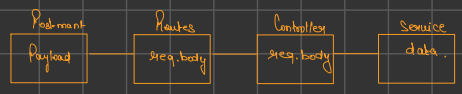
```
mysql> select * from quizzes;
+----------------------------------------------------------------------------------------------------------------+
| id | title                                                | updated_at                | created_at                |
+----------------------------------------------------------------------------------------------------------------+
| 1  | JavaScript Knowledge Quiz 2025 | 2025-10-03 19:57:54 | 2025-10-03 19:57:54 |
+----------------------------------------------------------------------------------------------------------------+
1 row in set (0.002 sec)
```

```
mysql> select * from questions;
+----------------------------------------------------------------------------------------------------------------+
| id | quiz_id | text                                                | type          | points | negative_points | created_at                | updated_at                |
+----------------------------------------------------------------------------------------------------------------+
| 1  |       1 | What is the keyword used to declare a constant in JavaScript?  | single_choice |   5.00 |            1.00 | 2025-10-03 19:58:07 | 2025-10-03 19:58:07 |
| 2  |       1 | Which symbol is used to write comments in JavaScript?          | single_choice |   5.00 |            1.00 | 2025-10-03 19:58:15 | 2025-10-03 19:58:15 |
| 3  |       1 | Which of the following are JavaScript primitive data types?    | multiple_choice |  5.00 |            2.00 | 2025-10-03 19:58:31 | 2025-10-03 19:58:31 |
| 4  |       1 | Select all array methods in JavaScript.                        | multiple_choice | 12.00 |            4.00 | 2025-10-03 19:58:43 | 2025-10-03 19:58:43 |
| 5  |       1 | Which of these are ways to declare a function in JavaScript?   | multiple_choice | 15.00 |            5.00 | 2025-10-03 19:58:53 | 2025-10-03 19:58:53 |
| 6  |       1 | Using which keyword we declare constant in javascript          | text          |   8.00 |            0.00 | 2025-10-03 19:59:02 | 2025-10-03 19:59:02 |
| 7  |       1 | Using which keyword we declare variables in javascript (answer any) | text          |   8.00 |            0.00 | 2025-10-03 19:59:11 | 2025-10-03 19:59:11 |
+----------------------------------------------------------------------------------------------------------------+
7 rows in set (0.003 sec)
```

```
mysql> select * from answers;
+----------------------------------------------------------------------------------------------------------------+
| id | question_id | text                                  | is_correct | created_at                | updated_at                |
+----------------------------------------------------------------------------------------------------------------+
|  1 |           1 | const                                 |          1 | 2025-10-03 19:58:07 | 2025-10-03 19:58:07 |
|  2 |           1 | let                                   |          0 | 2025-10-03 19:58:07 | 2025-10-03 19:58:07 |
|  3 |           1 | var                                   |          0 | 2025-10-03 19:58:07 | 2025-10-03 19:58:07 |
|  4 |           2 | //                                    |          1 | 2025-10-03 19:58:15 | 2025-10-03 19:58:15 |
|  5 |           2 | ##                                    |          0 | 2025-10-03 19:58:15 | 2025-10-03 19:58:15 |
|  6 |           2 | <!-- -->                              |          0 | 2025-10-03 19:58:15 | 2025-10-03 19:58:15 |
|  7 |           3 | String                                |          1 | 2025-10-03 19:58:31 | 2025-10-03 19:58:31 |
|  8 |           3 | Boolean                               |          1 | 2025-10-03 19:58:31 | 2025-10-03 19:58:31 |
|  9 |           3 | Object                                |          0 | 2025-10-03 19:58:31 | 2025-10-03 19:58:31 |
| 10 |           3 | Number                                |          1 | 2025-10-03 19:58:31 | 2025-10-03 19:58:31 |
| 11 |           4 | push                                  |          1 | 2025-10-03 19:58:43 | 2025-10-03 19:58:43 |
| 12 |           4 | pop                                   |          1 | 2025-10-03 19:58:43 | 2025-10-03 19:58:43 |
| 13 |           4 | length                                |          0 | 2025-10-03 19:58:43 | 2025-10-03 19:58:43 |
| 14 |           4 | indexOf                               |          1 | 2025-10-03 19:58:43 | 2025-10-03 19:58:43 |
| 15 |           5 | function Declaration                  |          1 | 2025-10-03 19:58:53 | 2025-10-03 19:58:53 |
| 16 |           5 | Arrow Function                        |          1 | 2025-10-03 19:58:53 | 2025-10-03 19:58:53 |
| 17 |           5 | Lambda Function                       |          0 | 2025-10-03 19:58:53 | 2025-10-03 19:58:53 |
