Data Cleaning - SQL

DBMS System - **MySQL Workbench**
Database Name - **World_layoffs**
Table Name - **layoffs**
Worked on table - **layoffs_staging2**

Note - Field and Column are interchangeably used, as is the case in SQL.

Objective of the Project:

1. Remove duplicate rows, if any.
2. Standardize the data, for example, if we have any issues with spellings or spaces and fix errors.
3. Remove null values or blank values.
4. Remove any unnecessary columns or rows, if relevant.

Analysis:

It's not the best practice to work with raw data, so we create a replica of the original table to preserve the raw table.

Table Name - layoffs_staging

1. Removing Duplicates

We use the ROW_NUMBER() function with grouping all the individual columns. We use CTE to fetch those values whose row_number is greater than 1, indicating a duplicate value.

Issue - We cannot delete data, directly from a CTE. So, we create another table to transfer data obtained from CTE to the new table.

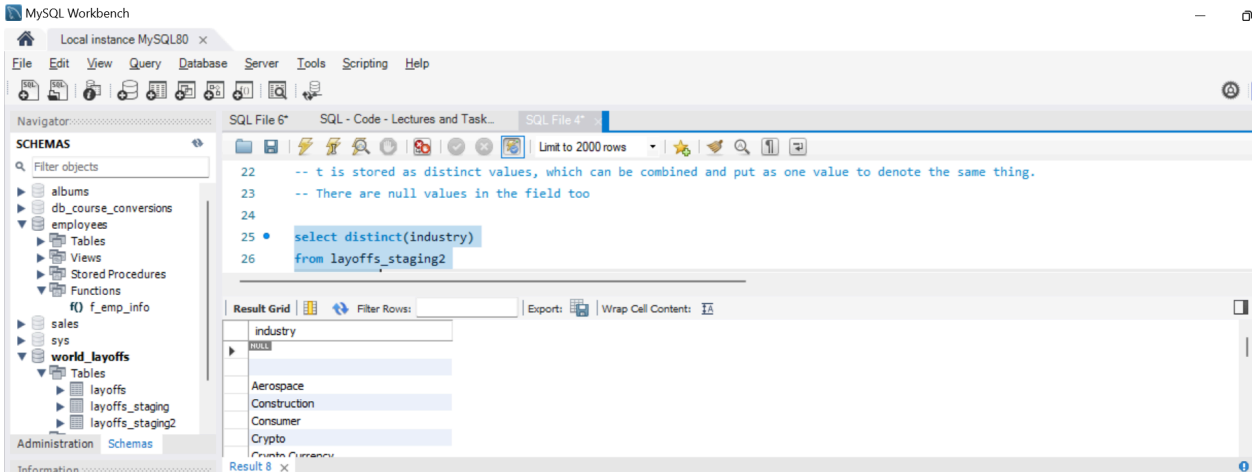
New Table - **layoffs_staging2**

We then use delete query to delete data from the new table, for row_num greater than 1, which are duplicate values.

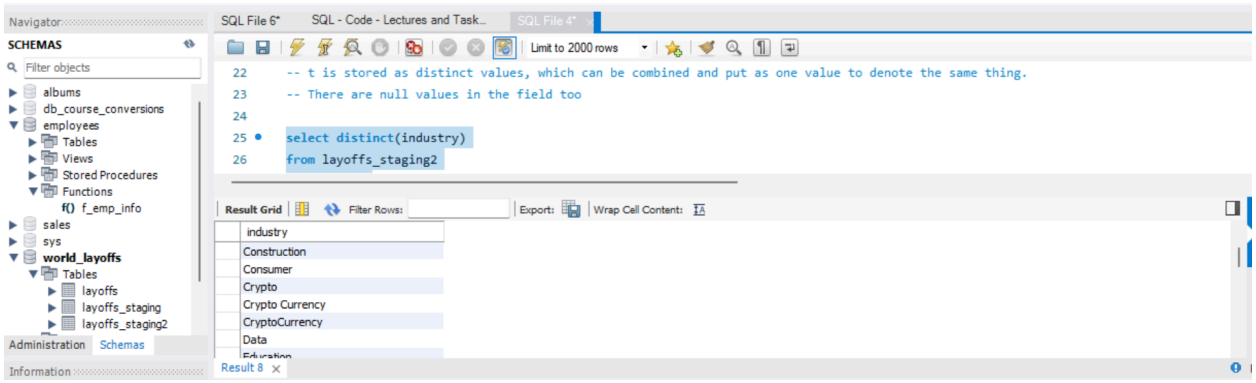
2. Standardizing Data

Below are the screenshots of the steps that came across during analysis.

