pellentesque diam volutpat commodo sed egestas egestas fringilla.
Euismod lacinia at quis risus.
                Natoque penatibus et magnis dis parturient montes nascetur ridiculus.
            </div>
        </div>

        <!-- Prize -->
        <div class='prizes'>
            <h5>Prize</h5>
            <div class='prize'>
                <div>1st</div>
                <div>" . $rew1 . "</div>
            </div>
            <div class='prize'>
                <div>2nd</div>
                <div>" . $rew2 . "</div>
            </div>
            <div class='prize'>
                <div>3rd</div>
                <div>" . $rew3 . "</div>
            </div>
        </div>
    }
}

```

```

        </div>

        </div>";
    }
} else {
    echo "0 results";
}
}

public static function displaySubmission($sub_id, $task_id)
{
    $question = SQLController::getCodingQuestionInfo($task_id);
    $size = $question->num_rows;
    if ($size > 0) {
        while ($row = $question->fetch_assoc()) {
            $problem_title = $row["title"];
        }
    }
    $result = SQLController::getSubmissionDetails($sub_id);
    $size = $result->num_rows;

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $problem_prog_lang = $row["prog_lang"];
            $problem_status = $row["status"];
            $problem_date = $row["date"];
            $problem_sub_id = $row['submission_id'];
            $problem_answer = $row['answer'];
            echo ' <div class="wrapper">

<!-- Title -->
<div class="title">
    <a href="#">
        <h5>' . $problem_title . '</h5>
    </a>
</div>

<!-- Header -->
<div class="head">
    <h4>
        Submission Detail
    </h4>
</div>

<!-- Detail -->
<div class="detail">
    <div class="detail-inside">

        <!-- Status -->
        <div></div>
        <div>Status: </div>
        <div class="text-danger">' . $problem_status . '</div>

        <!-- Date -->
        <div></div>
        <div> Submitted: </div>
        <div>' . $problem_date . '</div>

```



```

        <!-- Language -->
        <div></div>
        <div> Language:</div>
        <div>' . $problem_prog_lang . ' </div>

    </div>
</div>

<!-- Code -->
<div class="code">
    <div class="code-inside">

        <!-- Code Text -->
        <xmp class="code-text">' . $problem_answer . '</xmp>

        <!-- Code Test -->
        <div class="code-test">Test</div>
    </div>
</div>

<!-- Back Button -->
<div class="foot">
    <a href="submissions.php?problem_id=' . $task_id . '">Back To Problem</a>
</div>    </div>';
    }
} else {
    echo "0 results";
}
}

public static function displayCompanyInfo($int_id)
{
    $result = SQLController::getCompanyInfo($int_id);
    $size = $result->num_rows;

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $name = $row["name"];
            $position = $row["job_type"];
            $description = $row["description"];
            echo '<!-- Company Name -->
            <div class="c-name">
                <h3> ' . $name . ' </h3>
            </div>

            <!-- Job Position -->
            <div class="j-position">
                ' . $description . ": " . $position . '
            </div>';
        }
    } else {
        echo "0 results";
    }
}

public static function displaySolution($task_id)

```

```

{

$result = SQLController::getCodingQuestionInfo($task_id);
$size = $result->num_rows;

if ($size > 0) {
    while ($row = $result->fetch_assoc()) {
        $problem_solution = $row['solution'];

        echo ' <!-- Text Area -->
        <div class="cont2">
            <pre class="code-text">
' . $problem_solution . '
            </pre>
        </div>';
    }
} else {
    echo "0 results";
}
}

public static function displayQuestions($search_input, $category)
{

$result = SQLController::getPublicNonCodingTasks($search_input, $category);

$size = 0;
if ($result != NULL) {
    $size = $result->num_rows;
}

if ($size > 0) {
    while ($row = $result->fetch_assoc()) {
        $problem_id = $row["task_id"];
        $problem_name = $row["title"];
        //$problem_percentage = $row["success_rate"];
        $problem_difficulty = $row["difficulty"];
        $problem_info = $row["info"];
        $pill_color = "success";
        if ($problem_difficulty == "Medium") {
            $pill_color = "warning";
        } else if ($problem_difficulty == "Hard") {
            $pill_color = "danger";
        }

        echo ' <div class="question position-relative">
        <div class="title">
            <b>' . $problem_name . '</b>
        </div>
        <div class="details">' . $problem_info . ' </div>
        <a href="non-coding-question.php?problem_id=' . $problem_id . '"
        class="stretched-link"></a>
            <div> <span class="badge badge-pill badge-' . $pill_color . '">' .
$problem_difficulty . '</span> </div>
        </div>';
    }
} else {

```

```

        echo "0 results";
    }
}

public static function displayInterviewProblems($int_id, $user_id)
{
    $result = SQLController::getPublicCodingTasksInInterview($int_id);
    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $problem_id = $row["task_id"];
            $problem_name = $row["title"];
            $status = SQLController::getStatusForInterviewProblem($problem_id,
$user_id->fetch_assoc()['status']);
            echo ' <div class="problem">
                <a href="problem.php?problem_id=' . $problem_id . '&type=interview&int_id=' .
$int_id . '"> ' . $problem_name . '</a>
                <div class="mr-4 text-danger">' . $status . '</div>
            </div>';
        }
    } else {
        echo "0 results";
    }
}

public static function displayContestProblems($contest_id, $user_id)
{
    $result = SQLController::getPublicCodingTasksInContest($contest_id);
    $size = 0;
    if ($result != NULL) {
        $size = $result->num_rows;
    }

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $problem_id = $row["task_id"];
            $problem_name = $row["title"];
            $status = SQLController::getStatusForContestProblem($problem_id,
$user_id->fetch_assoc()['status']);
            echo ' <div class="problem">
                <a href="problem.php?problem_id=' . $problem_id . '&type=contest&contest_id=' .
$contest_id . '"> ' . $problem_name . '</a>
                <div class="mr-4 text-danger">' . $status . '</div>
            </div>';
        }
    } else {
        echo "0 results";
    }
}

```

```

public static function displayNonCodingQuestionDetails($task_id)
{

    $result = SQLController::getNonCodingQuestionInfo($task_id);
    $size = $result->num_rows;

    if ($size > 0) {
        while ($row = $result->fetch_assoc()) {
            $problem_id = $row["task_id"];
            $problem_name = $row["title"];
            $editor_id = $row["editor_id"];
            $date = $row['date'];
            $editor_name = SQLController::getAccountInfo($editor_id)->fetch_assoc()['name'];

            $problem_info = $row["info"];

            echo ' <!-- Title -->
            <div class="title">
                <div>
                    <h3> ' . $problem_name . ' </h3>
                </div>
            </div>

            <!-- Question -->
            <div class="question">

                <!-- Details -->
                <div class="details">

                    <!-- Username -->
                    <div>
                        <b> ' . $editor_name . ' </b>
                    </div>

                    <!-- Date -->
                    <div> ' . $date . ' </div>
                </div>

                <!-- Content -->
                <div class="content">
                    ' . $problem_info . '
                </div>

            </div>';
        }
    } else {
        echo "There is not a question with this id";
    }
}

public static function displayCodingQuestionDetails($task_id, $type)
{
    if ($type == "interview") {
        $result = SQLController::getInterviewCodingQuestionInfo($task_id);
    } else if ($type == "public") {
        $result = SQLController::getCodingQuestionInfo($task_id);
    }
}

```

```

    } else if ($type == "contest") {
        $result = SQLController::getContestCodingQuestionInfo($task_id);
    } else if ($type == "coding_bet") {
        $result = SQLController::getCodingBetQuestionInfo($task_id);
    }
}

$all_submissions = SQLController::getNumberOfAllSubmissionsOnPublicCodingQuestion($task_id);
$accepted_submissions = SQLController::getNumberOfPeopleSolvedPublicCodingQuestion($task_id);

$size = $result->num_rows;

if ($size > 0) {
    while ($row = $result->fetch_assoc()) {
        $problem_id = $row["task_id"];
        $problem_example = $row["example"];
        $problem_name = $row["title"];
        $problem_info = $row["info"];
        // $problem_percentage = $row["success_rate"];
        if ($type == "public") {
            $problem_difficulty = $row["difficulty"];

            $pill_color = "success";
            if ($problem_difficulty == "Medium") {
                $pill_color = "warning";
            } else if ($problem_difficulty == "Hard") {
                $pill_color = "danger";
            }
        }

        echo ' <div class="desc">
        <h6>1.' . $problem_name . '</h6>
        <p><span class="badge badge-pill badge-' . $pill_color . '"> ' .
        $problem_difficulty . '</span></p>
        <p>
        ' . $problem_info . '
        </p>
        <b>Example:</b>
        <samp>
        <br>
        ' . $problem_example . '<br>
        </samp>
        <div class="stats">
        <p>Accepted ' . $accepted_submissions . '</p>
        <p>Submissions ' . $all_submissions . '</p>
        </div>
        </div>';
    } else {
        echo ' <div class="desc">
        <h6>1.' . $problem_name . '</h6>
        <p></p>
        <p>
        ' . $problem_info . '
        </p>
        <b>Example:</b>
        <samp>

```

```

        <br>
        ' . $problem_example . ' <br>
    </samp>
    <div class="stats">
        <p>Accepted ' . $accepted_submissions . ' </p>
        <p>Submissions ' . $all_submissions . ' </p>
    </div>
</div>';
}
}
} else {
    echo "There is not a question with this id";
}
}

public static function displayProgress()
{
    $all_tasks = SQLController::getNumberOfPublicCodingTasks();
    $solved_tasks =
SQLController::getNumberOfSolvedPublicCodingTasks($_SESSION['account_id']);
    $attempted_task =
SQLController::getNumberOfAttemptedPublicCodingTasks($_SESSION['account_id']);
    $coins = SQLController::getNumberOfCoins($_SESSION['account_id']);

    // echo $_SESSION['account_id'];
    // echo '\n';
    // echo $all_tasks . ' ' . $solved_tasks . ' ' . $attempted_task;

    //if ($attempted_task && $all_tasks && $solved_tasks) {
    echo '
        <div class="prog">

            <!-- Title -->
            <div class="prog-title">
                <h4>Your Progress</h4>
            </div>

            <!-- Percentage -->
            <div class="prog-percent">
                <h1>' . number_format(($solved_tasks / $all_tasks * 100), 2) . '%</h1>
            </div>

            <!-- Details -->
            <div class="prog-details">

                <!-- Detail -->
                <div class="prog-detail">
                    <div>' . ($all_tasks - $solved_tasks) . ' </div>
                    <div>Todo</div>
                </div>

                <div class="prog-detail">
                    <div>' . $solved_tasks . '/' . $all_tasks . ' </div>
                    <div>Solved</div>
                </div>

                <div class="prog-detail">

```

```

        <div>' . $attempted_task . '</div>
        <div>Attempted</div>
        </div>
    </div>

    <!-- Gold -->
    <div class="prog-gold">
        <h4>' . $coins . ' Gold</h4>
    </div>

    </div> ';
    //}
}

public static function displayAccountInfo()
{
    if ($_SESSION["account_type"] == 'user') {
        echo '<div class="wrapper">

        <!-- Name -->
        <div class="username">
            <h1>' . $_SESSION['account_name'] . '</h1>
        </div>

        <!-- Account Type -->
        <div class="account">
            <b>Account Type:</b> ' . $_SESSION['account_type'] . '
        </div>

        <!-- Email -->
        <div class="email">
            <b>Email:</b> ' . $_SESSION['account_email'] . '
        </div>

        <!-- Info -->
        <div class="info">
            <b>Info:</b> ' . $_SESSION['account_info'] . '
        </div>

