Comprehensive Data Cleaning and Transformation in SQL: A Full Project Walkthrough

Data Cleaning in MySQL | Full Project

134K views • 4 months ago

Alex The Analyst

In this lesson we are going to be building a **data cleaning project** in **MySQL**! Download Dataset: ...

Venkatakris

LINK: Data Cleaning in MYSQL by Alex the Analyst

OUTLINE:

1. Data loading

2. Removing Duplicates

3. Standardizing Data

4. Dropping Unnecessary Columns

5. Export to CSV

Business Requirement

The layoffs data file provided contains several issues that need to be addressed before it can be used for meaningful analysis. First, there are numerous missing values (NULLs) in key columns such as total_laid_off, percentage_laid_off, and industry, which could lead to inaccurate insights if not handled properly. Additionally, the dataset contains duplicate records, where the same company appears multiple times with identical or near-identical data. There are also inconsistencies in formatting across various columns, such as location and industry, where different naming conventions or incomplete entries are present.

Moreover, some unnecessary columns that do not contribute to the analysis need to be removed. To ensure the data is ready for business use, it is essential to clean, standardize, and remove any irrelevant or problematic entries, providing a final cleaned dataset in CSV format.

Problem Statement & Project Goals

• Problem Statement: The layoffs dataset has issues such as missing values, duplicate records, inconsistent data formats (e.g., location, industry, date fields), and unnecessary columns. These issues hinder the ability to generate reliable insights and visualizations.

• Project Goals: The goal of this project is to clean and standardize the layoffs data by: 1. Handling missing values (NULL).

2. Removing duplicate records.

3. Standardizing text fields and formatting.

4. Dropping unnecessary columns.

5. Delivering the cleaned data in CSV format.

Use Case

The cleaned dataset will allow the client to analyze trends in layoffs across industries and locations. By handling missing values and removing duplicates, the client will gain accurate insights into which sectors and regions are experiencing the most layoffs. Standardized data will also make it easier to create consistent reports and visualizations, leading to better decision-making.

Technical Requirements

The following are the technical tasks and requirements to meet the project goals. This includes the specific steps for data cleaning, the tools, and how the final product has been delivered.

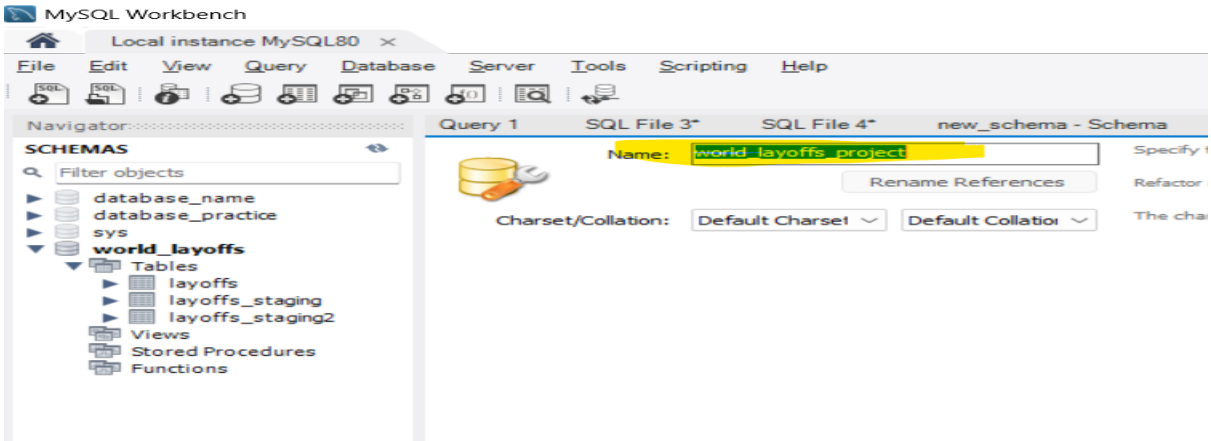
Technical Tasks:

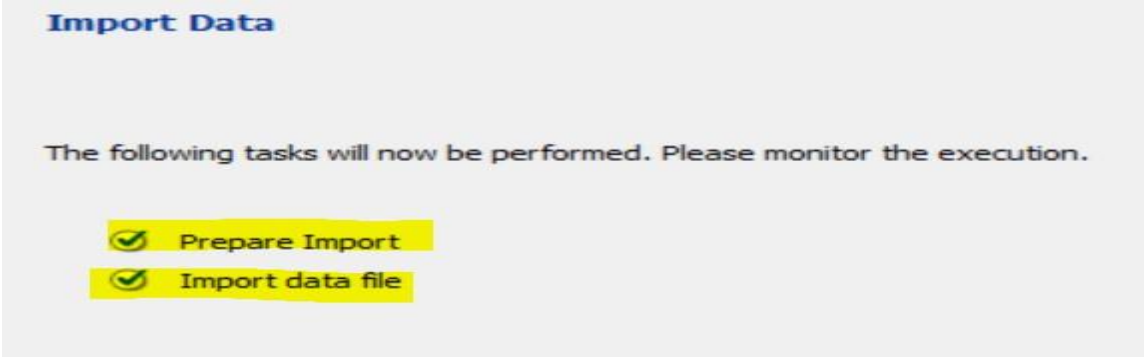
1. Data Loading:

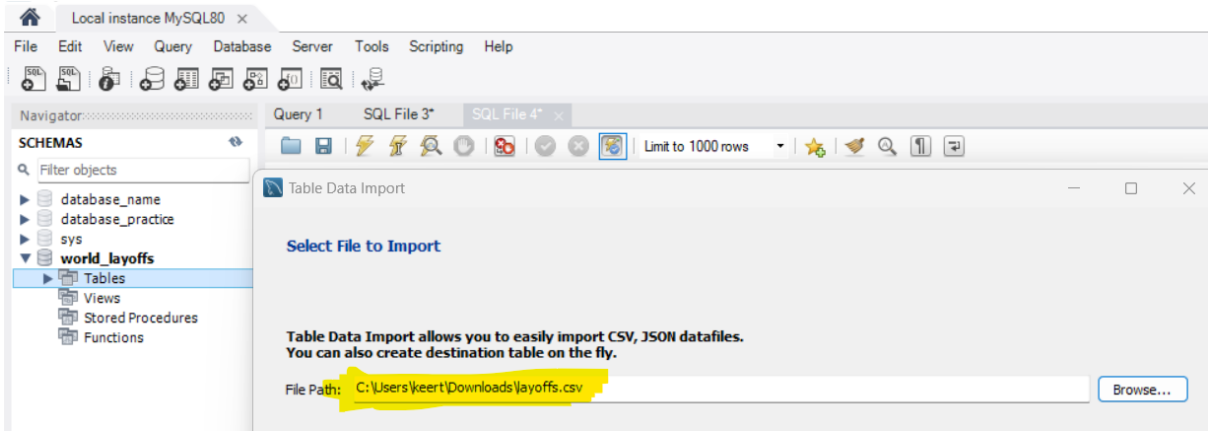
First, we will set up the schema for this project by creating a new schema named world_layoffs_project in MySQL Workbench, though this task can be completed using any SQL platform or tool. The schema will act as the structured environment where we will store