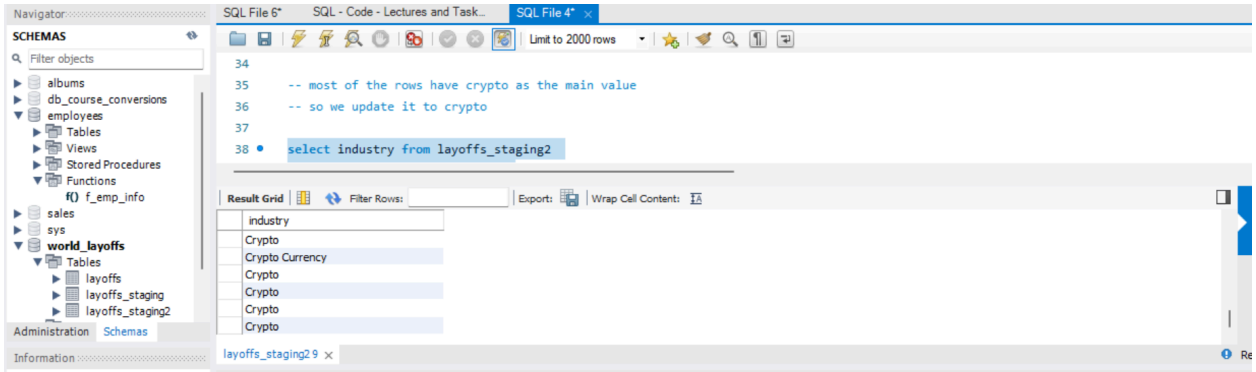
2.1 Issue - Null values encountered in the industry column of the dataset (solved in 3rd part)



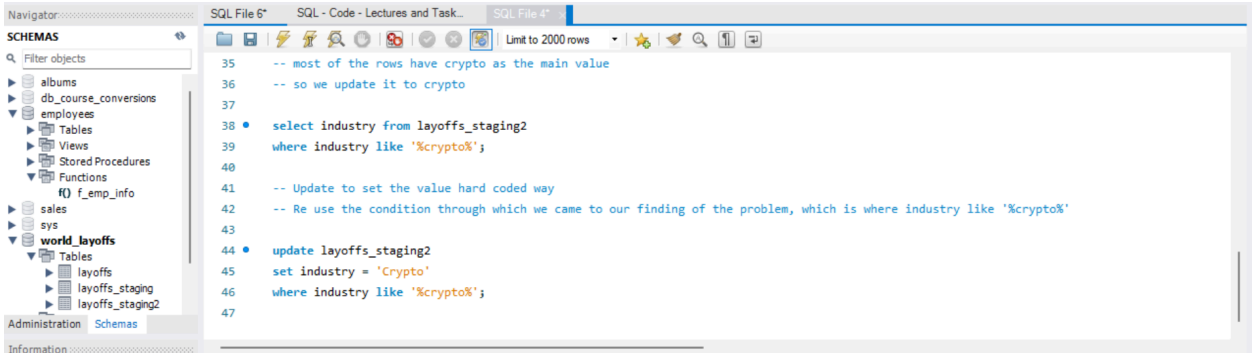
2.2 Issue - Distinct values have similar names



Crypto is the most common value, written as ‘Crypto Currency’ in a few places.

