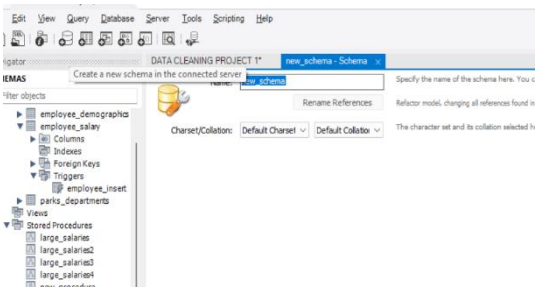
DATA CLEANING PROJECT

WORLD LAYOFFS 2021-2023

1. Creating new database (new schema) with existing excel (csv) file.



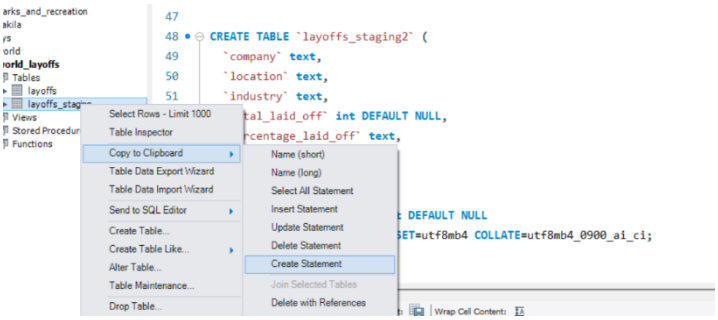
2. Creating new table to be able to clean the data for the future project (Project 2).
3. Check and remove duplicates.
4. Standardize the Data (if there are any issues).
5. Check for NULL or Blank values.
6. Remove Any columns (unnecessary columns).

```
WITH duplicate_cte AS
(
SELECT *,
ROW_NUMBER() OVER(
PARTITION BY company, industry, total_laid_off, percentage_laid_off, `date`) AS row_num
FROM layoffs_staging
)
SELECT *
FROM duplicate_cte
WHERE row_num >1;
```

6.1. We create another table (layoffs_staging) as we'll be changing our data a lot and we don't want to lose it.

6.2. It is really important (when there are more columns) to check some of data if that is really a duplicate. Example here – "**not including all the columns showed that some rows where taken as duplicates by mistake** (some columns had different data for those rows)". So it's important to include **all the columns** when checking table for duplicates.

7. Creating additional table to be able to delete duplicates and not lost data (MySQL*).



7.1. Then we use **DELETE** statement to delete duplicates from layoffs_staging2 table. We'll delete **row_num** column later. (*SQL Editor settings to turn off Safe mode)

8. Using **TRIM** to delete spaces in **company** column.

```
SELECT company, TRIM(company)
FROM layoffs_staging2;

UPDATE layoffs_staging2
SET company = TRIM(company);
```

9. Used **UPDATE** to clean data in **Industry** column. Proceeding the checking rest of the columns.

```sql
SELECT *
FROM layoffs_staging2
WHERE industry LIKE 'Crypto%'
ORDER BY industry;

UPDATE layoffs_staging2
SET industry = 'Crypto'
WHERE industry LIKE 'Crypto%';
```

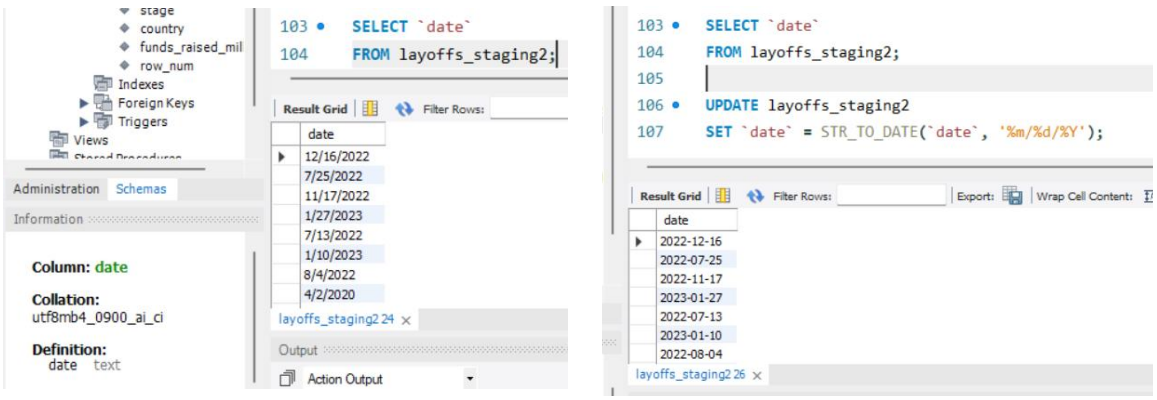
10. Cleaning **countries** in two ways (we had “United States” and “United States**.**” (with a dot))**.**