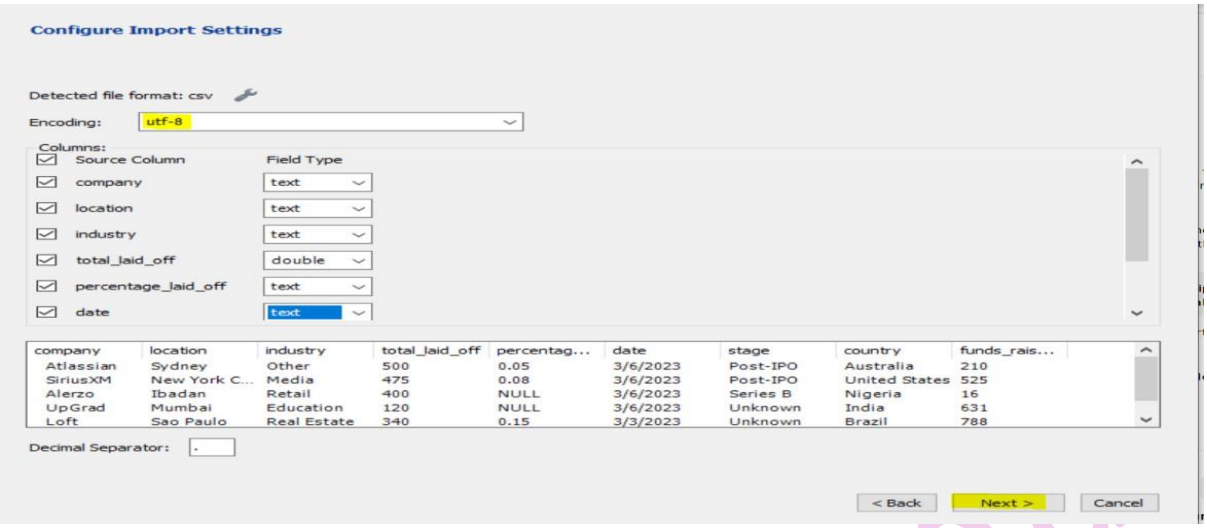
Venkatakrishna

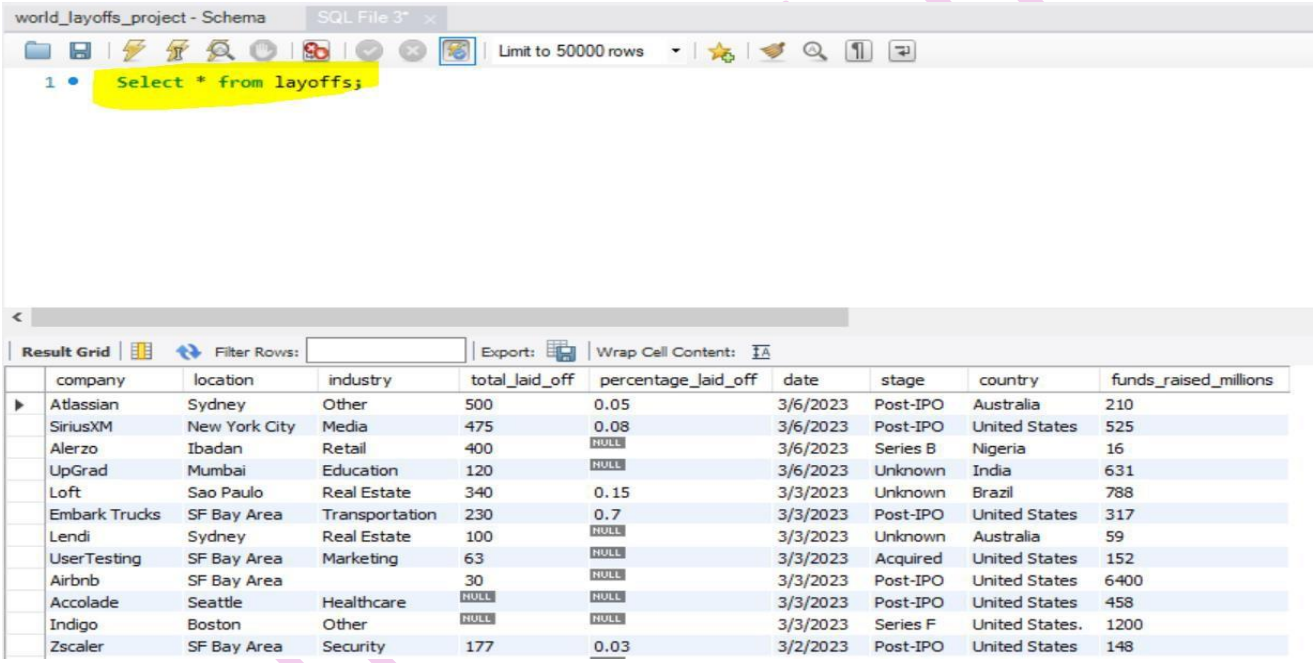
and manipulate 3 | Page the data. After creating the schema, we will load the provided layoffs dataset into it, ensuring that the table structure aligns with the raw data. This will allow us to begin the process of cleaning and transforming the data for further analysis.











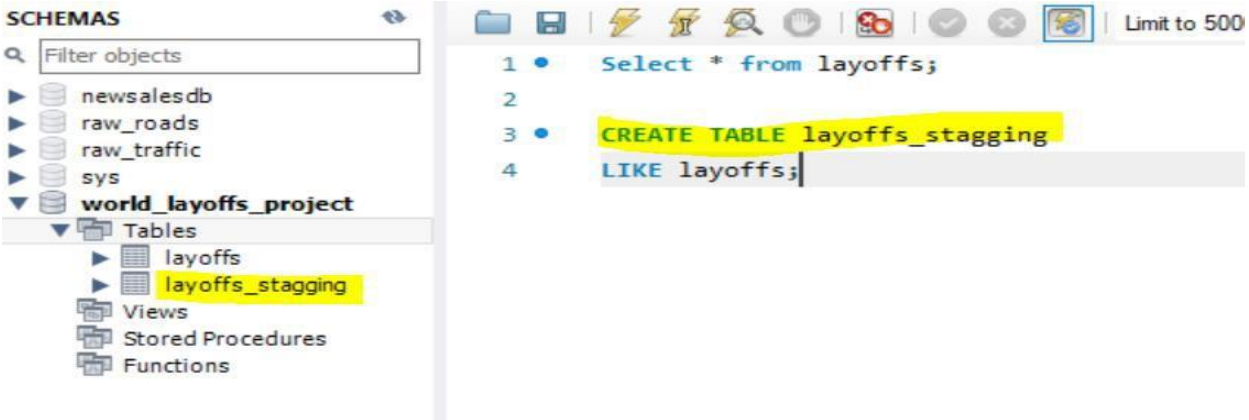
company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
Atlassian	Sydney	Other	500	0.05	3/6/2023	Post-IPO	Australia	210
SiriusXM	New York City	Media	475	0.08	3/6/2023	Post-IPO	United States	525
Alerzo	Ibadan	Retail	400	NULL	3/6/2023	Series B	Nigeria	16
UpGrad	Mumbai	Education	120	NULL	3/6/2023	Unknown	India	631
Loft	Sao Paulo	Real Estate	340	0.15	3/3/2023	Unknown	Brazil	788
Embark Trucks	SF Bay Area	Transportation	230	0.7	3/3/2023	Post-IPO	United States	317
Lendi	Sydney	Real Estate	100	NULL	3/3/2023	Unknown	Australia	59
UserTesting	SF Bay Area	Marketing	63	NULL	3/3/2023	Acquired	United States	152
Airbnb	SF Bay Area		30	NULL	3/3/2023	Post-IPO	United States	6400
Accolade	Seattle	Healthcare	NULL	NULL	3/3/2023	Post-IPO	United States	458
Indigo	Boston	Other	NULL	NULL	3/3/2023	Series F	United States.	1200
Zscaler	SF Bay Area	Security	177	0.03	3/2/2023	Post-IPO	United States	148