        <!-- Logout -->
        <div class="buttons">
            <a class="btn btn-primary m-1" href="interview-results-user.php"
role="button">Interviews</a>
            <a class="btn btn-primary m-1" href="index.php" role="button">Log Out</a>
        </div>    </div>';
    } else if ($_SESSION["account_type"] == 'editor') {

        echo '<!-- Name -->
        <div class="username">
            <h1>' . $_SESSION['account_name'] . '</h1>
        </div>

        <!-- Account Type -->
        <div class="account">
            <b>Account Type:</b> ' . $_SESSION['account_type'] . '
        </div>

```

```

        <!-- Email -->
        <div class="email">
            <b>Email:</b> ' . $_SESSION['account_email'] . '
        </div>

        <!-- Buttons -->
        <div class="buttons">
            <a class="btn btn-primary m-1" href="create-problem.php" role="button">Create
Problem</a>
            <a class="btn btn-primary m-1" href="create-non-coding-question.php"
role="button">Create Question</a>
            <a class="btn btn-primary m-1" href="contests-editor.php"
role="button">Contests</a>
            <a class="btn btn-primary m-1" href="index.php" role="button">Log Out</a>
        </div>';
    } else if ($_SESSION["account_type"] == 'company') {
        echo ' <!-- Name -->
        <div class="username">
            <h1>' . $_SESSION['account_name'] . ' </h1>
        </div>

        <!-- Account Type -->
        <div class="account">
            <b>Account Type:</b> ' . $_SESSION['account_type'] . '
        </div>

        <!-- Email -->
        <div class="email">
            <b>Email:</b> ' . $_SESSION['account_email'] . '
        </div>

        <!-- Buttons -->
        <div class="buttons">
            <a class="btn btn-primary m-1" href="interviews-company.php"
role="button">Interviews</a>
            <a class="btn btn-primary m-1" href="interviews.php" role="button">See Past
Interviews</a>
            <a class="btn btn-primary m-1" href="index.php" role="button">Log Out</a>
        </div>';
    }
}

```

```

import java.util.*;
import java.sql.SQLException;

public class InitTablesQueryBuilder {

    private static String getInsertQuery(String table, String... param) {
        StringBuilder builder = new StringBuilder();
        builder.append("INSERT INTO ");
        builder.append(table);
        builder.append(" VALUES(");
        for (int i = 0; i < param.length - 1; i++) {
            builder.append(param[i]);
            builder.append(",");
        }
    }
}

```



```

        builder.append(param[param.length - 1]);
        builder.append(");");
        return builder.toString();
    }

    public static void addAccount() {
        try {
            Controller.statement.executeUpdate(
                getInsertQuery("ACCOUNT", "1", "'AHMET'", "'ahmet@gmail.com'", "'a'", "'20
years experience'"));
            Controller.statement.executeUpdate(
                getInsertQuery("ACCOUNT", "2", "'KAAN'", "'kaan@gmail.com'", "'a'", "'2 year
experience'"));
            Controller.statement.executeUpdate(
                getInsertQuery("ACCOUNT", "3", "'MUSTAFA'", "'mustafa@gmail.com'", "'a'",
"'student'"));
            Controller.statement
                .executeUpdate(getInsertQuery("ACCOUNT", "4", "'DENIZ'", "'deniz@gmail.com'",
"'a'", "'student'"));
            Controller.statement.executeUpdate(
                getInsertQuery("ACCOUNT", "5", "'Google'", "'google@gmail.com'", "'a'",
"'Tech Giant'"));
            Controller.statement.executeUpdate(getInsertQuery("ACCOUNT", "6", "'Microsoft'",
"'microsoft@gmail.com'",
"'a'", "'Looking for Employees'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to account table...");
            E.printStackTrace();
        }
    }

    public static void addEditor() {
        try {
            Controller.statement.executeUpdate(getInsertQuery("EDITOR", "1", "1"));
            Controller.statement.executeUpdate(getInsertQuery("EDITOR", "2", "2"));
        } catch (SQLException E) {
            System.out.println("Error while adding editor table...");
            E.printStackTrace();
        }
    }

    public static void addUser() {
        try {
            Controller.statement.executeUpdate(getInsertQuery("USER", "3", "0", "100"));
            Controller.statement.executeUpdate(getInsertQuery("USER", "4", "0", "100"));
        } catch (SQLException E) {
            System.out.println("Error while adding to user table...");
            E.printStackTrace();
        }
    }

    public static void addCompany() {
        try {
            Controller.statement.executeUpdate(getInsertQuery("COMPANY", "5", "'google.com'"));
            Controller.statement.executeUpdate(getInsertQuery("COMPANY", "6",
"'microsoft.com'"));

```

```

        } catch (SQLException E) {
            System.out.println("Error while adding to company table...");
            E.printStackTrace();
        }
    }

    public static void addJob() {
        try {
            Controller.statement.executeUpdate(
                getInsertQuery("JOB", "5", "'game developer'", "'Hyper-casual genre game
developer.'"));
            Controller.statement
                .executeUpdate(getInsertQuery("JOB", "5", "'junior game developer'", "'Unity
Game developer.'"));
            Controller.statement
                .executeUpdate(getInsertQuery("JOB", "6", "'intern software developer'", "'3
months internship.'"));
            Controller.statement.executeUpdate(
                getInsertQuery("JOB", "6", "'senior software developer'", "'Machine learning
expert.'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to job table...");
            E.printStackTrace();
        }
    }

    public static void addInterview() {
        String tableName = "INTERVIEW";
        try {
            Controller.statement
                .executeUpdate(getInsertQuery(tableName, "1", "'internship'", "'Game
Developer'", "30"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "2", "'internship'",
"'Data Miner'", "30"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "3", "'full time'", "'AI
Expert'", "30"));
            Controller.statement
                .executeUpdate(getInsertQuery(tableName, "4", "'full time'", "'Full Stack
Developer'", "30"));
            Controller.statement
                .executeUpdate(getInsertQuery(tableName, "5", "'part time'", "'Python
Developer'", "30"));
            Controller.statement
                .executeUpdate(getInsertQuery(tableName, "6", "'part time'", "'Unity
Developer.'", "30"));
        } catch (SQLException E) {
            System.out.println("Error while adding to interview table...");
            E.printStackTrace();
        }
    }

    public static void addCodingContest() {
        String tableName = "CONTEST";
        try {
            Controller.statement.executeUpdate(getInsertQuery(tableName, "1", "'Weekly Contest
1'",
                "'2019-01-01 00:00:00'", "'2020-01-02 00:00:00'", "100", "50", "10"));

```

```

        Controller.statement.executeUpdate(getInsertQuery(tableName, "2", "'Weekly Contest
2'",
        "'2019-01-08 00:00:00'", "'2020-01-09 00:00:00'", "100", "50", "10"));
        Controller.statement.executeUpdate(getInsertQuery(tableName, "3", "'Weekly Contest
3'",
        "'2019-01-15 00:00:00'", "'2020-01-16 00:00:00'", "100", "50", "10"));
        Controller.statement.executeUpdate(getInsertQuery(tableName, "4", "'Weekly Contest
4'",
        "'2019-01-22 00:00:00'", "'2020-01-23 00:00:00'", "100", "50", "10"));
        Controller.statement.executeUpdate(getInsertQuery(tableName, "5", "'Weekly Contest
5'",
        "'2019-01-29 00:00:00'", "'2020-01-30 00:00:00'", "100", "50", "10"));
        Controller.statement.executeUpdate(getInsertQuery(tableName, "6", "'Weekly Contest
6'",
        "'2019-02-06 00:00:00'", "'2020-02-07 00:00:00'", "100", "50", "10"));
    } catch (SQLException E) {
        System.out.println("Error while adding to coding contest table...");
        E.printStackTrace();
    }
}

public static void addCodingBet() {
    String tableName = "CODING_BET";
    try {
        Controller.statement.executeUpdate(getInsertQuery(tableName, "1", "100", "'open'",
"3'"));
        Controller.statement.executeUpdate(getInsertQuery(tableName, "2", "200", "'ongoing'",
"3'"));
        Controller.statement.executeUpdate(getInsertQuery(tableName, "3", "300", "'closed'",
"3'"));
        Controller.statement.executeUpdate(getInsertQuery(tableName, "4", "400", "'open'",
"4'"));
        Controller.statement.executeUpdate(getInsertQuery(tableName, "5", "500", "'open'",
"4'"));

    } catch (SQLException E) {
        System.out.println("Error while adding to coding bet table...");
        E.printStackTrace();
    }
}

public static void addTask() {
    String table = "TASK";
    try {
        // Coding
        Controller.statement.executeUpdate(getInsertQuery(table, "100", "'Two Sum'",
"'Question 0'", "'data'",
        "'Given an array of integers, return indices of the two numbers such that
they add up to a specific target. You may assume that each input would have exactly one solution,
and you may not use the same element twice.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "101", "'Add Two Numbers'",
"'Question 1'",
        "'strings'",
        "'You are given two non-empty linked lists representing two non-negative
integers. The digits are stored in reverse order and each of their nodes contain a single digit.
Add the two numbers and return it as a linked list. You may assume the two numbers do not contain
any leading zero, except the number 0 itself.'"));
    }
}

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "102",
            "'Longest Substring Without Repeating Characters'", "'Question 2'",
            "'algorithms'",
            "'Given a string, find the length of the longest substring without repeating
characters.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "103", "'Median of Two
Sorted Arrays'",
            "'Question 3'", "'databases'",
            "'There are two sorted arrays nums1 and nums2 of size m and n respectively.
Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).
You may assume nums1 and nums2 cannot be both empty.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "104", "'Longest Palindromic
Substring'",
            "'Question 4'", "'os'",
            "'Given a string s, find the longest palindromic substring in s. You may
assume that the maximum length of s is 1000.'"));
        // //////////////////////////////////
        Controller.statement.executeUpdate(getInsertQuery(table, "105", "'What are the main
features of OOP?'",
            "'Question 5'", "'oop'", "'OOP'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "106", "'Parking Lot Design
Using OO Design'",
            "'Question 6'", "'data'", "'OOP'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "107",
            "'What is the difference between JAVA and c++ ?'", "'Question 7'",
            "'strings'", "'JAVA'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "108", "'5 flavors of
singleton!'",
            "'Design Patterns'", "'algorithms'", "'Singleton'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "109", "'Social Network Most
active users'",
            "'Question 9'", "'databases'", "'Social Network'"));
        //////////////
        Controller.statement
            .executeUpdate(getInsertQuery(table, "110", "'Title 10'", "'Question 10'",
            "'os'", "'Info 10'"));
        Controller.statement
            .executeUpdate(getInsertQuery(table, "111", "'Title 11'", "'Question 11'",
            "'oop'", "'Info 11'"));
        //////////////
        Controller.statement.executeUpdate(getInsertQuery(table, "112", "'ZigZag
Conversion'", "'Question 12'",
            "'data'",
            "'The string PAYPALISHIRING is written in a zigzag pattern on a given number
of rows like this: (you may want to display this pattern in a fixed font for better legibility) P
A H N A P L S I I G Y I R And then read line by line: PAHNAPLSIIGYIR Write the code that will
take a string and make this conversion given a number of rows:'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "113", "'Reverse Integer'",
            "'Question 13'",
            "'strings'", "'Given a 32-bit signed integer, reverse digits of an
integer.'"));
        //////////////
        Controller.statement.executeUpdate(getInsertQuery(table, "114", "'Palindrome
Number'", "'Question 14'",
            "'algorithms'",
            "'Determine whether an integer is a palindrome. An integer is a palindrome
when it reads the same backward as forward.'"));