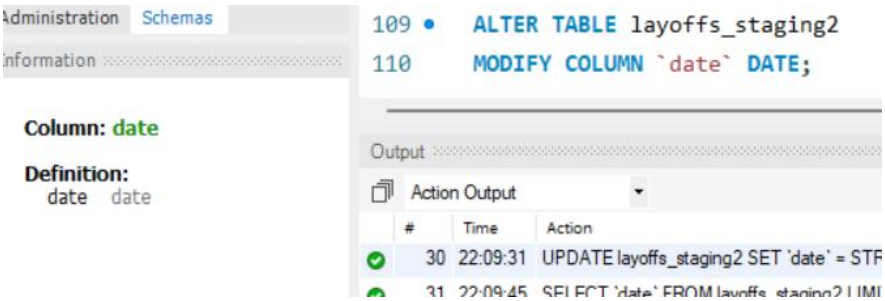
```sql
SELECT DISTINCT country
FROM layoffs_stagingsg2
WHERE country LIKE 'United St%';

UPDATE layoffs_staging2
SET country = 'United States'
WHERE country = 'United States.';
```

```sql
SELECT DISTINCT country, TRIM(TRAILING '.' FROM country)
FROM layoffs_staging2
ORDER BY 1;
```

11. Now we change data type of column **date** from Text to String, but it’ ll still keep the data type as Text.

11.1. To change it we use **ALTER TABLE.**





12. Some of **Industries** are Blank or NULL. If there are other data for those, we might need to populate it (check if it’s useless for us or no beforehand).

```sql
123
124 • SELECT *
125 FROM layoffs_staging2
126 WHERE industry IS NULL
127 OR industry = '';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company	location	industry	total_laid_off	percentage_laid_off	date
Airbnb	SF Bay Area		30	NULL	2023-03-03
Bally's Interactive	Providence	NULL	NULL	0.15	2023-01-18
Carvana	Phoenix		2500	0.12	2022-05-10
Juul	SF Bay Area		400	0.3	2022-11-10

12.1. Before we populate (change) anything that has NULLs or BLANKs we need to check (better on JOINS, or SELF JOINS in this case) if it’s true that there are NULLs and BLANKs for everything or we have information perhaps in other rows. Then we make an **UPDATE.**

```sql
SELECT t1.company, t1.location, t1.industry, t2.industry
FROM layoffs_staging2 t1
JOIN layoffs_staging2 t2
    ON t1.company = t2.company
    AND t1.location = t2.location
WHERE (t1.industry IS NULL OR t1.industry = '')
AND t2.industry IS NOT NULL;
```

```sql
UPDATE layoffs_staging2
SET industry = NULL
WHERE industry = '';

SELECT t1.company, t1.location, t1.industry, t2.industry
FROM layoffs_staging2 t1
JOIN layoffs_staging2 t2
    ON t1.company = t2.company
    AND t1.location = t2.location
WHERE (t1.industry IS NULL OR t1.industry = '')
AND t2.industry IS NOT NULL;

UPDATE layoffs_staging2 t1
JOIN layoffs_staging2 t2
    ON t1.company = t2.company
SET t1.industry = t2.industry
WHERE t1.industry IS NULL
AND t2.industry IS NOT NULL;
```

12.2. Here we also used JOIN to find what data was empty (blank) and what was having NULL so to be able to populate **the blanks with nulls at first**, and then **make an update with the existing** data.

13. Now we found that there are a lot of rows that **have no lay offs and no percentage of lay offs**, which means such rows are not really useful for our project as we can't populate, find this information, it was just not added there initially. So we can **DELET**E it.