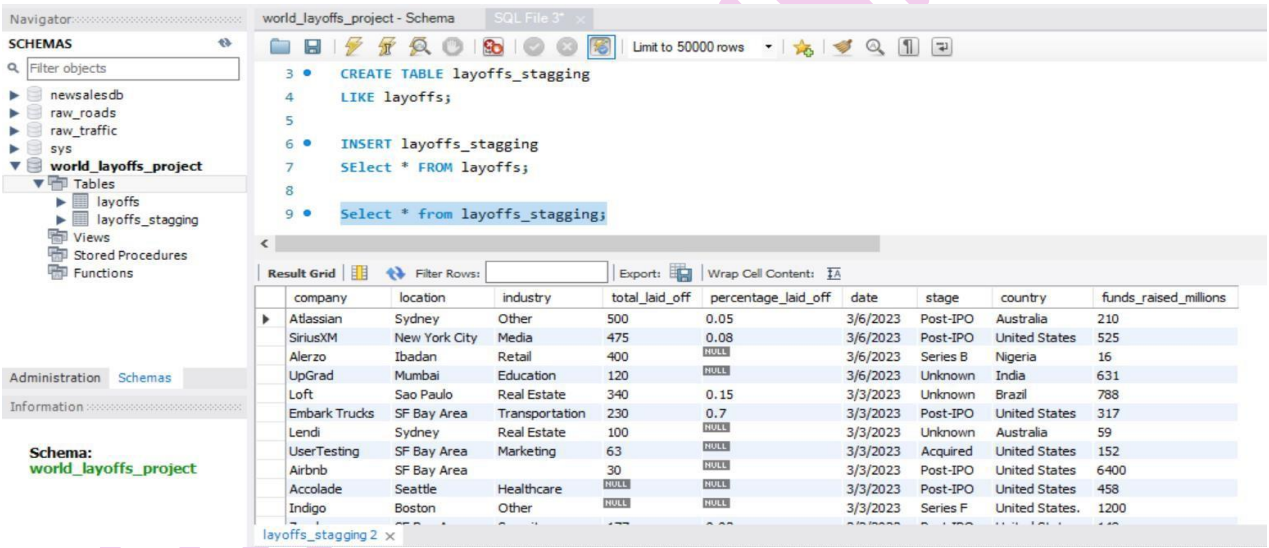
Staging Table:

The staging table is created because it provides a safe environment for data cleaning and manipulation. Working directly on the original data can be risky because any mistakes could lead to data loss or corruption. By using a staging table, you protect the raw data, ensuring you can always go back to the original version if something goes wrong.

The staging table allows you to experiment, clean, and transform data without worrying about damaging the original dataset. Once you're satisfied with the cleaned data, you can replace or update the original data if needed. It provides the flexibility to test multiple cleaning steps, rollback changes, and validate the results in an isolated environment before

applying them to the original dataset. In short, the staging table is necessary because it acts as a temporary workspace for performing potentially destructive tasks without endangering the original data.



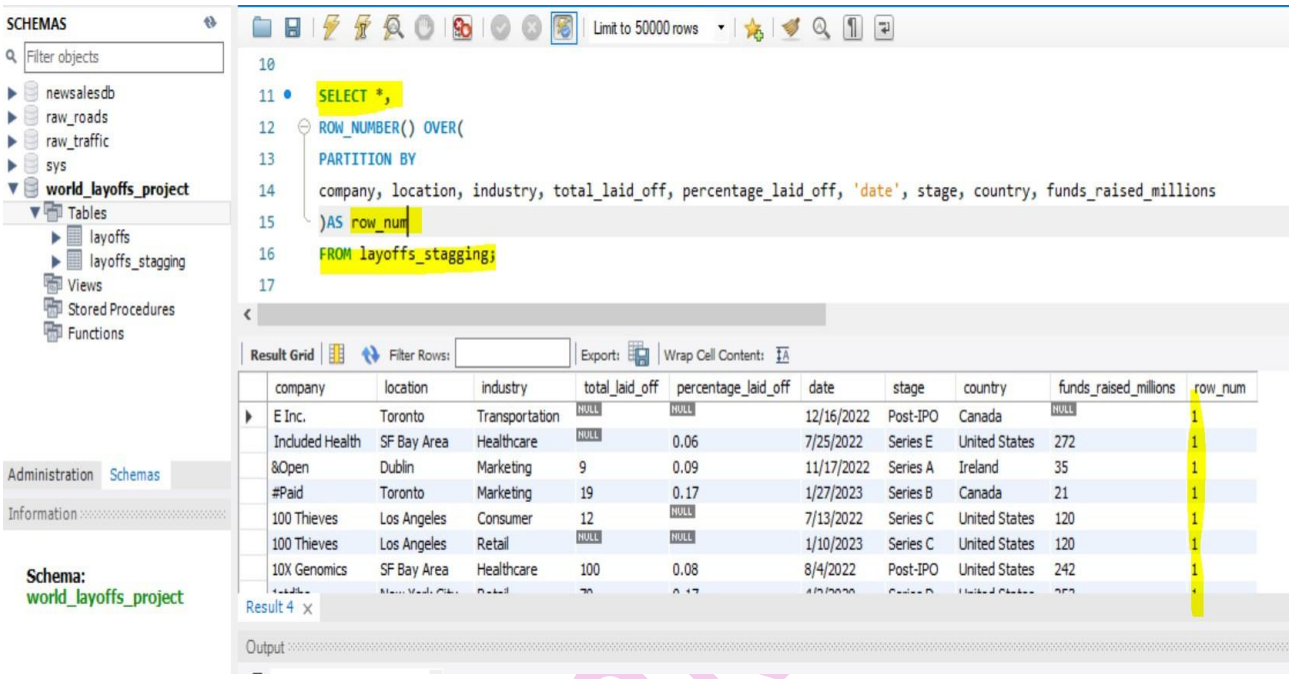


2. Removing Duplicates:

Identifying Duplicates in Dataset Using ROW_NUMBER() Function

The query is selecting the company, industry, total_laid_off, and date columns from the layoffs_staging table, while also adding a new column row_num that assigns a row number to each record. The ROW_NUMBER() function is used to generate a unique row number for each record within a partition. In this case, the partition is defined by company, industry, total_laid_off, and date. T his means that the row numbers will be reset within each combination of these values. The purpose of this query is likely to identify duplicates in the dataset. By using the ROW_NUMBER() function with the PARTITION BY clause, the query

helps to assign a unique number to each row within the same group of records that share the same values for company, industry, total_laid_off, and date. Records with a row_num greater than 1 are potential duplicates and can be flagged for removed



CTE for Identifying Potential Duplicate Records (duplicate_cte)

This query uses a Common Table Expression (CTE) named duplicate_cte to first compute a unique row number for each record in the layoffs_stagging table.

