SQL Data Cleaning & Exploratory Data Analysis (EDA)

1. Introduction

The objective of this project was to clean and analyze a dataset related to company layoffs. The dataset contained information about companies, industries, countries, and the number of employees laid off.

2. Data Cleaning Steps

The following steps were performed during data cleaning:

- - Removed duplicate rows
- - Standardized company names, industries, and country names
- - Handled missing/null values appropriately
- - Corrected date formats for consistency

3. Exploratory Data Analysis (EDA)

The following analyses were conducted:

- - Maximum layoffs observed in dataset
- - Companies with 100% layoffs
- - Top companies with most layoffs
- - Yearly layoff trends
- - Monthly layoff trends
- - Rolling total layoffs
- - Top companies by year

4. Key Insights & Findings

- Certain industries experienced significantly higher layoffs compared to others.
- Some companies had 100% of their workforce laid off.
- Layoffs showed clear yearly and monthly patterns, with spikes in specific periods.
- Rolling totals highlighted long-term layoff trends.

5. Conclusion

This project highlighted the importance of proper data cleaning before conducting analysis. The exploratory data analysis provided insights into how layoffs have impacted companies over time, revealing key trends and industry-specific effects.

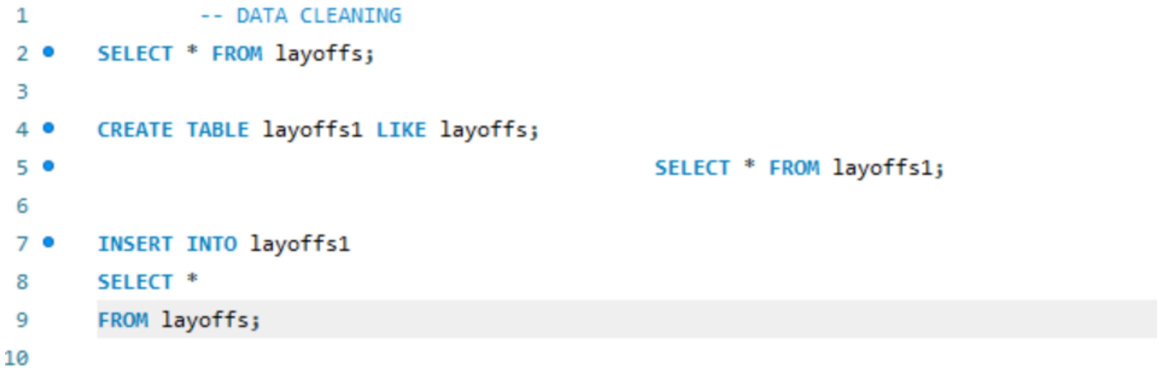
6. Appendix - SQL Queries

The project included SQL queries for data cleaning and EDA. Example queries:

> *SELECT company, SUM(total_laid_off) FROM layoffs GROUP BY company;*
>
> *SELECT YEAR(date), SUM(total_laid_off) FROM layoffs GROUP BY YEAR(date);*

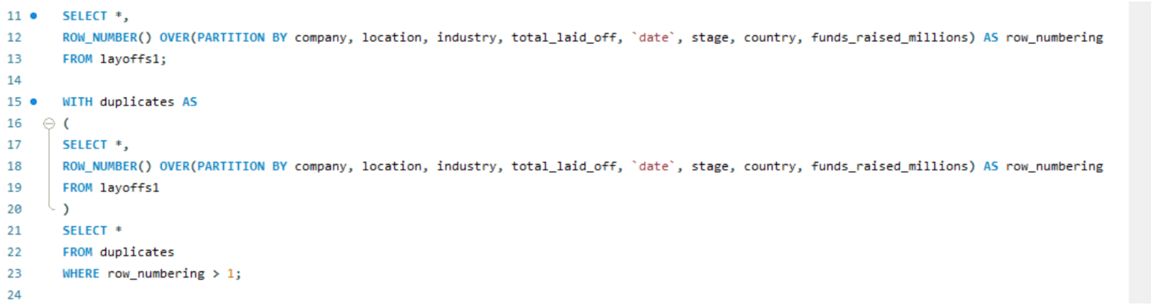
SQL Data Cleaning & Exploratory Data Analysis (EDA) – Layoffs Dataset:

1. **Creating a Backup Table :**

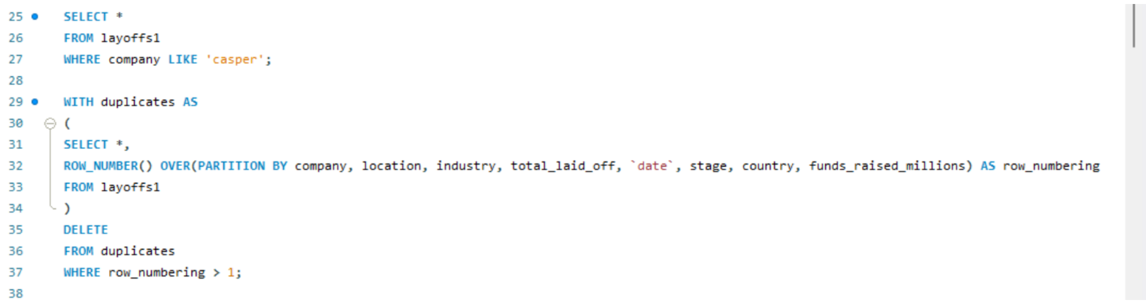


```
        -- DATA CLEANING
SELECT * FROM layoffs;

CREATE TABLE layoffs1 LIKE layoffs;
                                        SELECT * FROM layoffs1;

INSERT INTO layoffs1
SELECT *
FROM layoffs;
```

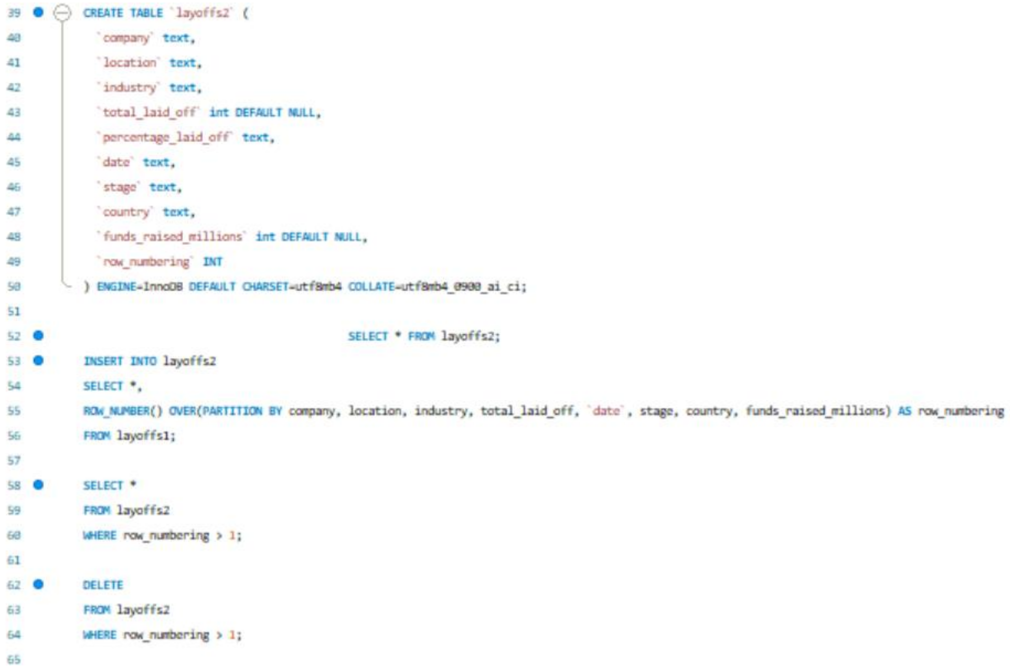
2. **Identifying and Removing Duplicates :**



```
SELECT *,
ROW_NUMBER() OVER(PARTITION BY company, location, industry, total_laid_off, `date`, stage, country, funds_raised_millions) AS row_numbering
FROM layoffs1;

WITH duplicates AS
(
SELECT *,
ROW_NUMBER() OVER(PARTITION BY company, location, industry, total_laid_off, `date`, stage, country, funds_raised_millions) AS row_numbering
FROM layoffs1
)
SELECT *
FROM duplicates
WHERE row_numbering > 1;
```