| 18 |           5 | function Expression                   |          1 | 2025-10-03 19:58:53 | 2025-10-03 19:58:53 |
| 19 |           6 | const                                 |          1 | 2025-10-03 19:59:02 | 2025-10-03 19:59:02 |
| 20 |           7 | let                                   |          1 | 2025-10-03 19:59:11 | 2025-10-03 19:59:11 |
| 21 |           7 | var                                   |          1 | 2025-10-03 19:59:11 | 2025-10-03 19:59:11 |
+----------------------------------------------------------------------------------------------------------------+
21 rows in set (0.003 sec)
```




```
async createSubmission(data) {
    // Start a database transaction
    const transaction = await this.sequelize.transaction();

    try {
        // Extract question IDs from user's answers
        const questionIds = data.answers.map(a => a.questionId);

        // Fetch quiz questions with their answers filtered by quizId and questionIds
        let quizQuestions = await this.questionRepository.findQuestionsWithAnswers(data.quizId, questionIds);

        // Convert Sequelize models to plain objects
        quizQuestions = quizQuestions.map(q => q.toJSON());

        // Create a submission record with initial score and total set to 0
        const submission = await this.submissionRepository.create(
            {
                quizId: data.quizId,
                score: 0,
                total: 0,
            }, { transaction });

        let totalScore = 0;    // Sum of max possible points for questions answered
        let obtainedScore = 0;  // Accumulated score from user's answers
        const submissionAnswerRecords = []; // To hold answer records for bulk insertion

        // Process each user answer: find corresponding question and score it accordingly
        for (const userAnswer of data.answers) {
            // Find question data corresponding to the user answer
            const question = quizQuestions.find(q => q.id === userAnswer.questionId);

            // Throw error if question not found in the quiz context
            if (!question) {
                throw new AppError(`Question ${userAnswer.questionId} not found for quiz`, StatusCodes.BAD_REQUEST);
            }

            // Add question's point value to total possible score
            totalScore += parseFloat(question.points);

            let result;

            // Choose appropriate scoring function based on question type
            switch (question.type) {
                case TEXT:
                    result = scoreTextQuestion(question, userAnswer, submission.id);
                    break;
                case SINGLE_CHOICE:
                    result = scoreSingleChoiceQuestion(question, userAnswer, submission.id);
                    break;
                case MULTIPLE_CHOICE:
                    result = scoreMultipleChoiceQuestion(question, userAnswer, submission.id);
                    break;
                default:
                    throw new AppError(`Unsupported question type: ${question.type}`, StatusCodes.BAD_REQUEST);
            }

            // Accumulate obtained score from the scoring result
            obtainedScore += result.score;

            // Append submission answer records to match array
            submissionAnswerRecords.push(...result.submissionAnswers);
        }

        // Bulk insert submission answer records into DB as a single operation within transaction
        await this.submissionAnswerRepository.bulkCreate(submissionAnswerRecords, transaction);

        // Update submission record with final scores
        submission.score = obtainedScore;
        submission.total = totalScore;

        // Save updated submission record within transaction
        await submission.save({ transaction });

        // Commit transaction to make all changes permanent
        await transaction.commit();

        // Return summary of submission scoring
        return {
            quizId: submission.quizId,
            totalScore: submission.total,
            ScoreObtained: submission.score,
        };
    } catch (err) {
        // Rollback transaction if anything goes wrong
        await transaction.rollback();
        throw err;
    }
}
```

```
mysql> select * from submissions;
+----------------------------------------------------------------------------------------------------------------+
| id | quiz_id | score | total | created_at                | updated_at                |
+----------------------------------------------------------------------------------------------------------------+
|  1 |       1 |  9.00 | 37.00 | 2025-10-04 18:30:12 | 2025-10-04 18:30:12 |
+----------------------------------------------------------------------------------------------------------------+
```