```sql
SELECT *
FROM layoffs_staging2
WHERE total_laid_off IS NULL
AND percentage_laid_off IS NULL;

DELETE
FROM layoffs_staging2
WHERE total_laid_off IS NULL
AND percentage_laid_off IS NULL;
```

Grid | Filter Rows: | Export: | Wrap

mpany	location	industry	total_laid_off	percenta
nc.	Toronto	Transportation	NULL	NULL
) Thieves	Los Angeles	Retail	NULL	NULL
colade	Seattle	Healthcare	NULL	NULL
a	Toronto	Support	NULL	NULL
ara	SF Bay Area	Travel	NULL	NULL

13.1. We also have column we added earlier to look for duplicates that we don't need anymore and we can **DROP** it.

```sql
ALTER TABLE layoffs_staging2
DROP COLUMN row_num;
```

14. Additionally, I wanted to clean the **NULLs** data rows from our database (in columns **total, percentage, stage** and **funds)**, however without losing all possible aggregation functions we can use in the future and without changing column's data types.

```sql
SELECT
    company,location,industry,
    CONCAT(IFNULL(total_laid_off, 'No data')) AS total_laid_off_display,
    CONCAT(IFNULL(percentage_laid_off, 'No data')) AS percentage_laid_off_display, -- as we don't want to change the type of column (from INT to STR)
    `date`,stage,country,funds_raised_millions
FROM layoffs_staging2;
```

15. For the future, just wanted to learn for myself, how can I quickly get all the columns I have in the table (as there can be way more that 3 for instance) and how to create a default view.

```sql
SELECT GROUP_CONCAT(column_name ORDER BY ordinal_position)  -- how to choose all your column names and create a new view if needed
FROM information_schema.columns
WHERE table_schema = 'world_layoffs'
AND table_name = 'layoffs_staging2';

CREATE VIEW layoffs_staging2_revised AS
SELECT company,location,industry,percentage_laid_off,`date`,stage,country,funds_raised_millions
FROM layoffs_staging2;

SELECT *
FROM layoffs_staging2_revised;
```

16. It, however, makes no harm to change from whatever type was in column **percentage** to **FLOAT.**

```sql
ALTER TABLE layoffs_staging2
MODIFY COLUMN percentage_laid_off FLOAT;
```

17. Next, I wanted to display how the **NULLs** disappeared from the report (replaced by **No data**) and kept all rows in table…
    17.1. …using previous **display** functionality with **CONCAT** & **IFNULL** and now **UNION ALL** to show the short report.

```sql
-- 1. Summary:
SELECT
    '= DATA CLEANING SUMMARY =' AS company,
    'NULLs replaced with "No data"' AS location,
    '' AS industry,
    CONCAT(
        (SELECT COUNT(*) FROM layoffs WHERE total_laid_off IS NULL),
        ' → 0'
    ) AS total_laid_off_display,
    CONCAT(
        (SELECT COUNT(*) FROM layoffs WHERE percentage_laid_off IS NULL),
        ' → 0'
    ) AS percentage_laid_off_display,
    '' AS `date`,
    CONCAT(
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company	location	industry	total_laid_off_display	percentage_laid_off_display	date	stage	country	funds_raised_millions
= DATA CLEANING SUMMARY =	NULLs replaced with "No data"		740 → 0	785 → 0		6 → 0		209 → 0
Company	Location	Industry	Total Laid Off	Percentage Laid Off	Date	Stage	Country	Funds Raised
Included Health	SF Bay Area	Healthcare	No data	0.06	2022-07-25	Series E	United States	272
&Open	Dublin	Marketing	9	0.09	2022-11-17	Series A	Ireland	35
#Paid	Toronto	Marketing	19	0.17	2023-01-27	Series B	Canada	21

```sql
        '' AS funds_raised_millions

    UNION ALL

    -- 2. Header row
    SELECT
        'Company' AS company,
        'Location' AS location,
        'Industry' AS industry,
        'Total Laid Off' AS total_laid_off_display,
        'Percentage Laid Off' AS percentage_laid_off_display,
        'Date' AS `date`,
        'Stage' AS stage,
        'Country' AS country,
        'Funds Raised ($M)' AS funds_raised_millions

    UNION ALL

    UNION ALL

    -- 3. Your cleaned data – all rows kept, NULLs shown as 'No data'
    SELECT
        company,
        location,
        industry,
        IFNULL(total_laid_off, 'No data') AS total_laid_off_display,
        IFNULL(percentage_laid_off, 'No data') AS percentage_laid_off_display,
        `date`,
        stage,
        country,
        funds_raised_millions
    FROM layoffs_staging2;
```