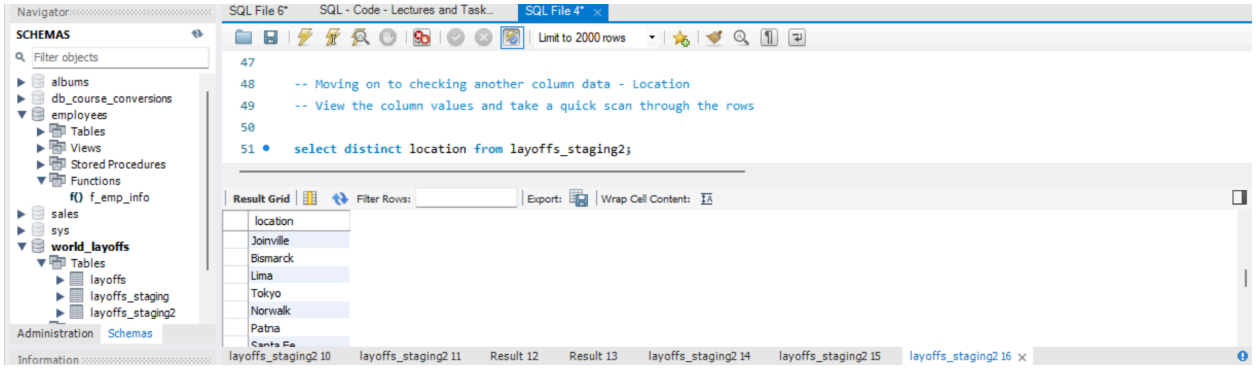


Solution - Change ‘Crypto Currency’ rows to the value of ‘Crypto’, as it’s more commonly used.



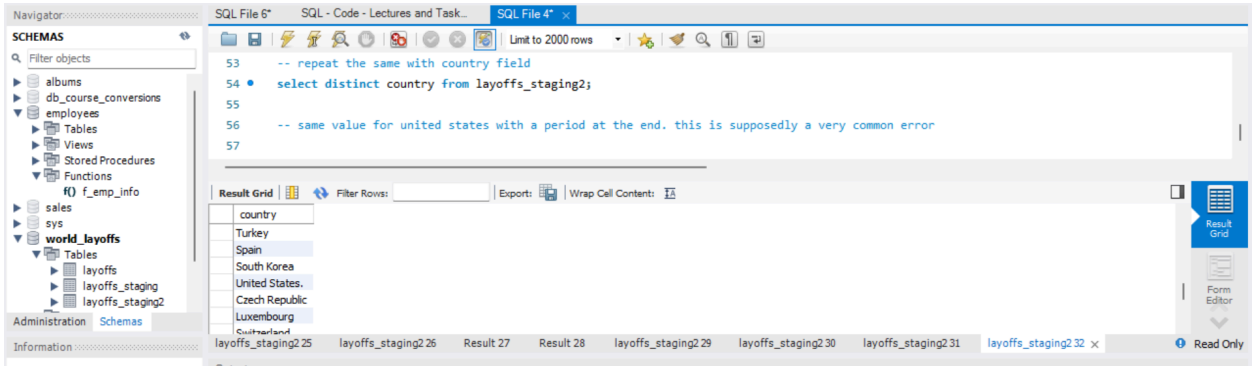
2.3 Checking data values of field - Location

Everything looks good here. No changes required.