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "115", "'Regular Expression
Matching'",
            "'Question 15'", "'databases'",
            "'Given an input string (s) and a pattern (p), implement regular expression
matching with support for . and *.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "116", "'Container With Most
Water'",
            "'Question 16'", "'os'",
            "'Given n non-negative integers a1, a2, ..., an , where each represents a
point at coordinate (i, ai). n vertical lines are drawn such that the two endpoints of line i is
at (i, ai) and (i, 0). Find two lines, which together with x-axis forms a container, such that
the container contains the most water.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "117", "'Integer to Roman'",
            "'Question 17'",
            "'oop'", "'Roman numerals are represented by seven different symbols: I, V,
X, L, C, D and M.'"));
        //////////
        Controller.statement.executeUpdate(getInsertQuery(table, "118", "'Count and Say'",
            "'Question 18'",
            "'data'",
            "'The count-and-say sequence is the sequence of integers with the first five
terms as following: 1 is read off as one 1 or 11. 11 is read off as two 1s or 21. 21 is read off
as one 2, then one 1 or 1211.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "119", "'Title 19'",
            "'search-insert-position'",
            "'strings'",
            "'Given a sorted array and a target value, return the index if the target is
found. If not, return the index where it would be if it were inserted in order. You may assume no
duplicates in the array.'"));
        //////////
        Controller.statement.executeUpdate(
            getInsertQuery(table, "120", "'Title 20'", "'Question 20'", "'algorithms'",
            "'Info 20'"));
        Controller.statement.executeUpdate(
            getInsertQuery(table, "121", "'Title 21'", "'Question 21'", "'databases'",
            "'Info 21'"));
        Controller.statement
            .executeUpdate(getInsertQuery(table, "122", "'Title 22'", "'Question 22'",
            "'os'", "'Info 22'"));
        Controller.statement
            .executeUpdate(getInsertQuery(table, "123", "'Title 23'", "'Question 23'",
            "'oop'", "'Info 23'"));
        Controller.statement
            .executeUpdate(getInsertQuery(table, "124", "'Title 24'", "'Question 24'",
            "'data'", "'Info 24'"));
        Controller.statement.executeUpdate(
            getInsertQuery(table, "125", "'Title 25'", "'Question 25'", "'strings'",
            "'Info 25'"));
        Controller.statement.executeUpdate(
            getInsertQuery(table, "126", "'Title 26'", "'Question 26'", "'algorithms'",
            "'Info 26'"));
        Controller.statement.executeUpdate(
            getInsertQuery(table, "127", "'Title 27'", "'Question 27'", "'databases'",
            "'Info 27'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to task table...");
        E.printStackTrace();
    }
}

```

```

    }
}

public static void addPublicTask() {
    String table = "PUBLIC_TASK";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "100", "'Easy'", "10",
"85"));
        Controller.statement.executeUpdate(getInsertQuery(table, "101", "'Medium'", "20",
"68"));
        Controller.statement.executeUpdate(getInsertQuery(table, "102", "'Medium'", "20",
"77"));
        Controller.statement.executeUpdate(getInsertQuery(table, "103", "'Hard'", "30",
"50"));
        Controller.statement.executeUpdate(getInsertQuery(table, "104", "'Medium'", "20",
"30"));

        Controller.statement.executeUpdate(getInsertQuery(table, "105", "'Easy'", "10",
"85"));
        Controller.statement.executeUpdate(getInsertQuery(table, "106", "'Hard'", "30",
"68"));
        Controller.statement.executeUpdate(getInsertQuery(table, "107", "'Easy'", "10",
"77"));
        Controller.statement.executeUpdate(getInsertQuery(table, "108", "'Medium'", "20",
"50"));
        Controller.statement.executeUpdate(getInsertQuery(table, "109", "'Medium'", "20",
"30"));
    } catch (SQLException E) {
        System.out.println("Error while adding to public task table...");
        E.printStackTrace();
    }
}

public static void addPublicCodingTask() {
    String table = "PUBLIC_CODING_TASK";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "100",
        "'Given nums = [2, 7, 11, 15], target = 9, Because nums[0] + nums[1] = 2 + 7
= 9, return [0, 1].'",
        "' class Solution { public String solution(int solution) { //solution }
}'", "'Hint 0'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "101",
        "'Input: (2 -> 4 -> 3) + (5 -> 6 -> 4) Output: 7 -> 0 -> 8 Explanation: 342 +
465 = 807.'",
        "' class Solution { public String solution(int solution) { //solution }
}'", "'Hint 1'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "102",
        "'Input: abcabcbb Output: 3 Explanation: The answer is abc, with the length
of 3.'",
        "' class Solution { public String solution(int solution) { //solution }
}'", "'Hint 2'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "103",
        "'nums1 = [1, 3] nums2 = [2] The median is 2.0'",
        "' class Solution { public String solution(int solution) { //solution }
}'", "'Hint 3'"));
    }
}

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "104",
            "'Input: babad Output: bab Note: aba is also a valid answer.'",
            "'      class Solution { public String solution(int solution) { //solution }
}'" , "'Hint 4'"));

    } catch (SQLException E) {
        System.out.println("Error while adding to public coding task table...");
        E.printStackTrace();
    }
}

public static void addPublicNonCodingTask() {
    String table = "PUBLIC_NON_CODING_TASK";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "105"));
        Controller.statement.executeUpdate(getInsertQuery(table, "106"));
        Controller.statement.executeUpdate(getInsertQuery(table, "107"));
        Controller.statement.executeUpdate(getInsertQuery(table, "108"));
        Controller.statement.executeUpdate(getInsertQuery(table, "109"));
    } catch (SQLException E) {
        System.out.println("Error while adding to public non coding task table...");
        E.printStackTrace();
    }
}

public static void addInterviewTask() {
    String table = "INTERVIEW_TASK";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "110", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "111", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "112", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "113", "1"));
    } catch (SQLException E) {
        System.out.println("Error while adding to interview task table...");
        E.printStackTrace();
    }
}

public static void addNonCodingInterviewTask() {
    String table = "NON_CODING_INTERVIEW_TASK";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "110"));
        Controller.statement.executeUpdate(getInsertQuery(table, "111"));
    } catch (SQLException E) {
        System.out.println("Error while adding to non coding interview task table...");
        E.printStackTrace();
    }
}

public static void addCodingInterviewTask() {
    String table = "CODING_INTERVIEW_TASK";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "112",
            "'Input: s = PAYPALISHIRING, numRows = 3 Output: PAHNAPLSIIGYIR'" ,

```

```

        "" class Solution { public String solution(int solution) { //solution }
    }", "'HINT 1'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "113", "'Input: 123 Output:
321'",
        "" class Solution { public String solution(int solution) { //solution }
    }", "'HINT 2'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to non coding interview task table...");
            E.printStackTrace();
        }
    }

    public static void addContestTask() {
        String table = "CONTEST_TASK";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "114", "2", "'Input: 121
Output: true'",
        "" class Solution { public String solution(int solution) { //solution }
    }", "'HINT 1'", "1'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "115", "2",
        "''. Matches any single character. * Matches zero or more of the preceding
element.'",
        "" class Solution { public String solution(int solution) { //solution }
    }", "'HINT 2'", "1'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "116", "2",
        "'Input: [1,8,6,2,5,4,8,3,7] Output: 49'",
        "" class Solution { public String solution(int solution) { //solution }
    }", "'HINT 3'", "1'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "117", "2", "'Input: 3
Output: III'",
        "" class Solution { public String solution(int solution) { //solution }
    }", "'HINT 4'", "1'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to interview task table...");
            E.printStackTrace();
        }
    }

    public static void addCodingBetTask() {
        String table = "CODING_BET_TASK";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "118", "'Input: 4 Output:
1211'",
        "" class Solution { public String solution(int solution) { //solution }
    }", "'HINT 1'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "119", "'Input: [1,3,5,6], 5
Output: 2'",
        "" class Solution { public String solution(int solution) { //solution }
    }", "'HINT 2'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to interview task table...");
            E.printStackTrace();
        }
    }

    public static void addSubmission() {
        String table = "SUBMISSION";

```



```

try {
    // Same user same public coding 4 submission
    Controller.statement.executeUpdate(getInsertQuery(table, "1000", "3", "100",
        "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

    Controller.statement.executeUpdate(getInsertQuery(table, "1001", "3", "100",
        "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

    Controller.statement.executeUpdate(getInsertQuery(table, "1002", "3", "100",
        "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

    Controller.statement.executeUpdate(getInsertQuery(table, "1003", "3", "100",
        "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

    // two different users same public non coding
    Controller.statement.executeUpdate(getInsertQuery(table, "1004", "3", "105",
        "'First the Java is interpreted Language which run on JVM, but C++ will be
compiled into the machine code which run directly on cpu. So commonly, C++ native code is
faster. With JVM, java provide the memory management system which can reduce the error from
pointer..'"));

    Controller.statement.executeUpdate(getInsertQuery(table, "1005", "4", "105",
        "'First the Java is interpreted Language which run on JVM, but C++ will be
compiled into the machine code which run directly on cpu. So commonly, C++ native code is
faster. With JVM, java provide the memory management system which can reduce the error from
pointer..'"));

    // same user interview two different tasks non coding
    Controller.statement.executeUpdate(getInsertQuery(table, "1006", "3", "110",
        "'First the Java is interpreted Language which run on JVM, but C++ will be
compiled into the machine code which run directly on cpu. So commonly, C++ native code is
faster. With JVM, java provide the memory management system which can reduce the error from
pointer..'"));

    Controller.statement.executeUpdate(getInsertQuery(table, "1007", "3", "111",
        "'First the Java is interpreted Language which run on JVM, but C++ will be
compiled into the machine code which run directly on cpu. So commonly, C++ native code is
faster. With JVM, java provide the memory management system which can reduce the error from
pointer..'"));

    // same user interview two tasks coding
    Controller.statement.executeUpdate(getInsertQuery(table, "1008", "3", "112",
        "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "1009", "3", "113",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        // same user contest two tasks coding
        Controller.statement.executeUpdate(getInsertQuery(table, "1010", "3", "116",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        Controller.statement.executeUpdate(getInsertQuery(table, "1011", "3", "117",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        // two users coding bet
        Controller.statement.executeUpdate(getInsertQuery(table, "1012", "3", "118",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        Controller.statement.executeUpdate(getInsertQuery(table, "1013", "4", "118",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        } catch (SQLException E) {
            System.out.println("Error while adding to submission table...");
            E.printStackTrace();
        }
    }

    public static void addNonCodingSubmission() {
        String table = "NON_CODING_SUBMISSION";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "1004"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1005"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1006"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1007"));

        } catch (SQLException E) {
            System.out.println("Error while adding to non coding submission table...");
            E.printStackTrace();
        }
    }

    public static void addCodingSubmission() {
        String table = "CODING_SUBMISSION";

        try {

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "1000", "'C'", "'solved'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1001", "'C++'",
"'solved'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1002", "'JAVA'",
"'failed'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1003", "'PYTHON'",
"'solved'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1008", "'PYTHON'",
"'solved'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1009", "'JAVA'",
"'solved'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1010", "'C'", "'failed'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1011", "'C++'",
"'solved'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1012", "'PYTHON'",
"'solved'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1013", "'C++'",
"'solved'"));

    } catch (SQLException E) {
        System.out.println("Error while adding to coding submission table...");
        E.printStackTrace();
    }
}

public static void addPublicNonCodingSubmission() {
    String table = "PUBLIC_NON_CODING_SUBMISSION";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1004", "'2019/01/01'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1005", "'2019/01/02'"));