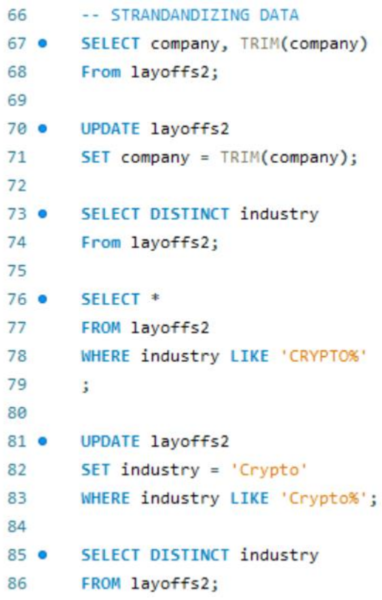


```sql
SELECT *
FROM layoffs1
WHERE company LIKE 'casper';

WITH duplicates AS
(
SELECT *,
ROW_NUMBER() OVER(PARTITION BY company, location, industry, total_laid_off, `date`, stage, country, funds_raised_millions) AS row_numbering
FROM layoffs1
)
DELETE
FROM duplicates
WHERE row_numbering > 1;
```

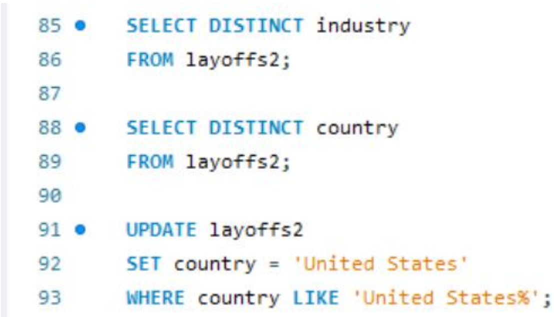


```sql
CREATE TABLE `layoffs2` (
  `company` text,
  `location` text,
  `industry` text,
  `total_laid_off` int DEFAULT NULL,
  `percentage_laid_off` text,
  `date` text,
  `stage` text,
  `country` text,
  `funds_raised_millions` int DEFAULT NULL,
  `row_numbering` INT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

                                        SELECT * FROM layoffs2;
INSERT INTO layoffs2
SELECT *,
ROW_NUMBER() OVER(PARTITION BY company, location, industry, total_laid_off, `date`, stage, country, funds_raised_millions) AS row_numbering
FROM layoffs1;

SELECT *
FROM layoffs2
WHERE row_numbering > 1;

DELETE
FROM layoffs2
WHERE row_numbering > 1;
```

3. Standardizing Text Data :

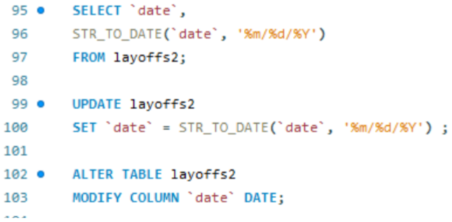


```sql
-- STRANDANDIZING DATA
SELECT company, TRIM(company)
From layoffs2;

UPDATE layoffs2
SET company = TRIM(company);

SELECT DISTINCT industry
From layoffs2;

SELECT *
FROM layoffs2
WHERE industry LIKE 'CRYPTO%'
;

UPDATE layoffs2
SET industry = 'Crypto'
WHERE industry LIKE 'Crypto%';

SELECT DISTINCT industry
FROM layoffs2;
```



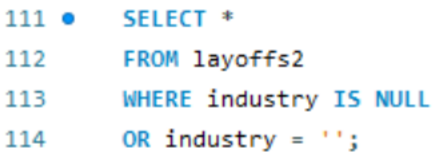
```sql
SELECT DISTINCT industry
FROM layoffs2;

SELECT DISTINCT country
FROM layoffs2;

UPDATE layoffs2
SET country = 'United States'
WHERE country LIKE 'United States%';
```