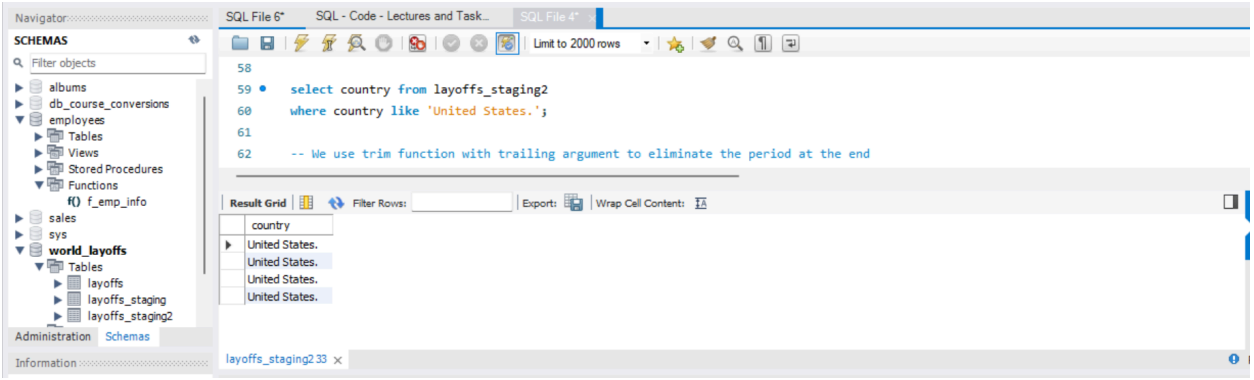


Next field - country

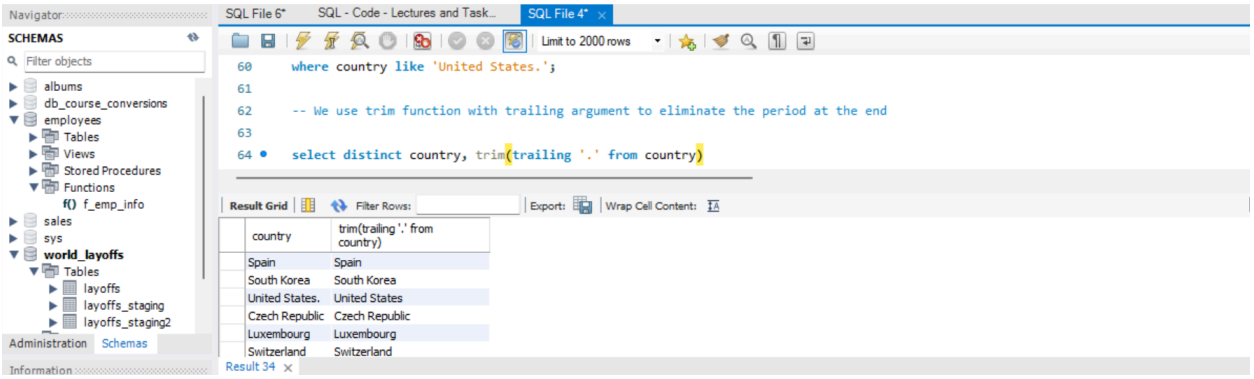
Issue - Two similar values encountered in the country field, namely - ‘United States’ and ‘United States.’.



Possible reasoning - Human error as only 4 rows has the error value.

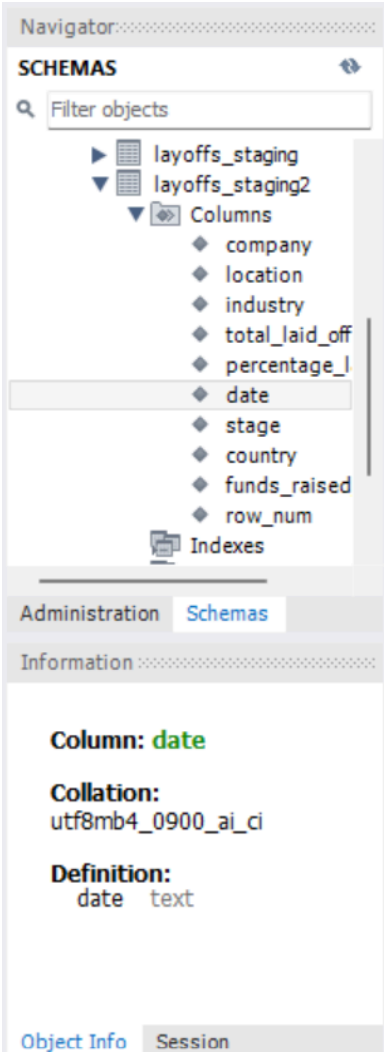


Solution - Trim function with trailing as an argument, to locate which part of the data is to be cleaned.