    } catch (SQLException E) {
        System.out.println("Error while adding to public non coding submission table...");
        E.printStackTrace();
    }
}

public static void addInterviewNonCodingSubmission() {
    String table = "INTERVIEW_NON_CODING_SUBMISSION";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1006"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1007"));

    } catch (SQLException E) {
        System.out.println("Error while adding to interview non coding submission table...");
        E.printStackTrace();
    }
}

public static void addInterviewCodingSubmission() {
    String table = "INTERVIEW_CODING_SUBMISSION";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1008"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1009"));

    } catch (SQLException E) {
        System.out.println("Error while adding to interview non coding submission table...");

```

```

        E.printStackTrace();
    }
}

public static void addPublicCodingSubmission() {
    String table = "PUBLIC_CODING_SUBMISSION";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1000", "'2019/02/01'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1001", "'2019/03/03'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1002", "'2018/08/11'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1003", "'2019/07/10'"));

    } catch (SQLException E) {
        System.out.println("Error while adding to public coding submission table...");
        E.printStackTrace();
    }
}

public static void addContestCodingSubmission() {
    String table = "CONTEST_CODING_SUBMISSION";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1010"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1011"));

    } catch (SQLException E) {
        System.out.println("Error while adding to contest coding submission table...");
        E.printStackTrace();
    }
}

public static void addCodingBetSubmission() {
    String table = "CODING_BET_SUBMISSION";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1012"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1013"));

    } catch (SQLException E) {
        System.out.println("Error while adding to coding bet submission table...");
        E.printStackTrace();
    }
}

public static void addCompanyInterviewTaskPrepare() {
    String table = "COMPANY_INTERVIEW_TASK_PREPARE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "110",
"'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "111",
"'2019/12/08'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "6", "112",
"'2019/12/09'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "6", "113",
"'2019/12/10'"));

    } catch (SQLException E) {

```

```

        System.out.println("Error while adding to company interview task prepare table...");
        E.printStackTrace();
    }
}

public static void addEditorPublicTaskPrepare() {
    String table = "EDITOR_PUBLIC_TASK_PREPARE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "100",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "101",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "102",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "103",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "104",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "105",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "106",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "107",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "108",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "109",
"2019/12/07'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to editor public task prepare table...");
        E.printStackTrace();
    }
}

public static void addEditorCodingContestPrepare() {
    String table = "EDITOR_CODING_CONTEST_PREPARE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "1", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "2", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "3", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "4", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "5", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "6", "'2019/12/07'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to editor contest prepare table...");
        E.printStackTrace();
    }
}

public static void addCompanyInterviewPrepare() {
    String table = "COMPANY_INTERVIEW_PREPARE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "2"));
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "3"));
        Controller.statement.executeUpdate(getInsertQuery(table, "6", "4"));
        Controller.statement.executeUpdate(getInsertQuery(table, "6", "5"));
    }
}

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "6", "6"));
    } catch (SQLException E) {
        System.out.println("Error while adding to company interview prepare table...");
        E.printStackTrace();
    }
}

public static void addUserContestParticipate() {
    String table = "USER_CONTEST_PARTICIPATE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "1", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "1", "2"));
    } catch (SQLException E) {
        System.out.println("Error while adding to user contest participate table...");
        E.printStackTrace();
    }
}

public static void addUserCodingBetParticipate() {
    String table = "USER_CODING_BET_PARTICIPATE";
    try {
        // userid - coding betid - place
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "1", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "2", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "3", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "3", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "2", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "4", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "5", "0"));
    } catch (SQLException E) {
        System.out.println("Error while adding to user coding bet participate table...");
        E.printStackTrace();
    }
}

public static void addUserInterviewPerform() {
    String table = "USER_INTERVIEW_PERFORM";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "1", "'2019/12/07'",
"waiting"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "2", "'2019/12/07'",
"accepted"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "3", "'2019/12/07'",
"rejected"));
    } catch (SQLException E) {
        System.out.println("Error while adding to user interview perform table...");
        E.printStackTrace();
    }
}

public static void addTestCase() {
    String table = "TEST_CASE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 1'", "'output
1'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 2'", "'output
2'"));
    }
}

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 3'", "'output
3'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 4'", "'output
4'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 5'", "'output
5'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to test case table...");
        E.printStackTrace();
    }
}

public static void addCodingBetTaskRelation() {
    String table = "CODING_BET_TASK_RELATION";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "118"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "118"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "118"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "119"));
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "119"));
    } catch (SQLException E) {
        System.out.println("Error while adding to coding bet task relation table...");
        E.printStackTrace();
    }
}

public static void addRateAnswer() {
    String table = "RATE_ANSWER";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "1004", "'Upvote'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "1004", "'Upvote'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "1005", "'Downvote'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "1005", "'Downvote'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to rate answer table...");
        E.printStackTrace();
    }
}
}

public class CreateProcedureQueryBuilder
{
    static String createPublicCodingProblemsByCategoryProcedure()
    {
        return "delimiter //" +
            "CREATE PROCEDURE selectAllPublicCodingTaskFromCategory " +
            "(@Category varchar(30)) " +
            "AS " +
            "BEGIN " +
            "SELECT * " +
            "FROM public_coding_task NATURAL JOIN public_task " +
            "NATURAL JOIN task " +
            "WHERE category = @Category " +
            "ORDER BY task_id ASC " +
            "END" +
            "delimiter ;";
    }
}

```

```

static String createPublicCodingProblemsByCategoryWithInputProcedure()
{
    return "delimiter //" +
        "CREATE PROCEDURE selectAllPublicCodingTaskFromCategoryWithSearch " +
        "(@category varchar(30), @search_input) " +
        "AS " +
        "BEGIN " +
        "SELECT * " +
        "FROM public_coding_task NATURAL JOIN public_task " +
        "NATURAL JOIN task " +
        "WHERE category = @category and title LIKE %search_input% " +
        "ORDER BY task_id ASC " +
        "END" +
        "delimiter ;";
}

static String createPublicNonCodingProblemsByCategoryProcedure()
{
    return "delimiter //" +
        "CREATE PROCEDURE selectAllPublicNonCodingFromCategory " +
        "(@category varchar(30)) " +
        "AS " +
        "BEGIN " +
        "SELECT * " +
        "FROM ( " +
        "public_non_coding_task NATURAL JOIN public_task " +
        "NATURAL JOIN task) as T " +
        "JOIN editor_public_task_prepare as E on E.task_id = T.task_id " +
        "WHERE category = @category " +
        "ORDER BY task_id ASC " +
        "END" +
        "delimiter ;";
}

static String createAccountInformationOfUserProcedure()
{
    return "delimiter //" +
        "create procedure getUserInformation " +
        "(in email_in varchar(30))" +
        "BEGIN " +
        "select count(*) FROM t;" +
        "END//" +
        "delimiter ;";
//    return "delimiter //" +
//        "CREATE PROCEDURE getUserInformation " +
//        "(IN email_in varchar(30), IN password_in varchar(10), IN account_type
varchar(10)) " +
//        "BEGIN " +
//        "IF account_type = 'user' " +
//        "BEGIN " +
//        "SELECT * " +
//        "FROM USER as U JOIN ACCOUNT as A on U.user_id = A.account_id " +
//        "WHERE email = email_in and password = password_in " +
//        "END " +
//        "ELSE IF account_type = 'editor' " +
//        "BEGIN " +

```



```

//          "SELECT * " +
//          "FROM EDITOR E JOIN ACCOUNT A on E.editor_id = A.account_id " +
//          "WHERE email = email_in and password = password_in " +
//          "END " +
//          "ELSE IF account_type = 'company' " +
//          "BEGIN " +
//          "SELECT * " +
//          "FROM " +
//          "COMPANY C JOIN ACCOUNT A on C.company_id = A.account_id " +
//          "WHERE email = email_in and password = password_in " +
//          "END " +
//          "END//" +
//          "delimiter ;";
}

static String createRankingInfoProcedure()
{
    return "delimiter //" +
        "CREATE PROCEDURE showRankingOfContest (@category varchar(30)) " +
        "AS " +
        "BEGIN " +
        "WITH user_points as SELECT S.user_id ,SUM(points_given) as total " +
        "FROM (contest_task NATURAL JOIN task) as T " +
        "JOIN (submission NATURAL JOIN user) as S ON T.task_id = S.task_id " +
        "WHERE contest_id = <contest_id> and status = 'solved' " +
        "GROUP BY S.user_id " +
        "ORDER BY total DESC, " +
        "SELECT distinct name, total " +
        "FROM user_points NATURAL JOIN user " +
        "END" +
        "delimiter ;";
}

static String dropPublicCodingProblemsByCategoryProcedure()
{
    return "DROP PROCEDURE selectAllPublicCodingTaskFromCategory;";
}

static String dropPublicCodingProblemsByCategorywithInputProcedure()
{
    return "DROP PROCEDURE selectAllPublicCodingTaskFromCategoryWithSearch;";
}

static String dropPublicNonCodingProblemsByCategoryProcedure()
{
    return "DROP PROCEDURE selectAllPublicNonCodingFromCategory;";
}

static String dropAccountInformationOfUserProcedure()
{
    return "DROP PROCEDURE getUserInformation;";
}

static String dropRankingInfoProcedure()
{
    return "DROP PROCEDURE showRankingOfContest;";
}

```

```
}
```

```
public class CreateTablesQueryBuilder {
```

```
    static String getCreateAccountTable()
```

```
    {
```

```
        return "CREATE TABLE ACCOUNT(\n" +  
            "account_id VARCHAR(13) PRIMARY KEY,\n" +  
            "name VARCHAR(50) NOT NULL,\n" +  
            "email VARCHAR(50) NOT NULL UNIQUE,\n" +  
            "password VARCHAR(50) NOT NULL,\n" +  
            "info VARCHAR(500)\n" +  
            ")ENGINE=InnoDB;\n";
```

```
    }
```

```
    static String getCreateUserTable()
```

```
    {
```

```
        return "CREATE TABLE USER(\n" +  
            "user_id VARCHAR(13) PRIMARY KEY,\n" +  
            "score INT,\n" +  
            "tech_coin INT,\n" +  
            "FOREIGN KEY(user_id) REFERENCES ACCOUNT (account_id)\n" +  
            "\tON DELETE CASCADE\n" +  
            "\tON UPDATE CASCADE\n" +  
            ")ENGINE=InnoDB;\n";
```

```
    }
```

```
    static String getCreateEditorTable()
```

```
    {
```

```
        return "CREATE TABLE EDITOR(\n" +  
            "editor_id VARCHAR(13) PRIMARY KEY,\n" +  
            "ranky INT,\n" +  
            "FOREIGN KEY (editor_id) REFERENCES ACCOUNT(account_id)\n" +  
            "\tON DELETE CASCADE\n" +  
            "\tON UPDATE CASCADE\n" +  
            ")ENGINE=InnoDB;\n";
```

```
    }
```

```
    static String getCreateCompanyTable()
```

```
    {
```

```
        return "CREATE TABLE COMPANY(\n" +  
            "company_id VARCHAR(13) PRIMARY KEY,\n" +  
            "website VARCHAR(50) NOT NULL,\n" +  
            "FOREIGN KEY (company_id) REFERENCES ACCOUNT(account_id)\n" +  
            "\tON DELETE CASCADE\n" +  
            "\tON UPDATE CASCADE\n" +  
            ")ENGINE=InnoDB;\n";
```