The ROW_NUMBER() function assigns this row number within each partition defined by multiple columns, including company, location, industry, total_laid_off, percentage_laid_off, date, stage, country, and funds_raised_millions. Records that share the same values across these columns are grouped together, and row numbers are assigned within each group. 7 | Page The outer query then selects records from the duplicate_cte where the row_num is greater than 1, effectively identifying and retrieving duplicate records based on the specified columns.

world_layoffs_project - Schema | SQL File 3*

Limit to 50000 rows

```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
)AS row_num
FROM layoffs_stagging
)
Select *
From duplicate_cte
where row_num > 1;
```

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
Better.com	New York City	Real Estate	NULL	NULL	8/26/2022	Unknown	United States	905	2
Casper	New York City	Retail	NULL	NULL	9/14/2021	Post-IPO	United States	339	2
Cazoo	London	Transportation	750	0.15	6/7/2022	Post-IPO	United Kingdom	2000	2
Elemy	SF Bay Area	Healthcare	NULL	NULL	12/5/2022	Series B	United States	323	2
ExtraHop	Seattle	Security	NULL	NULL	4/22/2020	Series C	United States	61	2
Hibob	Tel Aviv	HR	70	0.3	3/30/2020	Series A	Israel	45	2

Result 5

Creation of New Staging Table layoffs_stagging2 for Enhanced Data Handling

Created another staging table, **layoffs_stagging2**, is created to **improve data handling or** staging for further analysis or processing. We can also create through **right click** on layoffs_stagging, then click on **copy to Clipboard**, further click on **create statement** and **add column** named **row_num** with **INT** data type.

```sql
CREATE TABLE layoffs_stagging2 (
    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_num INT
);
```

```
INSERT INTO layoffs_stagging2
SELECT *,
ROW_NUMBER() OVER(
PARTITION BY
company, location, industry, total_laid_off, percentage_laid_off, 'date', stage, country, funds_raised_millions
)AS row_num
FROM layoffs_stagging;
```

Views
Stored Procedures
Functions

```
Select * From layoffs_stagging2
where row_num > 1;

DESCRIBE layoffs_stagging2;
```

Administration Schemas

Information

No object selected

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

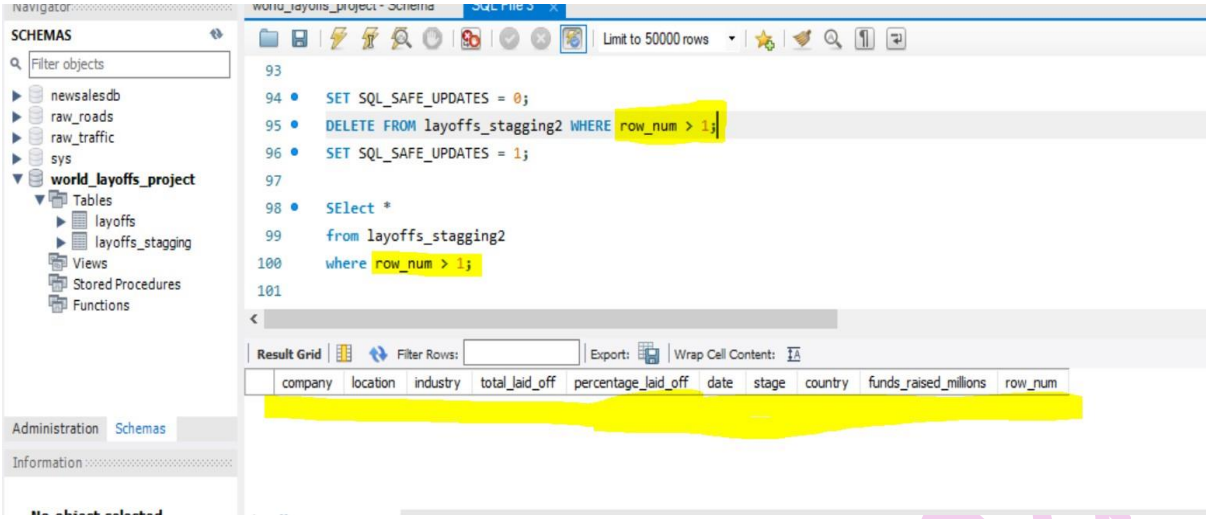
company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_num
Better.com	New York City	Real Estate	NULL	NULL	8/26/2022	Unknown	United States	905	2
Casper	New York City	Retail	NULL	NULL	9/14/2021	Post-IPO	United States	339	2
Cazoo	London	Transportation	750	0.15	6/7/2022	Post-IPO	United Kingdom	2000	2
Elemy	SF Bay Area	Healthcare	NULL	NULL	12/5/2022	Series B	United States	323	2
ExtraHop	Seattle	Security	NULL	NULL	4/22/2020	Series C	United States	61	2
Hibob	Tel Aviv	HR	70	0.3	3/30/2020	Series A	Israel	45	2
iFit	Logan	Fitness	NULL	NULL	12/8/2021	Private Equity	United States	200	2