18. Finally, I thought, why not to add some **data quality** measurement. And so…

```sql
ALTER TABLE layoffs_staging2
ADD COLUMN data_quality_score INT DEFAULT 0;

UPDATE layoffs_staging2
SET data_quality_score =
    (CASE WHEN total_laid_off IS NOT NULL THEN 1 ELSE 0 END) +
    (CASE WHEN percentage_laid_off IS NOT NULL THEN 1 ELSE 0 END) +
    (CASE WHEN industry IS NOT NULL AND industry != '' THEN 1 ELSE 0 END) +
    (CASE WHEN stage IS NOT NULL AND stage != 'Unknown' THEN 1 ELSE 0 END) +
    (CASE WHEN funds_raised_millions IS NOT NULL THEN 1 ELSE 0 END);

SELECT data_quality_score, COUNT(*)
FROM layoffs_staging2
GROUP BY 1
ORDER BY 1 DESC;

SELECT MAX(data_quality_score), MIN(data_quality_score)
FROM layoffs_staging2;
```

data_quality_score	COUNT(*)
5	954
4	829
3	178
2	34

19. Ahhh, and **created a view** just to simply underscore few last steps in returning one short command then

```sql
CREATE VIEW vw_layoffs_clean AS
SELECT
    company,
    TRIM(company) AS company_clean,
    location,
    industry,
    IFNULL(total_laid_off, 'No data') AS total_laid_off_display,
    IFNULL(percentage_laid_off, 'No data') AS percentage_laid_off_display,
    `date`,
    YEAR(`date`) AS year,
    MONTH(`date`) AS month,
    IFNULL(stage, 'Unknown') AS stage,
    country,
    IFNULL(funds_raised_millions, 'No data') AS funds_raised_millions,
    data_quality_score
FROM layoffs_staging2;
```

company	company_clean	location	industry	total_laid_off_display	percentage_laid_off_display	date	year	month	stage	country	funds_raised_millions	data_quality_sc
Included Health	Included Health	SF Bay Area	Healthcare	No data	0.06	2022-07-25	2022	7	Series E	United States	272	4
&Open	&Open	Dublin	Marketing	9	0.09	2022-11-17	2022	11	Series A	Ireland	35	5
#Paid	#Paid	Toronto	Marketing	19	0.17	2023-01-27	2023	1	Series B	Canada	21	5
100 Thieves	100 Thieves	Los Angeles	Consumer	12	No data	2022-07-13	2022	7	Series C	United States	120	4
10X Genomics	10X Genomics	SF Bay Area	Healthcare	100	0.08	2022-08-04	2022	8	Post-IPO	United States	242	5

20. Some calculation of difference cleaned data…

```sql
-- some calculation of difference removed dups
SELECT
    (SELECT COUNT(*) FROM layoffs) AS initial_data,
    (SELECT COUNT(*) FROM layoffs_staging2) AS after_cleaning_data,
    (SELECT COUNT(*) FROM layoffs) - (SELECT COUNT(*) FROM layoffs_staging2) AS removed_dups;
```

21. And, which might become handy in the future, some **automatization** to check for.

```sql
DELIMITER //
CREATE PROCEDURE CleanLayoffsData()
BEGIN
    DELETE t1 FROM layoffs_staging2 t1
    INNER JOIN layoffs_staging2 t2
    WHERE
        t1.company = t2.company AND
        t1.location = t2.location AND
        t1.`date` = t2.`date` AND
        t1.total_laid_off = t2.total_laid_off;

    UPDATE layoffs_staging2
    SET country = 'United States'
    WHERE country LIKE 'United States%';

    UPDATE layoffs_staging2 t1
    JOIN layoffs_staging2 t2 ON t1.company = t2.company
    SET t1.industry = t2.industry
    WHERE t1.industry IS NULL AND t2.industry IS NOT NULL;
END //
DELIMITER ;
```

- Did the whole data cleaning process: removed duplicates, standardized text, cleared NULLs and converted data types.
- "Added my own measurement system to verify data quality and assign the score (data_quality_score 0-5).
- Created analytical view and report that summarizes data cleaning process and the whole project.