```
    }
```

```
    static String getCreateJobTable()
```

```
    {
```

```
        return "CREATE TABLE JOB(\n" +  
            "company_id VARCHAR(13) NOT NULL,\n" +  
            "name VARCHAR(50) NOT NULL,\n" +  
            "requirements VARCHAR(50),\n" +  
            "FOREIGN KEY (company_id) REFERENCES COMPANY (company_id)\n" +  
            "\tON DELETE CASCADE\n" +  
            "\tON UPDATE CASCADE,\n" +  
            "PRIMARY KEY(company_id, name)\n" +  
            ")ENGINE=InnoDB;\n";
```

```

}
static String getCreateInterviewTable()
{
    return "CREATE TABLE INTERVIEW(\n" +
        "interview_id VARCHAR(13) PRIMARY KEY,\n" +
        "job_type VARCHAR(50) NOT NULL,\n" +
        "description VARCHAR(50),\n" +
        "duration INT NOT NULL\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreateCodingContestTable()
{
    return "CREATE TABLE CONTEST(\n" +
        "contest_id VARCHAR(13) PRIMARY KEY,\n" +
        "name VARCHAR(50) NOT NULL UNIQUE,\n" +
        "start_date TIMESTAMP NOT NULL,\n" +
        "end_date TIMESTAMP NOT NULL,\n" +
        "reward1 INT NOT NULL,\n" +
        "reward2 INT NOT NULL,\n" +
        "reward3 INT NOT NULL\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreateCodingBetTable()
{
    return "CREATE TABLE CODING_BET(\n" +
        "coding_bet_id VARCHAR(13) PRIMARY KEY,\n" +
        "bet_amount INT NOT NULL,\n" +
        "status VARCHAR(20),\n" +
        "owner_id VARCHAR(13) NOT NULL,\n" +
        "FOREIGN KEY (owner_id) REFERENCES USER (user_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreateTaskTable()
{
    return "CREATE TABLE TASK(\n" +
        "task_id VARCHAR(13) PRIMARY KEY,\n" +
        "title VARCHAR(50) NOT NULL UNIQUE,\n" +
        "question VARCHAR(500) NOT NULL UNIQUE,\n" +
        "category VARCHAR(20) NOT NULL,\n" +
        "info VARCHAR(500)\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreatePublicTaskTable()
{
    return "CREATE TABLE PUBLIC_TASK(\n" +
        "task_id VARCHAR(13) PRIMARY KEY,\n" +
        "difficulty VARCHAR(10) NOT NULL,\n" +
        "points_given INT NOT NULL,\n" +
        "success_rate INT,\n" +
        "FOREIGN KEY (task_id) REFERENCES TASK (task_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreatePublicNonCodingTaskTable()

```

```

{
    return "CREATE TABLE PUBLIC_NON_CODING_TASK(\n" +
        "task_id VARCHAR(13) PRIMARY KEY,\n" +
        "FOREIGN KEY (task_id) REFERENCES TASK (task_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreatePublicCodingTaskTable()
{
    return "CREATE TABLE PUBLIC_CODING_TASK(\n" +
        "task_id VARCHAR(13) PRIMARY KEY,\n" +
        "example VARCHAR(500) NOT NULL UNIQUE,\n" +
        "solution VARCHAR(500) NOT NULL,\n" +
        "hint VARCHAR(500),\n" +
        "FOREIGN KEY (task_id) REFERENCES TASK (task_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreateInterviewTaskTable()
{
    return "CREATE TABLE INTERVIEW_TASK(\n" +
        "task_id VARCHAR(13) PRIMARY KEY,\n" +
        "Interview_id VARCHAR(13) NOT NULL,\n" +
        "FOREIGN KEY (task_id) REFERENCES TASK (task_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE,\n" +
        "FOREIGN KEY (interview_id) REFERENCES INTERVIEW (interview_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreateNonCodingInterviewTaskTable()
{
    return "CREATE TABLE NON_CODING_INTERVIEW_TASK(\n" +
        "task_id VARCHAR(13) PRIMARY KEY,\n" +
        "FOREIGN KEY (task_id) REFERENCES INTERVIEW_TASK (task_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreateCodingInterviewTaskTable()
{
    return "CREATE TABLE CODING_INTERVIEW_TASK(\n" +
        "task_id VARCHAR(13) PRIMARY KEY,\n" +
        "example VARCHAR(500) NOT NULL UNIQUE,\n" +
        "solution VARCHAR(500) NOT NULL,\n" +
        "hint VARCHAR(500),\n" +
        "FOREIGN KEY (task_id) REFERENCES INTERVIEW_TASK (task_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
}
static String getCreateContestTaskTable()
{
    return "CREATE TABLE CONTEST_TASK(\n" +

```

```

        "task_id VARCHAR(13) PRIMARY KEY,\n" +
        "contest_point INT NOT NULL,\n" +
        "example VARCHAR(500) NOT NULL UNIQUE,\n" +
        "solution VARCHAR(500) NOT NULL,\n" +
        "hint VARCHAR(500),\n" +
        "contest_id VARCHAR(13) NOT NULL,\n" +
        "FOREIGN KEY (task_id) REFERENCES TASK (task_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE,\n" +
        "FOREIGN KEY (contest_id) REFERENCES CONTEST (contest_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        "\n" +
        ")ENGINE=InnoDB;\n";
    }
    static String getCreateCodingBetTaskTable()
    {
        return "CREATE TABLE CODING_BET_TASK(\n" +
            "task_id VARCHAR(13) PRIMARY KEY,\n" +
            "example VARCHAR(500) NOT NULL UNIQUE,\n" +
            "solution VARCHAR(500) NOT NULL,\n" +
            "hint VARCHAR(500),\n" +
            "FOREIGN KEY (task_id) REFERENCES TASK (task_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateSubmissionTable()
    {
        return "CREATE TABLE SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +
            "user_id VARCHAR(13) NOT NULL,\n" +
            "task_id VARCHAR(13) NOT NULL,\n" +
            "answer VARCHAR(500) NOT NULL,\n" +
            "FOREIGN KEY (user_id) REFERENCES USER (user_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (task_id) REFERENCES TASK (task_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateNonCodingSubmissionTable()
    {
        return "CREATE TABLE NON_CODING_SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +
            "FOREIGN KEY (submission_id) REFERENCES SUBMISSION (submission_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateCodingSubmissionTable()
    {
        return "CREATE TABLE CODING_SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +
            "prog_lang VARCHAR(20) NOT NULL,\n" +
            "status VARCHAR(20) NOT NULL,\n" +

```

```

        "FOREIGN KEY (submission_id) REFERENCES SUBMISSION (submission_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
    }
    static String getCreatePublicNonCodingSubmissionTable()
    {
        return "CREATE TABLE PUBLIC_NON_CODING_SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +
            "date TIMESTAMP NOT NULL,\n" +
            "FOREIGN KEY (submission_id) REFERENCES NON_CODING_SUBMISSION (submission_id)\n"
+
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreatePublicCodingSubmissionTable()
    {
        return "CREATE TABLE PUBLIC_CODING_SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +
            "date TIMESTAMP NOT NULL,\n" +
            "FOREIGN KEY (submission_id) REFERENCES CODING_SUBMISSION (submission_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateInterviewNonCodingSubmissionTable()
    {
        return "CREATE TABLE INTERVIEW_NON_CODING_SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +
            "FOREIGN KEY (submission_id) REFERENCES NON_CODING_SUBMISSION (submission_id)\n"
+
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateInterviewCodingSubmissionTable()
    {
        return "CREATE TABLE INTERVIEW_CODING_SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +
            "FOREIGN KEY (submission_id) REFERENCES CODING_SUBMISSION (submission_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateContestCodingSubmissionTable()
    {
        return "CREATE TABLE CONTEST_CODING_SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +
            "FOREIGN KEY (submission_id) REFERENCES CODING_SUBMISSION (submission_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateCodingBetSubmissionTable()
    {
        return "CREATE TABLE CODING_BET_SUBMISSION(\n" +
            "submission_id VARCHAR(13) PRIMARY KEY,\n" +

```

```

        "FOREIGN KEY (submission_id) REFERENCES CODING_SUBMISSION (submission_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE\n" +
        ")ENGINE=InnoDB;\n";
    }
    static String getCreateCompanyInterviewTaskPrepareTable()
    {
        return "CREATE TABLE COMPANY_INTERVIEW_TASK_PREPARE(\n" +
            "company_id VARCHAR(13) NOT NULL,\n" +
            "task_id VARCHAR(13) NOT NULL,\n" +
            "date TIMESTAMP NOT NULL,\n" +
            "FOREIGN KEY (company_id) REFERENCES COMPANY (company_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (task_id) REFERENCES INTERVIEW_TASK (task_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "PRIMARY KEY(company_id, task_id)\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateEditorPublicTaskPrepareTable()
    {
        return "CREATE TABLE EDITOR_PUBLIC_TASK_PREPARE(\n" +
            "editor_id VARCHAR(13) NOT NULL,\n" +
            "task_id VARCHAR(13) NOT NULL,\n" +
            "date TIMESTAMP NOT NULL,\n" +
            "FOREIGN KEY (editor_id) REFERENCES EDITOR (editor_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (task_id) REFERENCES PUBLIC_TASK (task_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "PRIMARY KEY(editor_id, task_id)\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateEditorCodingContestPrepareTable()
    {
        return "CREATE TABLE EDITOR_CODING_CONTEST_PREPARE(\n" +
            "editor_id VARCHAR(13) NOT NULL,\n" +
            "contest_id VARCHAR(13) NOT NULL,\n" +
            "date TIMESTAMP NOT NULL,\n" +
            "FOREIGN KEY (editor_id) REFERENCES EDITOR (editor_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (contest_id) REFERENCES CONTEST (contest_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "PRIMARY KEY(editor_id, contest_id)\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateCompanyInterviewPrepareTable()
    {
        return "CREATE TABLE COMPANY_INTERVIEW_PREPARE(\n" +
            "company_id VARCHAR(13) NOT NULL,\n" +
            "interview_id VARCHAR(13) NOT NULL,\n" +
            "FOREIGN KEY (company_id) REFERENCES COMPANY (company_id)\n" +
            "\tON DELETE CASCADE\n" +

```

```

        "\tON UPDATE CASCADE,\n" +
        "FOREIGN KEY (interview_id) REFERENCES INTERVIEW (interview_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE,\n" +
        "PRIMARY KEY(company_id, interview_id)\n" +
        ")ENGINE=InnoDB;\n";
    }
    static String getCreateUserContestParticipateTable()
    {
        return "CREATE TABLE USER_CONTEST_PARTICIPATE(\n" +
            "user_id VARCHAR(13) NOT NULL,\n" +
            "contest_id VARCHAR(13) NOT NULL,\n" +
            "place INT,\n" +
            "FOREIGN KEY (user_id) REFERENCES USER (user_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (contest_id) REFERENCES CONTEST (contest_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "PRIMARY KEY(user_id, contest_id)\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateUserCodingBetParticipateTable()
    {
        return "CREATE TABLE USER_CODING_BET_PARTICIPATE(\n" +
            "user_id VARCHAR(13) NOT NULL,\n" +
            "coding_bet_id VARCHAR(13) NOT NULL,\n" +
            "place INT,\n" +
            "FOREIGN KEY (user_id) REFERENCES USER (user_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (coding_bet_id) REFERENCES CODING_BET (coding_bet_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "PRIMARY KEY(user_id, coding_bet_id)\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateUserInterviewPerformTable()
    {
        return "CREATE TABLE USER_INTERVIEW_PERFORM(\n" +
            "user_id VARCHAR(13) NOT NULL,\n" +
            "interview_id VARCHAR(13) NOT NULL,\n" +
            "date TIMESTAMP NOT NULL,\n" +
            "result VARCHAR(20),\n" +
            "FOREIGN KEY (user_id) REFERENCES USER (user_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (interview_id) REFERENCES INTERVIEW (interview_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "PRIMARY KEY(user_id, interview_id)\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateTestCaseTable()
    {
        return "CREATE TABLE TEST_CASE(\n" +
            "task_id VARCHAR(13) NOT NULL,\n" +

```