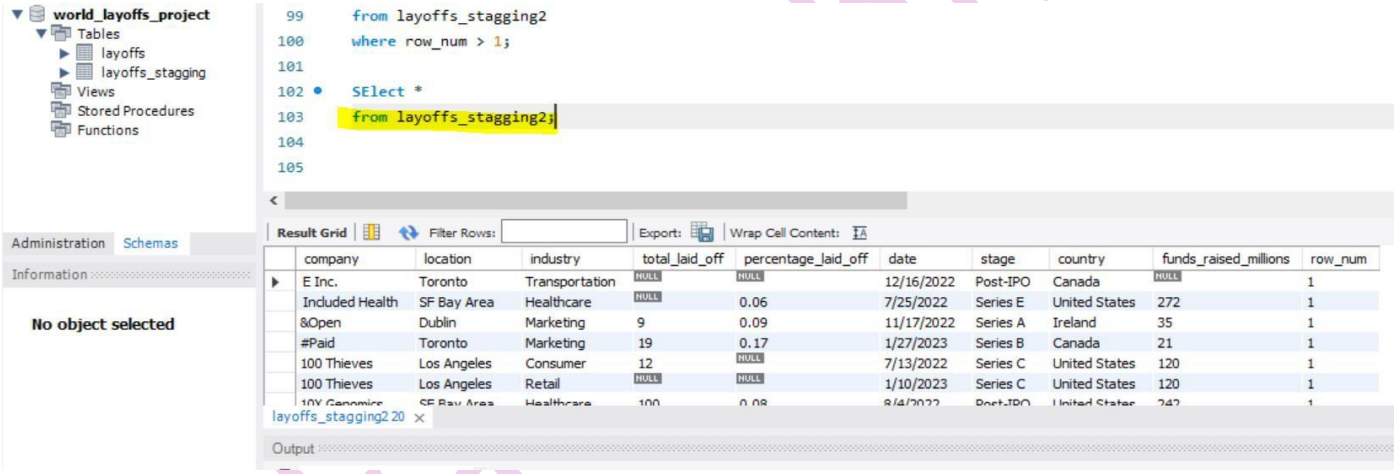
layoffs_stagging2 17

Deleting Duplicate Records from layoffs_stagging2 Table Based on row_num Values

The following query deletes **all records** that are considered **duplicates**, assuming that **row_num**

was assigned using a method that identifies duplicates by numbering them within each group. Records with **row_num values greater than 1** are typically the **duplicates** within those groups, so this query effectively cleans up the table by removing these duplicate entries.

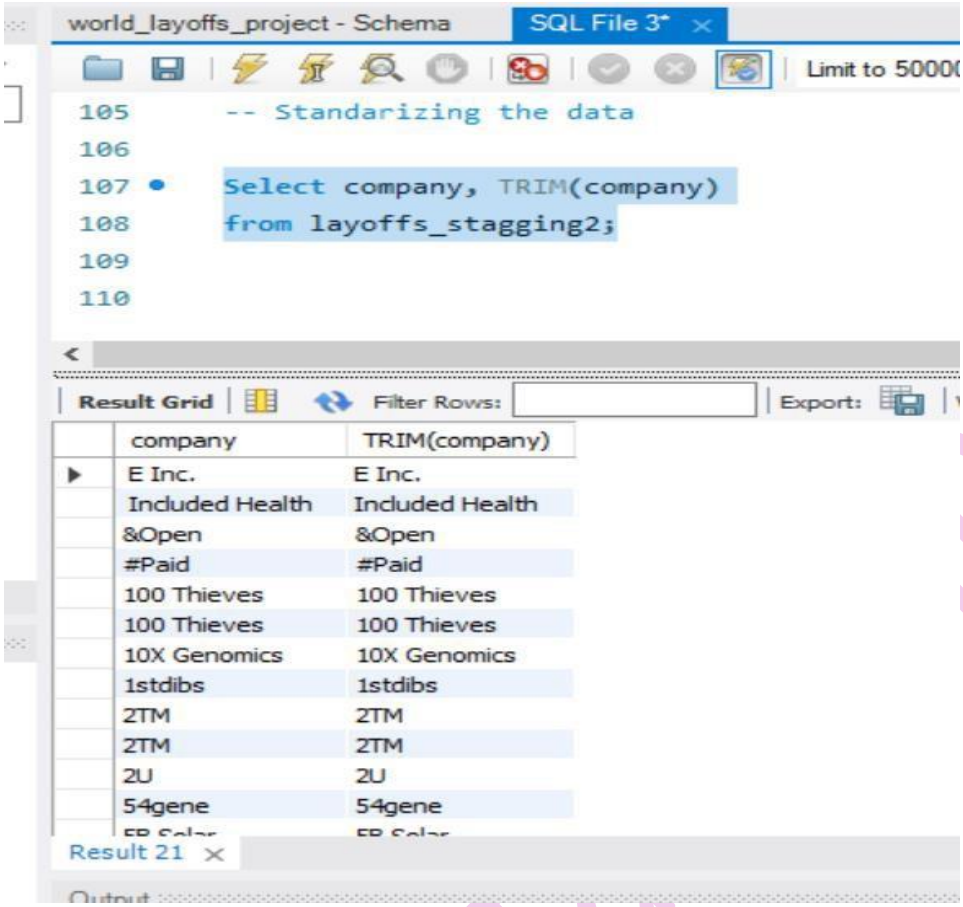




3. Standardizing Data

Removing Leading and Trailing Spaces from company Field Using TRIM() Function

The following query retrieves **two columns** from the layoffs_stagging2 table. The first column is the **company field** as it is stored in the **table**, while the **second column** is the same company field with any **leading or trailing spaces** removed. The **TRIM() function** is used to clean up the data by **removing these extra spaces**, which can help in **standardizing the company names** and improving data consistency. This query is useful for identifying and addressing any issues related to **unnecessary spaces** in the company field.