obtainedScore

Total Score

```
function scoreSingleChoiceQuestion(question, userAnswer, submissionId) {
    // Find ID of the correct answer
    const correctAnswerId = question.answers.find(a => a.isCorrect).id;

    // Compare user's selectedAnswerId with correct answer ID
    const isCorrect = String(userAnswer.selectedAnswerId) === String(correctAnswerId);

    // Assign positive or negative points accordingly
    const score = isCorrect ? parseFloat(question.points) : -parseFloat(question.negativePoints);

    return {
        score,
        submissionAnswers: [{
            submissionId,
            questionId: question.id,
            selectedAnswerId: userAnswer.selectedAnswerId || null,
            textAnswer: null,
            score,
        }]
    };
}
```

userAnswer

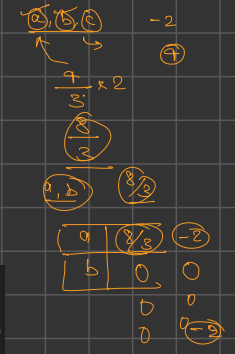
question

```
{ "questionId": 1, "selectedAnswerId": [1] }
```

```
quiz questions: [
  {
    id: 1,
    quizId: 1,
    text: 'What is the keyward used to declare a constant in JavaScript?',
    type: 'single_choice',
    points: '5.00',
    negativePoints: '1.00',
    createdAt: 2025-10-03T19:58:07.000Z,
    updatedAt: 2025-10-03T19:58:07.000Z,
    quiz_id: 1,
    answers: [ [Object], [Object], [Object] ]
  },
```

```
function scoreMultipleChoiceQuestion(question, userAnswer, submissionId) {
    // Get correct answer IDs for the question
    const correctIds = question.answers.filter(a => a.isCorrect).map(a => a.id);
    const negativePoints = parseFloat(question.negativePoints);
    const pointsPerCorrect = parseFloat(question.points) / correctIds.length;

    // Normalize selectedAnswerId to array
    const selectedIds = Array.isArray(userAnswer.selectedAnswerId)
        ? userAnswer.selectedAnswerId
        : [userAnswer.selectedAnswerId];

    // Check if any selected ID is incorrect (not in correctIds)
    const hasWrongSelected = selectedIds.some(id => !correctIds.includes(id));

    // Calculate question total score based on correctness
    let questionTotalScore = 0;
    if (hasWrongSelected) {
        questionTotalScore = -negativePoints; // Penalize for wrong selection
    } else {
        questionTotalScore = pointsPerCorrect * selectedIds.length; // Partial credit for each correct selected
    }

    // Prepare submission answer records: first selected gets full score, rest get 0
    const submissionAnswers = selectedIds.map((id, idx) => ({
        submissionId,
        questionId: question.id,
        selectedAnswerId: id,
        textAnswer: null,
        score: idx === 0 ? questionTotalScore : 0,
    }));

    return {
        score: questionTotalScore,
        submissionAnswers,
    };
}
```

```
{ "questionId": 3, "selectedAnswerId": [7, 8, 10] },
```


a b c
-2
+
a
b
c
a
b
c
a
b
c