```

        "sample_input VARCHAR(200) NOT NULL,\n" +
        "sample_output VARCHAR(200) NOT NULL,\n" +
        "FOREIGN KEY (task_id) REFERENCES TASK (task_id)\n" +
        "\tON DELETE CASCADE\n" +
        "\tON UPDATE CASCADE,\n" +
        "PRIMARY KEY(task_id, sample_input, sample_output)\n" +
        ")ENGINE=InnoDB;\n";
    }
    static String getCreateCodingBetTaskRelationTable()
    {
        return "CREATE TABLE CODING_BET_TASK_RELATION(\n" +
            "coding_bet_id VARCHAR(13) NOT NULL,\n" +
            "task_id VARCHAR(13) NOT NULL,\n" +
            "FOREIGN KEY (coding_bet_id) REFERENCES CODING_BET (coding_bet_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (task_id) REFERENCES CODING_BET_TASK (task_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "PRIMARY KEY(coding_bet_id)\n" +
            ")ENGINE=InnoDB;\n";
    }
    static String getCreateRateAnswerTable()
    {
        return "CREATE TABLE RATE_ANSWER(\n" +
            "user_id VARCHAR(13) NOT NULL,\n" +
            "answer_id VARCHAR(13) NOT NULL,\n" +
            "rate_answer VARCHAR(10) NOT NULL,\n" +
            "FOREIGN KEY (user_id) REFERENCES USER (user_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "FOREIGN KEY (answer_id) \n" +
            "REFERENCES PUBLIC_NON_CODING_SUBMISSION(submission_id)\n" +
            "\tON DELETE CASCADE\n" +
            "\tON UPDATE CASCADE,\n" +
            "PRIMARY KEY(user_id, answer_id)\n" +
            "\n" +
            ")ENGINE=InnoDB;\n";
    }
}

}

public class CreateViewsQueryBuilder{

    static String createPublicAccountView(){
        return "CREATE VIEW account_public(id,name,info) as " +
            "SELECT id, name, info " +
            "FROM account;";
    }

    static String createUserSubmissionDetailView(){
        return "CREATE VIEW user_submission_detail " +

```

```

        "(submission_id, user_id, task_id, answer, prog_lang, status, date) " +
        "SELECT * " +
        "FROM public_coding_submission NATURAL JOIN coding_submission " +
        "NATURAL JOIN submission;";
    }

    static String createUserSubmissionsView(){
        return "CREATE VIEW user_submission " +
        "(submission_id, user_id, task_id, prog_lang, status, date) as " +
        "SELECT submission_id, user_id, task_id, prog_lang, status, date " +
        "FROM user_submission_detail;";
    }
}

public class DropTablesQueryBuilder {

    static String getDropAccountTable()
    {
        return "DROP TABLE ACCOUNT";
    }
    static String getDropUserTable()
    {
        return "DROP TABLE USER";
    }
    static String getDropEditorTable()
    {
        return "DROP TABLE EDITOR";
    }
    static String getDropCompanyTable()
    {
        return "DROP TABLE COMPANY";
    }
    static String getDropJobTable()
    {
        return "DROP TABLE JOB";
    }
    static String getDropInterviewTable()
    {
        return "DROP TABLE INTERVIEW";
    }
    static String getDropCodingContestTable()
    {
        return "DROP TABLE CONTEST";
    }
    static String getDropCodingBetTable()
    {
        return "DROP TABLE CODING_BET";
    }
    static String getDropTaskTable()
    {
        return "DROP TABLE TASK";
    }
    static String getDropPublicTaskTable()
    {
        return "DROP TABLE PUBLIC_TASK";
    }
    static String getDropPublicNonCodingTaskTable()

```

```

{
    return "DROP TABLE PUBLIC_NON_CODING_TASK";
}
static String getDropPublicCodingTaskTable()
{
    return "DROP TABLE PUBLIC_CODING_TASK";
}
static String getDropInterviewTaskTable()
{
    return "DROP TABLE INTERVIEW_TASK";
}
static String getDropNonCodingInterviewTaskTable()
{
    return "DROP TABLE NON_CODING_INTERVIEW_TASK";
}
static String getDropCodingInterviewTaskTable()
{
    return "DROP TABLE CODING_INTERVIEW_TASK";
}
static String getDropContestTaskTable()
{
    return "DROP TABLE CONTEST_TASK";
}
static String getDropCodingBetTaskTable()
{
    return "DROP TABLE CODING_BET_TASK";
}
static String getDropSubmissionTable()
{
    return "DROP TABLE SUBMISSION";
}
static String getDropNonCodingSubmissionTable()
{
    return "DROP TABLE NON_CODING_SUBMISSION";
}
static String getDropCodingSubmissionTable()
{
    return "DROP TABLE CODING_SUBMISSION";
}
static String getDropPublicNonCodingSubmissionTable()
{
    return "DROP TABLE PUBLIC_NON_CODING_SUBMISSION";
}
static String getDropPublicCodingSubmissionTable()
{
    return "DROP TABLE PUBLIC_CODING_SUBMISSION";
}
static String getDropInterviewNonCodingSubmissionTable()
{
    return "DROP TABLE INTERVIEW_NON_CODING_SUBMISSION";
}
static String getDropInterviewCodingSubmissionTable()
{
    return "DROP TABLE INTERVIEW_CODING_SUBMISSION";
}
static String getDropContestCodingSubmissionTable()
{
    return "DROP TABLE CONTEST_CODING_SUBMISSION";
}

```

```

    }
    static String getDropCodingBetSubmissionTable()
    {
        return "DROP TABLE CODING_BET_SUBMISSION";
    }
    static String getDropCompanyInterviewTaskPrepareTable()
    {
        return "DROP TABLE COMPANY_INTERVIEW_TASK_PREPARE";
    }
    static String getDropEditorPublicTaskPrepareTable()
    {
        return "DROP TABLE EDITOR_PUBLIC_TASK_PREPARE";
    }
    static String getDropEditorCodingTaskPrepareTable()
    {
        return "DROP TABLE EDITOR_CODING_CONTEST_PREPARE";
    }
    static String getDropCompanyInterviewPrepareTable()
    {
        return "DROP TABLE COMPANY_INTERVIEW_PREPARE";
    }
    static String getDropUserContestParticipateTable()
    {
        return "DROP TABLE USER_CONTEST_PARTICIPATE";
    }
    static String getDropUserCodingBetParticipateTable()
    {
        return "DROP TABLE USER_CODING_BET_PARTICIPATE";
    }
    static String getDropUserInterviewPerformTable()
    {
        return "DROP TABLE USER_INTERVIEW_PERFORM";
    }
    static String getDropTestCaseTable()
    {
        return "DROP TABLE TEST_CASE";
    }
    static String getDropCodingBetTaskRelationTable()
    {
        return "DROP TABLE CODING_BET_TASK_RELATION";
    }
    static String getDropRateAnswerTable()
    {
        return "DROP TABLE RATE_ANSWER";
    }
}

import java.util.*;
import java.sql.SQLException;

public class InitTablesQueryBuilder {

    private static String getInsertQuery(String table, String... param) {
        StringBuilder builder = new StringBuilder();
        builder.append("INSERT INTO ");
        builder.append(table);
        builder.append(" VALUES(");
    }

```

```

        for (int i = 0; i < param.length - 1; i++) {
            builder.append(param[i]);
            builder.append(",");
        }
        builder.append(param[param.length - 1]);
        builder.append(");");
        return builder.toString();
    }

    public static void addAccount() {
        try {
            Controller.statement.executeUpdate(
                getInsertQuery("ACCOUNT", "1", "'AHMET'", "'ahmet@gmail.com'", "'a'", "'20
years experience'"));
            Controller.statement.executeUpdate(
                getInsertQuery("ACCOUNT", "2", "'KAAN'", "'kaan@gmail.com'", "'a'", "'2 year
experience'"));
            Controller.statement.executeUpdate(
                getInsertQuery("ACCOUNT", "3", "'MUSTAFA'", "'mustafa@gmail.com'", "'a'",
"'student'"));
            Controller.statement
                .executeUpdate(getInsertQuery("ACCOUNT", "4", "'DENIZ'", "'deniz@gmail.com'",
"'a'", "'student'"));
            Controller.statement.executeUpdate(
                getInsertQuery("ACCOUNT", "5", "'Google'", "'google@gmail.com'", "'a'",
"'Tech Giant'"));
            Controller.statement.executeUpdate(getInsertQuery("ACCOUNT", "6", "'Microsoft'",
"'microsoft@gmail.com'",
"'a'", "'Looking for Employees'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to account table...");
            E.printStackTrace();
        }
    }

    public static void addEditor() {
        try {
            Controller.statement.executeUpdate(getInsertQuery("EDITOR", "1", "1"));
            Controller.statement.executeUpdate(getInsertQuery("EDITOR", "2", "2"));
        } catch (SQLException E) {
            System.out.println("Error while adding editor table...");
            E.printStackTrace();
        }
    }

    public static void addUser() {
        try {
            Controller.statement.executeUpdate(getInsertQuery("USER", "3", "0", "100"));
            Controller.statement.executeUpdate(getInsertQuery("USER", "4", "0", "100"));
        } catch (SQLException E) {
            System.out.println("Error while adding to user table...");
            E.printStackTrace();
        }
    }

    public static void addCompany() {

```

```

        try {
            Controller.statement.executeUpdate(getInsertQuery("COMPANY", "5", "'google.com'"));
            Controller.statement.executeUpdate(getInsertQuery("COMPANY", "6",
"'microsoft.com'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to company table...");
            E.printStackTrace();
        }
    }

    public static void addJob() {
        try {
            Controller.statement.executeUpdate(
                getInsertQuery("JOB", "5", "'game developer'", "'Hyper-casual genre game
developer.'"));
            Controller.statement
                .executeUpdate(getInsertQuery("JOB", "5", "'junior game developer'", "'Unity
Game developer.'"));
            Controller.statement
                .executeUpdate(getInsertQuery("JOB", "6", "'intern software developer'", "'3
months internship.'"));
            Controller.statement.executeUpdate(
                getInsertQuery("JOB", "6", "'senior software developer'", "'Machine learning
expert.'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to job table...");
            E.printStackTrace();
        }
    }

    public static void addInterview() {
        String tableName = "INTERVIEW";
        try {
            Controller.statement
                .executeUpdate(getInsertQuery(tableName, "1", "'internship'", "'Game
Developer'", "30"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "2", "'internship'",
"'Data Miner'", "30"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "3", "'full time'", "'AI
Expert'", "30"));
            Controller.statement
                .executeUpdate(getInsertQuery(tableName, "4", "'full time'", "'Full Stack
Developer'", "30"));
            Controller.statement
                .executeUpdate(getInsertQuery(tableName, "5", "'part time'", "'Python
Developer'", "30"));
            Controller.statement
                .executeUpdate(getInsertQuery(tableName, "6", "'part time'", "'Unity
Developer.'", "30"));
        } catch (SQLException E) {
            System.out.println("Error while adding to interview table...");
            E.printStackTrace();
        }
    }

    public static void addCodingContest() {
        String tableName = "CONTEST";

```