Standardizing Industry Values for Accurate Exploratory Data Analysis: Grouping Variants of 'Crypto' and 'Crypto Currency

The following **sql query** is used to **retrieve and sort unique values** from the industry column in the **layoffs_stagging2** table. During EDA, we noticed **inconsistencies** in the industry column, where some entries were listed as "**crypto**" and others as "**crypto currency**." To streamline the analysis and ensure that all related entries are **grouped correctly**, we standardized all such entries to "**crypto**." This approach ensures that the **exploratory analysis** is accurate and efficient, reducing the **potential for errors** and making the dataset easier to work with.

```sql
Select DISTINCT industry
from layoffs_stagging2
order by 1;
```

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

industry
NULL
Aerospace
Construction
Consumer
Crypto
Crypto Currency
CryptoCurrency

layoffs_stagging2 22

world_layoffs_project - Schema | SQL File 3*

Limit to 50000 rows

```sql
select *
from layoffs_stagging2
where industry
like 'crypto%';
```

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
2TM	Sao Paulo	Crypto	90	0.12	6/1/2022	Unknown	Brazil	250	1
2TM	Sao Paulo	Crypto	100	0.15	9/1/2022	Unknown	Brazil	250	1
Abra	SF Bay Area	Crypto	12	0.05	6/30/2022	Series C	United States	106	1
Amber Group	Hong Kong	Crypto	NULL	0.1	9/9/2022	Series B	Hong Kong	328	1
Autograph	Los Angeles	Crypto	NULL	NULL	12/16/2022	Series B	United States	205	1
Bakkt	Atlanta	Crypto	NULL	0.15	12/8/2022	Post-IPO	United States	932	1
Banxa	Melbourne	Crypto	70	0.3	6/27/2022	Post-IPO	Australia	13	1

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

company	location	industry	total_laid_off	percentage_laid_off	date
TaxBit	Salt Lake City	Crypto	NULL	NULL	12/13/2022
TIFIN	Boulder	Crypto	24	0.1	6/14/2022
Unchained C...	Austin	Crypto	NULL	0.15	11/18/2022
Unstoppable...	SF Bay Area	Crypto Currency	42	0.25	7/14/2022
WazirX	Mumbai	Crypto	60	0.4	10/2/2022
WeTrade	Bengaluru	Crypto	NULL	1	2/9/2023
Wyre	SF Bay Area	Crypto	NULL	1	1/4/2023
ZenLedger	Seattle	Crypto	NULL	0.1	12/9/2022

layoffs_stagging2 23

Output

```
Select DISTINCT industry
from layoffs_stagging2
order by industry;
```

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

industry
Aerospace
Construction
Consumer
Crypto
Data
Education
Energy
Fin-Tech
Finance
Fitness

layoffs_stagging2 30

Output

Cleaning and Updating 'country' Field: Removing Trailing Periods and Standardizing Entries

This query retrieves **all records** from the layoffs_stagging2 table where the country column starts with "**United States**" (including variations such as "United States of America"). The results are sorted based on the first column in the SELECT statement, which is typically country in this case.

```
select * from layoffs_stagging2
where country LIKE 'United States%'
order by 1;
```

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

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_rais
100 Thieves	Los Angeles	Consumer	12	NULL	7/13/2022	Series C	United States	120
100 Thieves	Los Angeles	Retail	NULL	NULL	1/10/2023	Series C	United States	120
10X Genomics	SF Bay Area	Healthcare	100	0.08	8/4/2022	Post-IPO	United States	242
1stdibs	New York City	Retail	70	0.17	4/2/2020	Series D	United States	253
2U	Washington D.C.	Education	NULL	0.2	7/28/2022	Post-IPO	United States	426
54gene	Washington D.C.	Healthcare	95	0.3	8/29/2022	Series B	United States	44
6sense	SF Bay Area	Sales	150	0.1	10/12/2022	Series E	United States	426
80 Acres Farms	Cincinnati	Food	NULL	0.1	1/18/2023	Unknown	United States	275
8x8	SF Bay Area	Support	155	0.07	1/18/2023	Post-IPO	United States	253
8x8	SF Bay Area	Support	200	0.09	10/4/2022	Post-IPO	United States	253
98point6	Seattle	Healthcare	NULL	0.1	7/21/2022	Series E	United States	247
Abra	SF Bay Area	Crypto	12	0.05	6/30/2022	Series C	United States	106
Absci	Vancouver	Healthcare	40	NULL	8/9/2022	Post-IPO	United States	237
Accolade	Seattle	Healthcare	NULL	NULL	3/3/2023	Post-IPO	United States	458
Acorns	Portland	Finance	50	NULL	5/26/2020	Unknown	United States	207
Actifio	Boston	Data	54	NULL	12/16/2020	Acquired	United States	352
ActiveCampaign	Chicago	Marketing	NULL	0.15	10/3/2022	Series C	United States	360