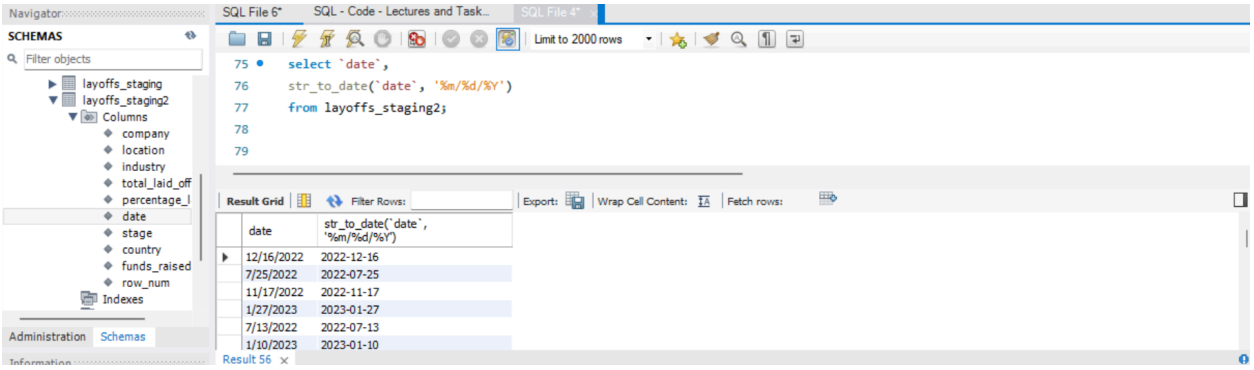


2.4 Checking data types of columns

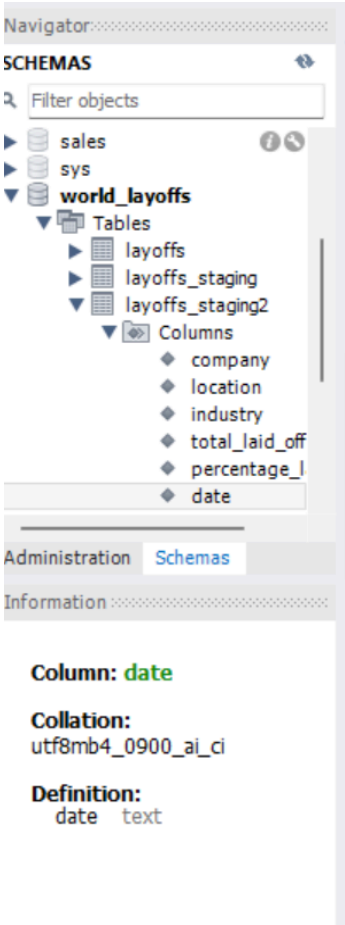
Troublemaker - Date field has data type of text, which will be a problem if we try to do time-series analysis during exploratory data analysis.



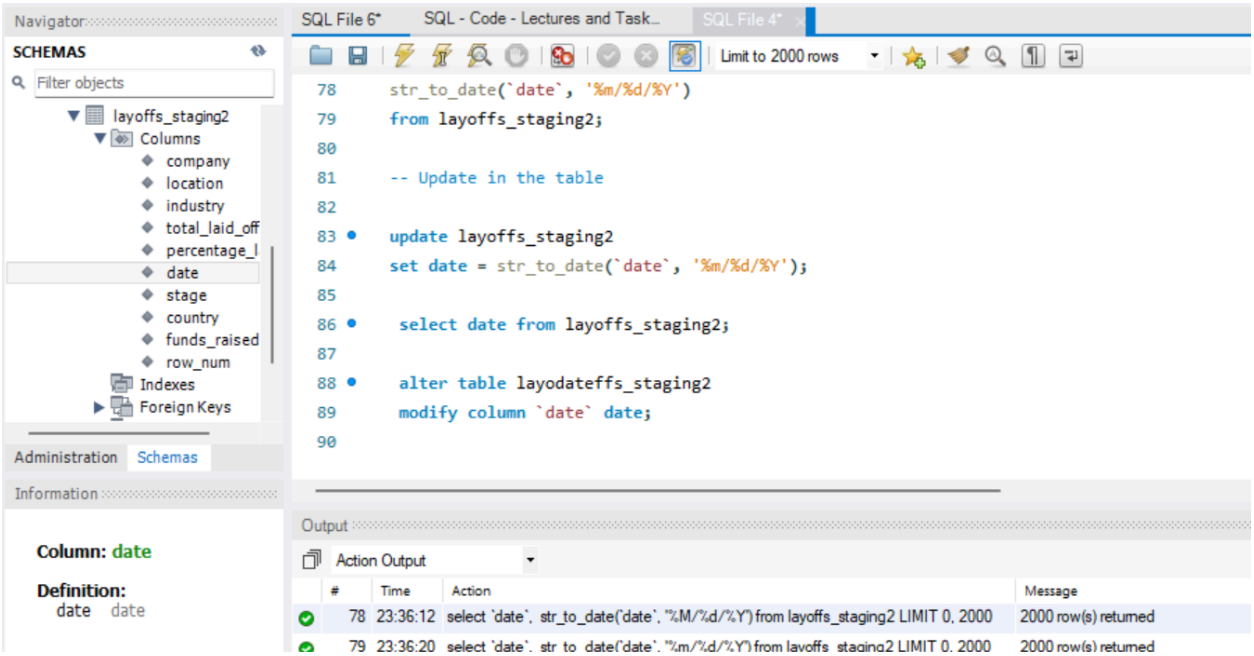
Approach - Let's change it to '`date`' data type using str_to_date() function.



