Name : Vaibhav kumar gupta

Date : 13-02-2025

SQL:-

SQL Concepts

Creating and Populating Tables

```
INSERT INTO target_table (col1, col2)
SELECT col1, col2 FROM source_table;
```

Cascade Operations in SQL

These constraints help maintain referential integrity when performing `DELETE` or `UPDATE` operations on parent-child tables.

- **ON DELETE CASCADE**: Automatically deletes related child records when the parent is deleted.
- **ON UPDATE CASCADE**: Updates child records when the parent key changes.
- **ON DELETE SET NULL**: Sets child records to `NULL` when the parent is deleted.

```
CREATE TABLE parent (
    id INT PRIMARY KEY
);

CREATE TABLE child (
    id INT PRIMARY KEY,
    parent_id INT,
    FOREIGN KEY (parent_id) REFERENCES parent(id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Using Triggers for Logging

Triggers automatically execute actions when certain database events occur.

```sql
CREATE TRIGGER log_changes
AFTER INSERT ON employees
FOR EACH ROW
INSERT INTO logs (action, timestamp)
VALUES ('New Employee Added', NOW());
```

Terminal and Command Line Basics

- The terminal is a text-based interface for executing commands in various systems (Node.js, Git, Linux, etc.).
- Git Bash (for Windows) allows running Unix-based Git and terminal commands.

Command Shortcuts

- Autocomplete commands → Press Tab
- Repeat recent commands → Use ↑ / ↓ arrow keys
- Change directory → cd folder_name
- Move back a directory → cd ..
- Move multiple levels → cd ../../target_directory
- Absolute path navigation → cd /home/user/projects

- # File and Directory Management

Creating a Folder

```
mkdir my_project
```

Creating a File

```
touch index.html
```

Removing Files and Folders

```
rm file.txt        # Deletes a file
rmdir empty_dir    # Removes an empty folder
rm -rf my_folder   # Force deletes a folder and its contents
```

Git Basics

Setting Up Git

```
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
```

Initialize a Repository

```
git init
```

Clone an Existing Repository

```
git clone https://github.com/user/repository.git
```

Checking Git Status

```
git status
```

Staging and Committing Changes

```
git add filename     # Stage a single file
git add .            # Stage all changes
git commit -m "Added new features"
```

Pushing Changes to GitHub

```
git push origin main
```

Branching and Merging in Git

Creating a New Branch

```
git branch feature-branch
```

Switching to a Branch

```
git checkout feature-branch
```

Merging a Branch

```
git checkout main
git merge feature-branch
```

Checking Differences Between Branches

```
git diff main feature-branch
```

Pulling Changes from Remote Repository

```
git pull origin main
```

Undoing Changes in Git

Reset Staged Changes

```
sh
CopyEdit
git reset filename
git reset
```

Undo Last Commit

```
sh
CopyEdit
git reset --soft HEAD~1   # Undo last commit, keep changes
git reset --hard HEAD~1   # Undo last commit, discard changes
```

Reverting to a Specific Commit

```
sh
CopyEdit
git log                 # Find the commit hash
git reset --hard <commit_hash>  # Revert to a previous commit
```

Working with GitHub

Creating a New GitHub Repository

1. Go to GitHub and create a new repository.
2. Copy the repository link.
3. Set the remote repository in Git:

```
sh
CopyEdit
git remote add origin https://github.com/user/repository.git
```

4. Push your project to GitHub:

```
sh
CopyEdit
git push -u origin main
```

Git Branching and Pull Requests (PRs)

Forking and Contributing to Other Repositories

- **Fork a repository** → Click "Fork" on GitHub.
- **Make changes** → Modify files in your local fork.
- **Create a Pull Request (PR)** → Request merging changes into the original repository.

Pushing Changes from a Feature Branch

```
CopyEdit
git push origin feature-branch
```

Merging a PR on GitHub

1. Open the PR on GitHub.
2. Click **Compare & pull request**.
3. Merge the PR if no conflicts exist.
4. Delete the feature branch if needed.

Handling Merge Conflicts in VS Code

If conflicts arise when merging, Git allows you to resolve them manually:

```
CopyEdit
git merge feature-branch
```

- Open conflicting files in **VS Code**.
- Choose which changes to keep.
- Stage and commit the resolved file.

Git Reset and Undoing Commits

Reset Staged Changes

```
sh
CopyEdit
git reset HEAD filename
```

Undo a Committed Change

```
sh
CopyEdit
git revert HEAD   # Reverts the last commit
```

Undo Multiple Commits

```
sh
CopyEdit
git reset --hard HEAD~2   # Undo last 2 commits
```