**CS6.302 - Software System Development**

**International Institute of Information Technology, Hyderabad**

**QUIZ – 1 Duration: 45 Minutes Total Marks: 20**

***Please use the empty space to answer your questions***

**ROLL NUMBER:** _____

---

**Q1: Considering GIT as your underlying tool, Provide your responses in a single line.**

**(a)** Considering the underlying tool as GIT, let us assume you modify 15 files in a repository with 100 files and run the following commands:

```
git commit -m "I made some changes"
git push
```

However, when you open GitHub, you don't see the changes. What do you think could have gone wrong? **[1 Marks]**

**(b)** Considering GIT as your tool, write the git command to switch from the `master` branch to the `testing` branch. **[1 Marks]**

**(c)** Considering GIT as your tool, what is the command to undo the last commit while keeping your changes? **[1 Marks]**

**(d)** Your teammate updates the `codebase()` function and pushes the changes; you later modify the same function locally without pulling their changes. What happens when you push, and how can you resolve it? **[1 Marks]**

**(e)** You are tasked to correct an HTML file in your new company and commit your fixed code to an incorrect repo. How do you undo the commit? **[1 Mark]**

**1a.** Your changes didn't show up on GitHub because either you didn't stage them before committing, or you pushed to a different branch/remote than the one you're viewing.

**1b.** You can switch from the master branch to the testing branch using '*git checkout testing*'

**1c:** *'git reset --soft HEAD~1'* - This undoes the last commit but keeps all your changes staged. (OR) If you want the changes unstaged but still in your working directory, use: *'git reset --mixed HEAD~1'*

**1d:** Your push will be rejected as a non-fast-forward because your branch is behind; fix it by integrating the remote changes first, resolving the conflict in `codebase()` locally, and then pushing

**1e:** If you haven't pushed yet, run `git reset --soft HEAD~1` in the wrong repo to undo the commit but keep your changes, then apply them in the correct repo (OR)
if history rewrite is allowed, `git reset --hard HEAD~1 && git push --force-with-lease`), then move the fix to the right repo (e.g., `git cherry-pick <bad-commit-sha>` or apply a patch).

**Q2: A company maintains an students table with the details mentioned in the table below.**

| Column Name | Data Type | Description |
|---|---|---|
| id | INT | Unique identifier for each student |
| name | VARCHAR | Student's full name |
| branch | VARCHAR | Engineering branch that a student enrolled into |
| subject | VARCHAR | Engineering Subject associate with branch |
| score | FLOAT | Score against respective subject with scale of 100 |
| grade | INT | Student's letter grade (A, A-, B, B-, C, C-, D, E) |

**(a)** Write an SQL query to retrieve the **names** and **grade** of students in the 'EEE' Engineering branch whose grade is below than grade 'B' **[1 Mark]**

**(b)** Write an SQL query to find the average score of students for each branch, sorted in descending order of average score. **[1 Mark]**

**(c)** Write an SQL query to find a student with `name "Sai Anirudh"` and cut down the grade to 'D' wherever the score is between 55 and 60 . **[1 Mark]**

**(d)** Write an SQL query to list the top 2 branches that contain the highest number of students with 'E' grade. **[1 Marks]**

**(e)** Write an SQL query to display top scorer details in each subject **[1 Marks]**

**2a:**

```
SELECT name, grade FROM students WHERE branch = 'EEE'   AND grade IN ('B-','C','C-','D','E');
```

**2b:**

```
SELECT branch, AVG(score) AS avg_score FROM students GROUP BY branch
ORDER BY avg_score DESC;
```

**2c:**

```
UPDATE students SET grade = 'D' WHERE name = 'Sai Anirudh'  AND score BETWEEN 55 AND 60;
```

**2d:**

```
SELECT branch, COUNT(DISTINCT id) AS e_students FROM students WHERE grade = 'E'
GROUP BY branch ORDER BY e_students DESC LIMIT 2;
```

**2e:**

```
SELECT s.subject, s.id, s.name, s.branch, s.score, s.grade FROM students s
JOIN (  SELECT subject, MAX(score) AS max_score   FROM students   GROUP BY subject ) m
  ON m.subject = s.subject AND s.score = m.max_score ORDER BY s.subject;
```

**Q3: Consider the student table definition from previous question as a student Collection. Answer the following questions using NoSQL.**

**(a)** Write a NoSQL query to insert 2 student documents using a single NoSQL query **[1 Mark]**

**(b)** Write a NoSQL query to update letter grade of student `'Sai Anirudh'` to `'A'` despite the score is less than '50' **[1 Mark]**

**(c)** Write a NoSQL query to display top scorer details in each subject **[1 Mark]**

**(d)** Write a NoSQL query to find the average score of students for each branch **[1 Mark]**

**(e)** Write a NoSQL query to retrieve the **names** and **grade** of students in the 'EEE' Engineering branch whose grade is below grade 'B' **[1 Mark]**

**3a:**

```
db.students.insertMany([
{ id: 101, name: "Sai Anirudh", branch: "EEE", subject: "Circuits",  score: 88.5, grade: "A" },
{ id: 102, name: "Anita Sharma", branch: "CSE", subject: "Data Structures", score: 79.0, grade: "B" }
]);
```

**3b:**

```
db.students.updateMany(
 { name: "Sai Anirudh", score: { $lt: 50 } },
 { $set: { grade: "A" } }
);
```

**3c:**

```
 db.students.aggregate([
{ $sort: { subject: 1, score: -1 } },
{ $group:
{ _id: "$subject",
id: { $first: "$id" },
name: { $first: "$name" },
branch: { $first: "$branch" },
score: { $first: "$score" },
grade: { $first: "$grade" } }},
{ $project: { _id: 0, subject: "$_id", id: 1, name: 1, branch: 1, score: 1, grade: 1 } } ]);
```

**3d:**

```
db.students.aggregate([
{ $group: { _id: "$branch", avgScore: { $avg: "$score" } } },
{ $project: { _id: 0, branch: "$_id", avgScore: { $round: ["$avgScore", 2] } } } ]);
```

**3e:**

```
db.students_scores.find(
 { branch: "EEE", grade: { $in: ["B-", "C", "C-", "D", "E"] }},
 { _id: 0, name: 1, grade: 1 }
);
```

**Q4: Using HTML and CSS, answer the following in single line**

**(a)** Provide a simple example to illustrate inline CSS **[1 Mark]**

**(b)** Creating a Hyperlink: Write an HTML snippet to create a hyperlink that opens **`https://sports.iiit.ac.in`** in a new tab when clicked. **[1 Mark]**

**(c)** Displaying an Audio file: Write an HTML snippet to display audio tag and pass required arguments to load the source audio file. **[1 Mark]**

**(d)** What attributes can a **`<img>`** tag have? **[1 Mark]**

**(e)** What is difference between **`<span>`** and **`<div>`**. **[1 Mark]**

**4a:**

```
<p style="color: #334155; line-height: 1.6;">
    This paragraph is styled directly on the element using the <code>style</code> attribute.
 </p>
```

**4b:**

```
<a href="https://sports.iiit.ac.in" target="_blank" rel="noopener noreferrer">Open IIIT Sports</a>
```

**4c:** `<audio controls src="media/song.mp3"></audio>`

**4d:**

- **src** – image URL
- **alt** – accessible text (required for good HTML)
- **width**, **height** – display dimensions (ideally match intrinsic size)

**4e**:

 is a block-level container;  is an inline container.  creates a new line and can use width/height;  stays inline and ignores width/height by default.