Data Cleaning of Layoff Records

1. **Project Overview**

- **Title**: Data Cleaning of Layoff Records
- **Objective**: To clean and standardize a dataset containing records of layoffs, ensuring that the data is accurate, consistent, and ready for analysis.

2. **Data Source**

- **Description**:The dataset contains records of layoffs across various companies and industries. Each record provides essential information about the layoffs

Dataset Schema

Column Name	Data Type	Description
company	Text	The name of the company that conducted the layoffs.
location	Text	The geographic location of the company (city/state/country).
industry	Text	The industry sector to which the company belongs.
total_laid_off	Integer	The total number of employees laid off.
percentage_laid_off	Text	The percentage of the workforce that was laid off.
date	Text	The date when the layoffs occurred (formatted as MM/DD/YYYY).
stage	Text	The stage of the company (e.g., restructuring, downsizing).
country	Text	The country where the company is headquartered.
funds_raised_millions	Integer	The amount of funds raised by the company in millions.

3. **Data Cleaning Steps**

- **Step 1: Remove Duplicates**
  - **Description**: Identify and remove duplicate records based on key columns.
  - **SQL Code**:

```sql
WITH duplicate_cte AS (
    SELECT *,
        ROW_NUMBER() OVER (PARTITION BY company, location, industry, total_laid_off, percentage_laid_off, `date`,
stage, country, funds_raised_millions ORDER BY company) AS row_num
    FROM layoffs_staging
)
DELETE FROM duplicate_cte WHERE row_num > 1;
```

- Step 2: Standardize Data
  - **Description**: Clean up whitespace, unify naming conventions, and format dates.
  - SQL Code:

```sql
UPDATE layoffs_staging2
SET company = TRIM(company),
    industry = CASE WHEN industry LIKE 'Crypto%' THEN 'Crypto' ELSE TRIM(industry) END,
    country = TRIM(TRAILING '.' FROM country),
    `date` = STR_TO_DATE(`date`, '%m/%d/%Y');
```

- Step 3: Handle Null Values
  - **Description**: Identify and manage records with null or blank values.
  - SQL Code:

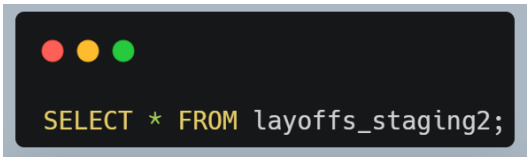
```sql
DELETE FROM layoffs_staging2 WHERE total_laid_off IS NULL OR percentage_laid_off IS NULL;
```

- Step 4: Remove Unwanted Columns
  - **Description**: Drop any columns that are not necessary for analysis.
  - SQL Code:

```sql
ALTER TABLE layoffs_staging2 DROP COLUMN row_num;  -- If you had a row_num column
```

4. Final Data Structure

- **Description**: Provide the schema of the cleaned dataset.
- SQL Code:



5. **Results and Insights**

- **Description**: The cleaned data reveals significant layoffs, particularly in technology and finance, with peaks in early 2023, concentrated in the U.S., highlighting economic challenges affecting various industries and companies.

6. **Documentation**

- **Comments**: Ensure your SQL code is well-commented to explain what each part does.
- **Readme File**: Create a README file that explains the project, how to run the SQL scripts, and any dependencies required.

7. **Presentation**

- **Visualizations**: Consider using a tool like Tableau or Power BI to create visualizations from the cleaned data.
- **Portfolio Website**: If you have a personal website, include a section for this project with an overview, code snippets, and visualizations.

8. **Future Work**

- **Improvements**: Discuss any potential improvements or additional analyses you could perform with the cleaned data.
- **Automation**: Consider how you could automate this cleaning process for future datasets.

Best Practices

- **Version Control**: Use Git to track changes in your SQL scripts and document your progress.
- **Testing**: Test your SQL queries on a small subset of data before running them on the entire dataset.
- **Backup Data**: Always keep a backup of the original raw data before performing any cleaning operations.