```

        try {
            Controller.statement.executeUpdate(getInsertQuery(tableName, "1", "'Weekly Contest
1'",
                "'2019-01-01 00:00:00'", "'2020-01-02 00:00:00'", "100", "50", "10"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "2", "'Weekly Contest
2'",
                "'2019-01-08 00:00:00'", "'2020-01-09 00:00:00'", "100", "50", "10"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "3", "'Weekly Contest
3'",
                "'2019-01-15 00:00:00'", "'2020-01-16 00:00:00'", "100", "50", "10"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "4", "'Weekly Contest
4'",
                "'2019-01-22 00:00:00'", "'2020-01-23 00:00:00'", "100", "50", "10"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "5", "'Weekly Contest
5'",
                "'2019-01-29 00:00:00'", "'2020-01-30 00:00:00'", "100", "50", "10"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "6", "'Weekly Contest
6'",
                "'2019-02-06 00:00:00'", "'2020-02-07 00:00:00'", "100", "50", "10"));
        } catch (SQLException E) {
            System.out.println("Error while adding to coding contest table...");
            E.printStackTrace();
        }
    }

    public static void addCodingBet() {
        String tableName = "CODING_BET";
        try {
            Controller.statement.executeUpdate(getInsertQuery(tableName, "1", "100", "'open'",
"'3'"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "2", "200", "'ongoing'",
"3"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "3", "300", "'closed'",
"3"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "4", "400", "'open'",
"4"));
            Controller.statement.executeUpdate(getInsertQuery(tableName, "5", "500", "'open'",
"4"));

        } catch (SQLException E) {
            System.out.println("Error while adding to coding bet table...");
            E.printStackTrace();
        }
    }

    public static void addTask() {
        String table = "TASK";
        try {
            // Coding
            Controller.statement.executeUpdate(getInsertQuery(table, "100", "'Two Sum'",
"'Question 0'", "'data'",
                "'Given an array of integers, return indices of the two numbers such that
they add up to a specific target. You may assume that each input would have exactly one solution,
and you may not use the same element twice.'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "101", "'Add Two Numbers'",
"'Question 1'",
                "'strings'",

```

```

        ""You are given two non-empty linked lists representing two non-negative
        integers. The digits are stored in reverse order and each of their nodes contain a single digit.
        Add the two numbers and return it as a linked list. You may assume the two numbers do not contain
        any leading zero, except the number 0 itself.'");
        Controller.statement.executeUpdate(getInsertQuery(table, "102",
            ""Longest Substring Without Repeating Characters'", ""Question 2'",
            ""algorithms'",
            ""Given a string, find the length of the longest substring without repeating
            characters.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "103", ""Median of Two
        Sorted Arrays'",
            ""Question 3'", ""databases'",
            ""There are two sorted arrays nums1 and nums2 of size m and n respectively.
            Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).
            You may assume nums1 and nums2 cannot be both empty.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "104", ""Longest Palindromic
        Substring'",
            ""Question 4'", ""os'",
            ""Given a string s, find the longest palindromic substring in s. You may
            assume that the maximum length of s is 1000.'"));
        // //////////
        Controller.statement.executeUpdate(getInsertQuery(table, "105", ""What are the main
        features of OOP?'"',
            ""Question 5'", ""oop'", ""OOP'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "106", ""Parking Lot Design
        Using OO Design'",
            ""Question 6'", ""data'", ""OOP'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "107",
            ""What is the difference between JAVA and c++ ?'", ""Question 7'",
            ""strings'", ""JAVA'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "108", ""5 flavors of
        singleton!'",
            ""Design Patterns'", ""algorithms'", ""Singleton'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "109", ""Social Network Most
        active users'",
            ""Question 9'", ""databases'", ""Social Network'"));
        //////////
        Controller.statement
            .executeUpdate(getInsertQuery(table, "110", ""Title 10'", ""Question 10'",
            ""os'", ""Info 10'"));
        Controller.statement
            .executeUpdate(getInsertQuery(table, "111", ""Title 11'", ""Question 11'",
            ""oop'", ""Info 11'"));
        //////////
        Controller.statement.executeUpdate(getInsertQuery(table, "112", ""ZigZag
        Conversion'", ""Question 12'",
            ""data'",
            ""The string PAYPALISHIRING is written in a zigzag pattern on a given number
            of rows like this: (you may want to display this pattern in a fixed font for better legibility) P
            A H N A P L S I I G Y I R And then read line by line: PAHNAPLSIIGYIR Write the code that will
            take a string and make this conversion given a number of rows:'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "113", ""Reverse Integer'",
            ""Question 13'",
            ""strings'", ""Given a 32-bit signed integer, reverse digits of an
            integer.'"));
        //////////

```



```

        Controller.statement.executeUpdate(getInsertQuery(table, "114", "'Palindrome
Number'", "'Question 14'",
        "'algorithms'",
        "'Determine whether an integer is a palindrome. An integer is a palindrome
when it reads the same backward as forward.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "115", "'Regular Expression
Matching'",
        "'Question 15'", "'databases'",
        "'Given an input string (s) and a pattern (p), implement regular expression
matching with support for . and *.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "116", "'Container With Most
Water'",
        "'Question 16'", "'os'",
        "'Given n non-negative integers a1, a2, ..., an , where each represents a
point at coordinate (i, ai). n vertical lines are drawn such that the two endpoints of line i is
at (i, ai) and (i, 0). Find two lines, which together with x-axis forms a container, such that
the container contains the most water.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "117", "'Integer to Roman'",
        "'Question 17'",
        "'oop'", "'Roman numerals are represented by seven different symbols: I, V,
X, L, C, D and M.'"));
        //////////
        Controller.statement.executeUpdate(getInsertQuery(table, "118", "'Count and Say'",
        "'Question 18'",
        "'data'",
        "'The count-and-say sequence is the sequence of integers with the first five
terms as following: 1 is read off as one 1 or 11. 11 is read off as two 1s or 21. 21 is read off
as one 2, then one 1 or 1211.'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "119", "'Title 19'",
        "'search-insert-position'",
        "'strings'",
        "'Given a sorted array and a target value, return the index if the target is
found. If not, return the index where it would be if it were inserted in order. You may assume no
duplicates in the array.'"));
        //////////
        Controller.statement.executeUpdate(
        getInsertQuery(table, "120", "'Title 20'", "'Question 20'", "'algorithms'",
        "'Info 20'"));
        Controller.statement.executeUpdate(
        getInsertQuery(table, "121", "'Title 21'", "'Question 21'", "'databases'",
        "'Info 21'"));
        Controller.statement
        .executeUpdate(getInsertQuery(table, "122", "'Title 22'", "'Question 22'",
        "'os'", "'Info 22'"));
        Controller.statement
        .executeUpdate(getInsertQuery(table, "123", "'Title 23'", "'Question 23'",
        "'oop'", "'Info 23'"));
        Controller.statement
        .executeUpdate(getInsertQuery(table, "124", "'Title 24'", "'Question 24'",
        "'data'", "'Info 24'"));
        Controller.statement.executeUpdate(
        getInsertQuery(table, "125", "'Title 25'", "'Question 25'", "'strings'",
        "'Info 25'"));
        Controller.statement.executeUpdate(
        getInsertQuery(table, "126", "'Title 26'", "'Question 26'", "'algorithms'",
        "'Info 26'"));
        Controller.statement.executeUpdate(

```

```

        getInsertQuery(table, "127", "'Title 27'", "'Question 27'", "'databases'",
"'Info 27'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to task table...");
        E.printStackTrace();
    }
}

public static void addPublicTask() {
    String table = "PUBLIC_TASK";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "100", "'Easy'", "10",
"85"));
        Controller.statement.executeUpdate(getInsertQuery(table, "101", "'Medium'", "20",
"68"));
        Controller.statement.executeUpdate(getInsertQuery(table, "102", "'Medium'", "20",
"77"));
        Controller.statement.executeUpdate(getInsertQuery(table, "103", "'Hard'", "30",
"50"));
        Controller.statement.executeUpdate(getInsertQuery(table, "104", "'Medium'", "20",
"30"));

        Controller.statement.executeUpdate(getInsertQuery(table, "105", "'Easy'", "10",
"85"));
        Controller.statement.executeUpdate(getInsertQuery(table, "106", "'Hard'", "30",
"68"));
        Controller.statement.executeUpdate(getInsertQuery(table, "107", "'Easy'", "10",
"77"));
        Controller.statement.executeUpdate(getInsertQuery(table, "108", "'Medium'", "20",
"50"));
        Controller.statement.executeUpdate(getInsertQuery(table, "109", "'Medium'", "20",
"30"));
    } catch (SQLException E) {
        System.out.println("Error while adding to public task table...");
        E.printStackTrace();
    }
}

public static void addPublicCodingTask() {
    String table = "PUBLIC_CODING_TASK";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "100",
"'Given nums = [2, 7, 11, 15], target = 9, Because nums[0] + nums[1] = 2 + 7
= 9, return [0, 1].'",
        "' class Solution { public String solution(int solution) { //solution }
}'", "'Hint 0'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "101",
"'Input: (2 -> 4 -> 3) + (5 -> 6 -> 4) Output: 7 -> 0 -> 8 Explanation: 342 +
465 = 807.'",
        "' class Solution { public String solution(int solution) { //solution }
}'", "'Hint 1'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "102",
"'Input: abcabcbbb Output: 3 Explanation: The answer is abc, with the length
of 3.'",

```

```

        "" class Solution { public String solution(int solution) { //solution }
}"" , ""Hint 2'");
    Controller.statement.executeUpdate(getInsertQuery(table, "103",
        ""nums1 = [1, 3] nums2 = [2] The median is 2.0'");
        "" class Solution { public String solution(int solution) { //solution }
}"" , ""Hint 3'");
    Controller.statement.executeUpdate(getInsertQuery(table, "104",
        ""Input: babad Output: bab Note: aba is also a valid answer.'",
        "" class Solution { public String solution(int solution) { //solution }
}"" , ""Hint 4'");

    } catch (SQLException E) {
        System.out.println("Error while adding to public coding task table...");
        E.printStackTrace();
    }
}

public static void addPublicNonCodingTask() {
    String table = "PUBLIC_NON_CODING_TASK";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "105"));
        Controller.statement.executeUpdate(getInsertQuery(table, "106"));
        Controller.statement.executeUpdate(getInsertQuery(table, "107"));
        Controller.statement.executeUpdate(getInsertQuery(table, "108"));
        Controller.statement.executeUpdate(getInsertQuery(table, "109"));
    } catch (SQLException E) {
        System.out.println("Error while adding to public non coding task table...");
        E.printStackTrace();
    }
}

public static void addInterviewTask() {
    String table = "INTERVIEW_TASK";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "110", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "111", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "112", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "113", "1"));
    } catch (SQLException E) {
        System.out.println("Error while adding to interview task table...");
        E.printStackTrace();
    }
}

public static void addNonCodingInterviewTask() {
    String table = "NON_CODING_INTERVIEW_TASK";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "110"));
        Controller.statement.executeUpdate(getInsertQuery(table, "111"));
    } catch (SQLException E) {
        System.out.println("Error while adding to non coding interview task table...");
        E.printStackTrace();
    }
}
}

```