4. Fixing Dates :

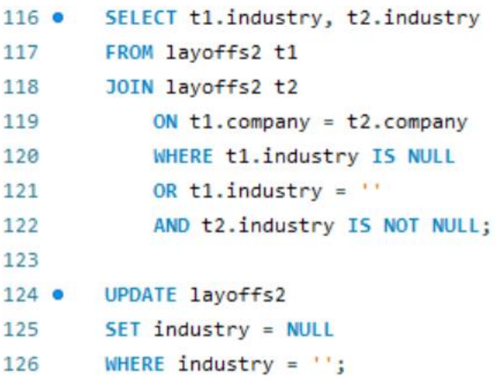


```sql
SELECT `date`,
STR_TO_DATE(`date`, '%m/%d/%Y')
FROM layoffs2;

UPDATE layoffs2
SET `date` = STR_TO_DATE(`date`, '%m/%d/%Y') ;

ALTER TABLE layoffs2
MODIFY COLUMN `date` DATE;
```

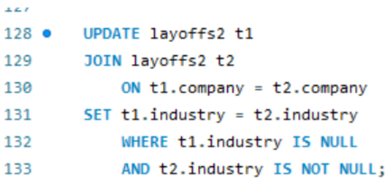
5. Handling Null & Missing Values :



```sql
SELECT *
FROM layoffs2
WHERE industry IS NULL
OR industry = '';
```



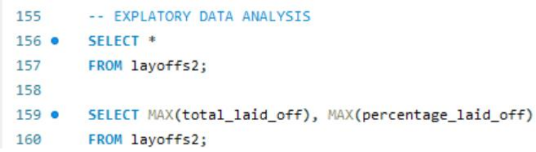
```sql
SELECT t1.industry, t2.industry
FROM layoffs2 t1
JOIN layoffs2 t2
    ON t1.company = t2.company
    WHERE t1.industry IS NULL
    OR t1.industry = ''
    AND t2.industry IS NOT NULL;

UPDATE layoffs2
SET industry = NULL
WHERE industry = '';
```



```sql
UPDATE layoffs2 t1
JOIN layoffs2 t2
    ON t1.company = t2.company
SET t1.industry = t2.industry
    WHERE t1.industry IS NULL
    AND t2.industry IS NOT NULL;
```

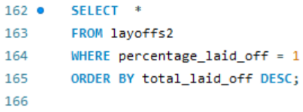
Exploratory Data Analysis (EDA) :

5 Maximum layoffs :



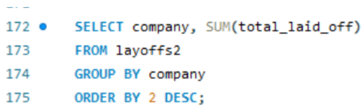
```
-- EXPLATORY DATA ANALYSIS
SELECT *
FROM layoffs2;

SELECT MAX(total_laid_off), MAX(percentage_laid_off)
FROM layoffs2;
```

6 Companies with 100% layoffs :



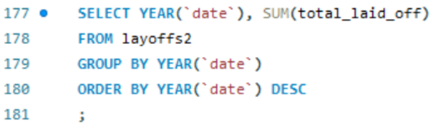
```
SELECT  *
FROM layoffs2
WHERE percentage_laid_off = 1
ORDER BY total_laid_off DESC;

```

7 Companies with most layoffs :



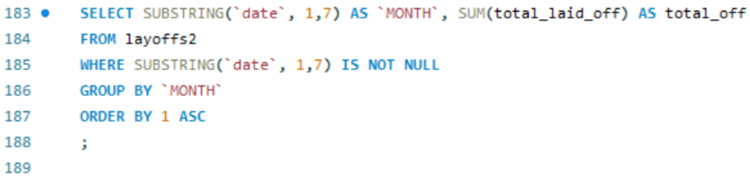
```
SELECT company, SUM(total_laid_off)
FROM layoffs2
GROUP BY company
ORDER BY 2 DESC;
```