To our surprise, the data type for `date` remains the same.

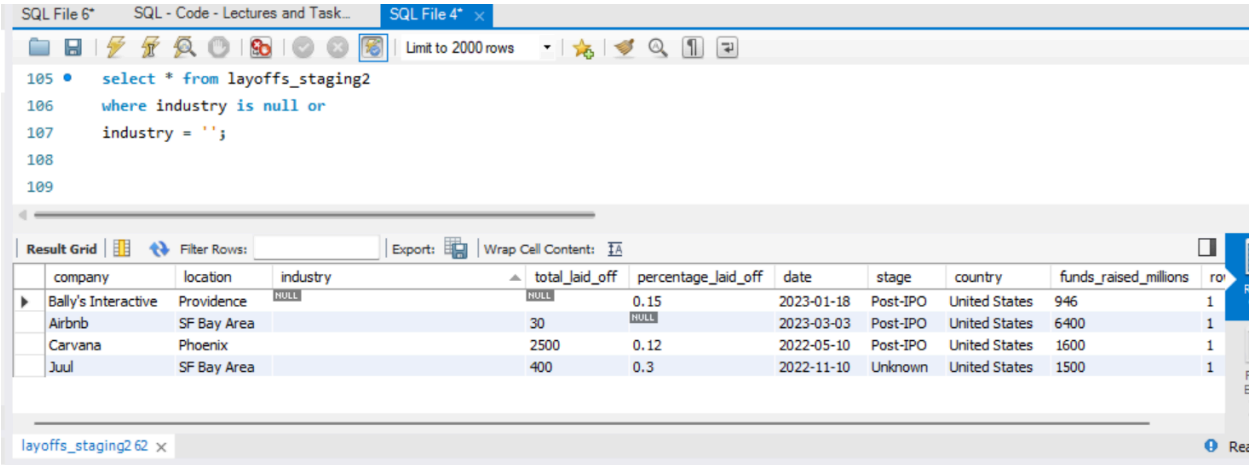


So, we bring in ‘ALTER TABLE’ to do the deed.

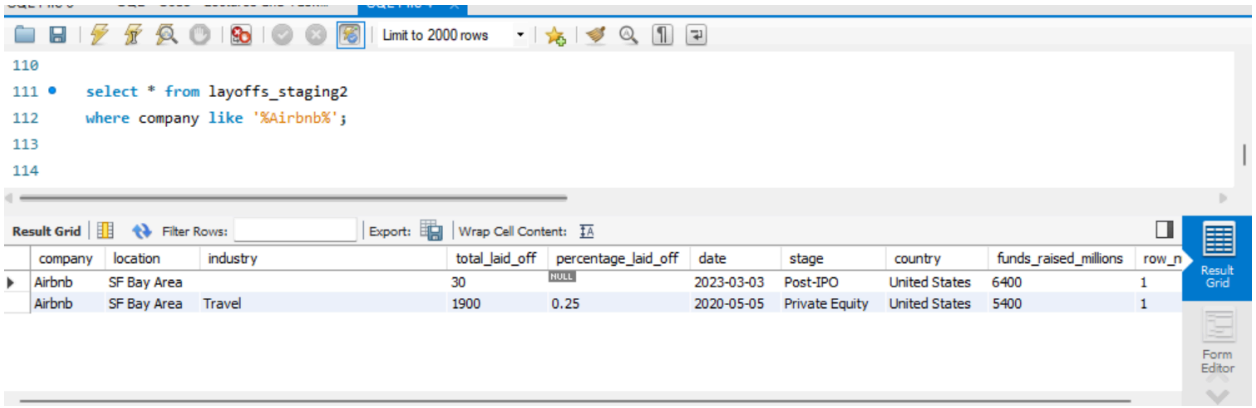


3. Removing Null Values

We have null values in the ‘industry’ field. So, we look at the rows which have null values in the industry column.