```

    public static void addCodingInterviewTask() {
        String table = "CODING_INTERVIEW_TASK";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "112",
                "'Input: s = PAYPALISHIRING, numRows = 3 Output: PAHNAPLSIIGYIR'",
                "' class Solution { public String solution(int solution) { //solution }
}'", "'HINT 1'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "113", "'Input: 123 Output:
321'",
                "' class Solution { public String solution(int solution) { //solution }
}'", "'HINT 2'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to non coding interview task table...");
            E.printStackTrace();
        }
    }

    public static void addContestTask() {
        String table = "CONTEST_TASK";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "114", "2", "'Input: 121
Output: true'",
                "' class Solution { public String solution(int solution) { //solution }
}'", "'HINT 1'", "1"));
            Controller.statement.executeUpdate(getInsertQuery(table, "115", "2",
                "'Matches any single character. * Matches zero or more of the preceding
element.'",
                "' class Solution { public String solution(int solution) { //solution }
}'", "'HINT 2'", "1"));
            Controller.statement.executeUpdate(getInsertQuery(table, "116", "2",
                "'Input: [1,8,6,2,5,4,8,3,7] Output: 49'",
                "' class Solution { public String solution(int solution) { //solution }
}'", "'HINT 3'", "1"));
            Controller.statement.executeUpdate(getInsertQuery(table, "117", "2", "'Input: 3
Output: III'",
                "' class Solution { public String solution(int solution) { //solution }
}'", "'HINT 4'", "1"));
        } catch (SQLException E) {
            System.out.println("Error while adding to interview task table...");
            E.printStackTrace();
        }
    }

    public static void addCodingBetTask() {
        String table = "CODING_BET_TASK";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "118", "'Input: 4 Output:
1211'",
                "' class Solution { public String solution(int solution) { //solution }
}'", "'HINT 1'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "119", "'Input: [1,3,5,6], 5
Output: 2'",
                "' class Solution { public String solution(int solution) { //solution }
}'", "'HINT 2'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to interview task table...");
            E.printStackTrace();
        }
    }

```

```

    }
}

public static void addSubmission() {
    String table = "SUBMISSION";

    try {
        // Same user same public coding 4 submission
        Controller.statement.executeUpdate(getInsertQuery(table, "1000", "3", "100",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        Controller.statement.executeUpdate(getInsertQuery(table, "1001", "3", "100",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        Controller.statement.executeUpdate(getInsertQuery(table, "1002", "3", "100",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        Controller.statement.executeUpdate(getInsertQuery(table, "1003", "3", "100",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

        // two different users same public non coding
        Controller.statement.executeUpdate(getInsertQuery(table, "1004", "3", "105",
            "'First the Java is interpreted Language which run on JVM, but C++ will be
compiled into the machine code which run directly on cpu. So commonly, C++ native code is
faster. With JVM, java provide the memory management system which can reduce the error from
pointer..'"));

        Controller.statement.executeUpdate(getInsertQuery(table, "1005", "4", "105",
            "'First the Java is interpreted Language which run on JVM, but C++ will be
compiled into the machine code which run directly on cpu. So commonly, C++ native code is
faster. With JVM, java provide the memory management system which can reduce the error from
pointer..'"));

        // same user interview two different tasks non coding
        Controller.statement.executeUpdate(getInsertQuery(table, "1006", "3", "110",
            "'First the Java is interpreted Language which run on JVM, but C++ will be
compiled into the machine code which run directly on cpu. So commonly, C++ native code is
faster. With JVM, java provide the memory management system which can reduce the error from
pointer..'"));

        Controller.statement.executeUpdate(getInsertQuery(table, "1007", "3", "111",
            "'First the Java is interpreted Language which run on JVM, but C++ will be
compiled into the machine code which run directly on cpu. So commonly, C++ native code is
faster. With JVM, java provide the memory management system which can reduce the error from
pointer..'"));

        // same user interview two tasks coding
    }
}

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "1008", "3", "112",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "1009", "3", "113",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

```

```

        // same user contest two tasks coding
        Controller.statement.executeUpdate(getInsertQuery(table, "1010", "3", "116",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "1011", "3", "117",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

```

```

        // two users coding bet
        Controller.statement.executeUpdate(getInsertQuery(table, "1012", "3", "118",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "1013", "4", "118",
            "'class      class Solution { public String solution(int solution) {
//solution      }      }public      int      searchInsert(int[]      nums,      int      target)      {
Controller.statement.executeUpdate(getInsertQuery(table, 119, Input: [1,3,5,6], 5 Output:
2));}}'"));

```

```

    } catch (SQLException E) {
        System.out.println("Error while adding to submission table...");
        E.printStackTrace();
    }
}

public static void addNonCodingSubmission() {
    String table = "NON_CODING_SUBMISSION";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1004"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1005"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1006"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1007"));

    } catch (SQLException E) {
        System.out.println("Error while adding to non coding submission table...");
        E.printStackTrace();
    }
}

```

```

    public static void addCodingSubmission() {
        String table = "CODING_SUBMISSION";

        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "1000", "'C'", "'solved'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1001", "'C++'",
"'solved'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1002", "'JAVA'",
"'failed'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1003", "'PYTHON'",
"'solved'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1008", "'PYTHON'",
"'solved'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1009", "'JAVA'",
"'solved'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1010", "'C'", "'failed'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1011", "'C++'",
"'solved'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1012", "'PYTHON'",
"'solved'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "1013", "'C++'",
"'solved'"));

            } catch (SQLException E) {
                System.out.println("Error while adding to coding submission table...");
                E.printStackTrace();
            }
        }

        public static void addPublicNonCodingSubmission() {
            String table = "PUBLIC_NON_CODING_SUBMISSION";
            try {
                Controller.statement.executeUpdate(getInsertQuery(table, "1004", "'2019/01/01'"));
                Controller.statement.executeUpdate(getInsertQuery(table, "1005", "'2019/01/02'"));

            } catch (SQLException E) {
                System.out.println("Error while adding to public non coding submission table...");
                E.printStackTrace();
            }
        }

        public static void addInterviewNonCodingSubmission() {
            String table = "INTERVIEW_NON_CODING_SUBMISSION";
            try {
                Controller.statement.executeUpdate(getInsertQuery(table, "1006"));
                Controller.statement.executeUpdate(getInsertQuery(table, "1007"));

            } catch (SQLException E) {
                System.out.println("Error while adding to interview non coding submission table...");
                E.printStackTrace();
            }
        }

        public static void addInterviewCodingSubmission() {
            String table = "INTERVIEW_CODING_SUBMISSION";
            try {

```

```

        Controller.statement.executeUpdate(getInsertQuery(table, "1008"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1009"));

    } catch (SQLException E) {
        System.out.println("Error while adding to interview non coding submission table...");
        E.printStackTrace();
    }
}

public static void addPublicCodingSubmission() {
    String table = "PUBLIC_CODING_SUBMISSION";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1000", "'2019/02/01'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1001", "'2019/03/03'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1002", "'2018/08/11'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1003", "'2019/07/10'"));

    } catch (SQLException E) {
        System.out.println("Error while adding to public coding submission table...");
        E.printStackTrace();
    }
}

public static void addContestCodingSubmission() {
    String table = "CONTEST_CODING_SUBMISSION";

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1010"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1011"));

    } catch (SQLException E) {
        System.out.println("Error while adding to contest coding submission table...");
        E.printStackTrace();
    }
}

public static void addCodingBetSubmission() {
    String table = "CODING_BET_SUBMISSION";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1012"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1013"));

    } catch (SQLException E) {
        System.out.println("Error while adding to coding bet submission table...");
        E.printStackTrace();
    }
}

public static void addCompanyInterviewTaskPrepare() {
    String table = "COMPANY_INTERVIEW_TASK_PREPARE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "110",
"'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "111",
"'2019/12/08'"));
    }
}

```



```

        Controller.statement.executeUpdate(getInsertQuery(table, "6", "112",
"2019/12/09'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "6", "113",
"2019/12/10'"));

    } catch (SQLException E) {
        System.out.println("Error while adding to company interview task prepare table...");
        E.printStackTrace();
    }
}

public static void addEditorPublicTaskPrepare() {
    String table = "EDITOR_PUBLIC_TASK_PREPARE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "100",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "101",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "102",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "103",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "104",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "105",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "106",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "107",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "108",
"2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "109",
"2019/12/07'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to editor public task prepare table...");
        E.printStackTrace();
    }
}

public static void addEditorCodingContestPrepare() {
    String table = "EDITOR_CODING_CONTEST_PREPARE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "1", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "2", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "1", "3", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "4", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "5", "'2019/12/07'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "2", "6", "'2019/12/07'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to editor contest prepare table...");
        E.printStackTrace();
    }
}

public static void addCompanyInterviewPrepare() {
    String table = "COMPANY_INTERVIEW_PREPARE";

```

```

    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "2"));
        Controller.statement.executeUpdate(getInsertQuery(table, "5", "3"));
        Controller.statement.executeUpdate(getInsertQuery(table, "6", "4"));
        Controller.statement.executeUpdate(getInsertQuery(table, "6", "5"));
        Controller.statement.executeUpdate(getInsertQuery(table, "6", "6"));
    } catch (SQLException E) {
        System.out.println("Error while adding to company interview prepare table...");
        E.printStackTrace();
    }
}

public static void addUserContestParticipate() {
    String table = "USER_CONTEST_PARTICIPATE";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "1", "1"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "1", "2"));
    } catch (SQLException E) {
        System.out.println("Error while adding to user contest participate table...");
        E.printStackTrace();
    }
}

public static void addUserCodingBetParticipate() {
    String table = "USER_CODING_BET_PARTICIPATE";
    try {
        // userid - coding betid - place
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "1", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "2", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "3", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "3", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "2", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "4", "0"));
        Controller.statement.executeUpdate(getInsertQuery(table, "4", "5", "0"));
    } catch (SQLException E) {
        System.out.println("Error while adding to user coding bet participate table...");
        E.printStackTrace();
    }
}

public static void addUserInterviewPerform() {
    String table = "USER_INTERVIEW_PERFORM";
    try {
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "1", "'2019/12/07'",
"'waiting'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "2", "'2019/12/07'",
"'accepted'"));
        Controller.statement.executeUpdate(getInsertQuery(table, "3", "3", "'2019/12/07'",
"'rejected'"));
    } catch (SQLException E) {
        System.out.println("Error while adding to user interview perform table...");
        E.printStackTrace();
    }
}

public static void addTestCase() {

```

```

        String table = "TEST_CASE";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 1'", "'output
1'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 2'", "'output
2'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 3'", "'output
3'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 4'", "'output
4'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "100", "'input 5'", "'output
5'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to test case table...");
            E.printStackTrace();
        }
    }

    public static void addCodingBetTaskRelation() {
        String table = "CODING_BET_TASK_RELATION";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "1", "118"));
            Controller.statement.executeUpdate(getInsertQuery(table, "2", "118"));
            Controller.statement.executeUpdate(getInsertQuery(table, "3", "118"));
            Controller.statement.executeUpdate(getInsertQuery(table, "4", "119"));
            Controller.statement.executeUpdate(getInsertQuery(table, "5", "119"));
        } catch (SQLException E) {
            System.out.println("Error while adding to coding bet task relation table...");
            E.printStackTrace();
        }
    }

    public static void addRateAnswer() {
        String table = "RATE_ANSWER";
        try {
            Controller.statement.executeUpdate(getInsertQuery(table, "3", "1004", "'Upvote'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "4", "1004", "'Upvote'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "3", "1005", "'Downvote'"));
            Controller.statement.executeUpdate(getInsertQuery(table, "4", "1005", "'Downvote'"));
        } catch (SQLException E) {
            System.out.println("Error while adding to rate answer table...");
            E.printStackTrace();
        }
    }
}

public class Main {

    public static void main(String[] args)
    {
        Controller.initController();
        Controller.dropTables();
        Controller.createTables();
        Controller.initTables();
        // Controller.removeProcedures();
        // Controller.addProcedures();
    }
}

```