8 Layoffs by Year :



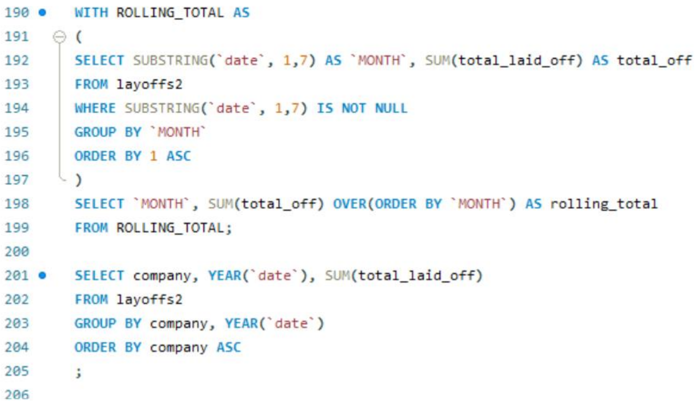
```
SELECT YEAR(`date`), SUM(total_laid_off)
FROM layoffs2
GROUP BY YEAR(`date`)
ORDER BY YEAR(`date`) DESC
;
```

9 Monthly Trends :



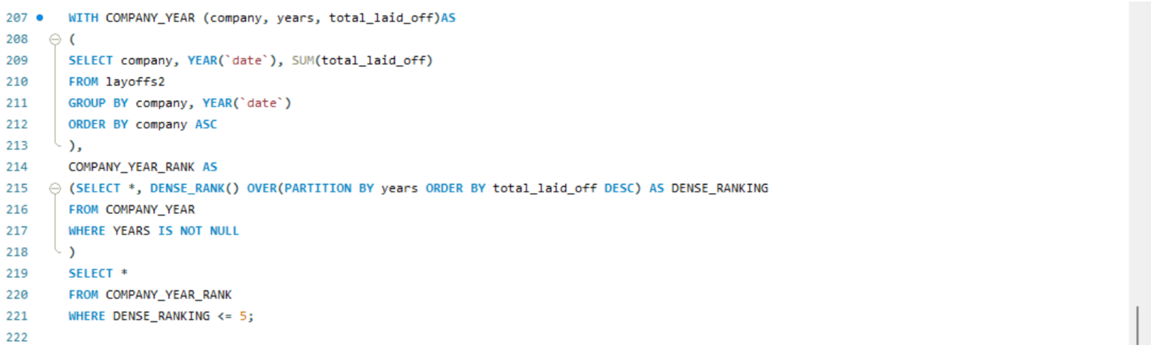
```
SELECT SUBSTRING(`date`, 1,7) AS `MONTH`, SUM(total_laid_off) AS total_off
FROM layoffs2
WHERE SUBSTRING(`date`, 1,7) IS NOT NULL
GROUP BY `MONTH`
ORDER BY 1 ASC
;

```

10 Rolling Total of Layoffs :



```
WITH ROLLING_TOTAL AS
(
SELECT SUBSTRING(`date`, 1,7) AS `MONTH`, SUM(total_laid_off) AS total_off
FROM layoffs2
WHERE SUBSTRING(`date`, 1,7) IS NOT NULL
GROUP BY `MONTH`
ORDER BY 1 ASC
)
SELECT `MONTH`, SUM(total_off) OVER(ORDER BY `MONTH`) AS rolling_total
FROM ROLLING_TOTAL;

SELECT company, YEAR(`date`), SUM(total_laid_off)
FROM layoffs2
GROUP BY company, YEAR(`date`)
ORDER BY company ASC
;

```

11 **Top Companies by Year :**



```sql
WITH COMPANY_YEAR (company, years, total_laid_off)AS
(
SELECT company, YEAR(`date`), SUM(total_laid_off)
FROM layoffs2
GROUP BY company, YEAR(`date`)
ORDER BY company ASC
),
COMPANY_YEAR_RANK AS
(SELECT *, DENSE_RANK() OVER(PARTITION BY years ORDER BY total_laid_off DESC) AS DENSE_RANKING
FROM COMPANY_YEAR
WHERE YEARS IS NOT NULL
)
SELECT *
FROM COMPANY_YEAR_RANK
WHERE DENSE_RANKING <= 5;
```

Conclusion:

This project demonstrated the importance of structured data cleaning and exploratory analysis in deriving meaningful insights from raw datasets. By addressing duplicates, inconsistencies, and missing values, the data was transformed into a reliable foundation for analysis. The EDA highlighted key layoff patterns, industry impacts, and temporal trends, offering valuable perspectives for strategic decision-making. Overall, the study reinforces that clean, well-analyzed data is critical to driving informed business outcomes.