But we see that one row of the company - ‘Airbnb’, has the industry field mentioned as ‘Travel’. ‘Industry’ is a categorical and generic variable. So, we can update the field value from the populated row to row with **null** as value.



This issue is encountered with other companies too. So, we use self join to compare null values against the populated value.

But first, we change **BLANK** industry values to **NULL** values.

```sql
-- so, the wise step is to always set the blank industry value to null values before updating the changes

update layoffs_staging2
set industry = NULL
where industry = '';
```

industry	industry
	Travel
	Transportation
	Transportation
	Consumer

Resulting set

```sql
update layoffs_staging2
set industry = NULL
where industry = '';
```

industry	industry
NULL	Travel
NULL	Transportation
NULL	Transportation
NULL	Consumer

One company is still left with **null** value in the industry column, because it does not have a respective field with populated value, which is to say, it has one single row of data.

Bally's Interactive has no other row data where the industry is mentioned. So, we can't populate the industry field for this company. We let the null value be, as it is. Nothing can be done about it.

```
select * from layoffs_staging2
where industry is null;
```

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row
Bally's Interactive	Providence	NULL	NULL	0.15	2023-01-18	Post-IPO	United States	946	1

Issue - ‘total_laid_off’ and it’s dependent variable - ‘percentage_laid_off’ have null values too.

```
select *
from layoffs_staging2
where  total_laid_off is null and
percentage_laid_off is null;
```

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row
E Inc.	Toronto	Transportation	NULL	NULL	2022-12-16	Post-IPO	Canada	NULL	1
100 Thieves	Los Angeles	Retail	NULL	NULL	2023-01-10	Series C	United States	120	1
Accolade	Seattle	Healthcare	NULL	NULL	2023-03-03	Post-IPO	United States	458	1
Ada	Toronto	Support	NULL	NULL	2023-02-01	Series C	Canada	190	1
Adara	SF Bay Area	Travel	NULL	NULL	2020-03-31	Series C	United States	67	1
Addi	Bogota	Finance	NULL	NULL	2022-06-14	Series C	Colombia	376	1

Output

#	Time	Action	Message	Duration / Fetch
81	23:40:44	select `date`, str_to_date(`date`, '%m/%d/%Y') from layoffs_staging2 LIMIT 0, ...	2000 row(s) returned	0.000 sec / 0.000 sec
82	23:41:52	select date from layoffs_staging2 LIMIT 0, 2000	2000 row(s) returned	0.000 sec / 0.000 sec
83	23:44:48	alter table layoffs_staging2 modify column `date` date	2356 row(s) affected Records: 2356 Duplicates: 0 Warnings: 0	0.172 sec
84	23:46:10	select total_laid_off from layoffs_staging2 LIMIT 0, 2000	2000 row(s) returned	0.015 sec / 0.000 sec
85	23:49:00	select total_laid_off, percentage_laid_off from layoffs_staging2 LIMIT 0, 2000	2000 row(s) returned	0.000 sec / 0.000 sec
86	23:49:56	select * from layoffs_staging2 where total_laid_off is null and percentage_laid...	361 row(s) returned	0.000 sec / 0.000 sec

Since both the fields have **null** values in it, but the objective of the table is primarily concerned with the layoffs in various companies, and thus it makes less sense to keep those data, for which there’s no information about the layoffs.

So, we consider it as unnecessary data and delete it from both the fields.

```
-- Getting rid of inconsistent data
delete from layoffs_staging2
where   total_laid_off IS NULL
        AND percentage_laid_off IS NULL;
```

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

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_num

4. Remove unnecessary Rows or Columns

Finally, delete the row_num column from ‘layoffs_staging2’ table and our clean data is ready to be further processed, for exploratory data analysis.

```
-- Removing the row_num column from the table
alter table layoffs_staging2
drop column row_num;

select * from layoffs_staging2
```

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

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
Included Health	SF Bay Area	Healthcare	NULL	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
100 Thieves	Los Angeles	Consumer	12	NULL	2022-07-13	Series C	United States	120
10X Genomics	SF Bay Area	Healthcare	100	0.08	2022-08-04	Post-IPO	United States	242

layoffs_staging2 84