It retrieves **unique values** from the country column in the **layoffs_stagging2** table, while also removing any trailing periods ('.') from these values using the **TRIM() function**. The DISTINCT keyword ensures that only **unique country names** are listed, and the results are sorted based on the **first column**, which is country after trimming

```
Select distinct country, TRIM(TRAILING '.' from country)
from layoffs_stagging2
order by 1;
```

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

country	TRIM(TRAILING '.' from country)
Argentina	Argentina
Australia	Australia
Austria	Austria
Bahrain	Bahrain
Belgium	Belgium
Brazil	Brazil
Bulgaria	Bulgaria
Canada	Canada
Chile	Chile
China	China
Colombia	Colombia
Czech R...	Czech Republic
Denmark	Denmark
Egypt	Egypt

Result 32

This sequence of **commands updates** the country column in the **layoffs_stagging2** table for records where country starts with "**United States**." It removes any trailing periods ('.') from the country values.

```
SET SQL_SAFE_UPDATES = 0;
Update layoffs_stagging2
set country = TRIM(TRAILING '.' from country)
where country LIKE 'United States%';
SET SQL_SAFE_UPDATES = 1;

select * from layoffs_stagging2;
```

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

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_nu
E Inc.	Toronto	Transportation	NULL	NULL	12/16/2022	Post-IPO	Canada	NULL	1
Included Health	SF Bay Area	Healthcare	NULL	0.06	7/25/2022	Series E	United States	272	1
&Open	Dublin	Marketing	9	0.09	11/17/2022	Series A	Ireland	35	1
#Paid	Toronto	Marketing	19	0.17	1/27/2023	Series B	Canada	21	1
100 Thieves	Los Angeles	Consumer	12	NULL	7/13/2022	Series C	United States	120	1
100 Thieves	Los Angeles	Retail	NULL	NULL	1/10/2023	Series C	United States	120	1
10X Genomics	SF Bay Area	Healthcare	100	0.08	8/4/2022	Post-IPO	United States	242	1
1stdibs	New York City	Retail	70	0.17	4/2/2020	Series D	United States	253	1

layoffs_stagging2 33

Converting and Updating Date Format in layoffs_stagging2 Table Using STR_TO_DATE Function

This query selects the **date column** from the layoffs_stagging2 table and converts the date values from a string format **('%m/%d/%Y'**) to a proper date format using the **STR_TO_DATE() function**. It shows both the original string date and the converted date for comparison.

Furthermore, updates the **date column** in the layoffs_stagging2 table by converting the **string format of dates** to a proper date format using the **STR_TO_DATE()** function. This operation replaces the original string dates with properly formatted date values.

```
-- Update the date column by converting the string date to a date format
UPDATE
    layoffs_stagging2
SET
    date = STR_TO_DATE(date, '%m/%d/%Y');


-- Modify the column type of date to DATE
ALTER TABLE
    layoffs_stagging2
MODIFY COLUMN
    date DATE;
```

```sql
-- Select the date column and convert it using STR_TO_DATE function
SELECT
    date,
    STR_TO_DATE(date, '%m/%d/%Y')
FROM
    layoffs_stagging2;
```

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

date	STR_TO_DATE(date, '%m/%d/%Y')
12/16/2022	2022-12-16
7/25/2022	2022-07-25
11/17/2022	2022-11-17
1/27/2023	2023-01-27
7/13/2022	2022-07-13
1/10/2023	2023-01-10
8/4/2022	2022-08-04
4/2/2020	2020-04-02

Result 34

Output

Action Output

Properly Formatted Date

```sql
Select * from layoffs_stagging2;
```

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
E Inc.	Toronto	Transportation	NULL	NULL	2022-12-16	Post-IPO	Canada	NULL	1
Included Health	SF Bay Area	Healthcare	NULL	0.06	2022-07-25	Series E	United States	272	1
&Open	Dublin	Marketing	9	0.09	2022-11-17	Series A	Ireland	35	1
#Paid	Toronto	Marketing	19	0.17	2023-01-27	Series B	Canada	21	1
100 Thieves	Los Angeles	Consumer	12	NULL	2022-07-13	Series C	United States	120	1
100 Thieves	Los Angeles	Retail	NULL	NULL	2023-01-10	Series C	United States	120	1
10X Genomics	SF Bay Area	Healthcare	100	0.08	2022-08-04	Post-IPO	United States	242	1
1stdibs	New York City	Retail	70	0.17	2020-04-02	Series D	United States	253	1
2TM	Sao Paulo	Crypto	90	0.12	2022-06-01	Unknown	Brazil	250	1
2TM	Sao Paulo	Crypto	100	0.15	2022-09-01	Unknown	Brazil	250	1
2U	Washington	Education	NULL	0.2	2022-07-28	Post-IPO	United States	426	1

layoffs_stagging2 35

4. Dropping Unnecessary Columns

Identifying Records with Missing or Empty Values in layoffs_stagging2 Table

Retrieves all records from the **layoffs_stagging2** table where both the **total_laid_off** and **percentage_laid_off** columns have **NULL values**. This is useful for identifying records with **missing data** in these specific fields.

```
select *
from layoffs_stagging2
where total_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
E Inc.	Toronto	Transportation	NULL	NULL	2022-12-16	Post-IPO	Canada	NULL	1
Included Health	SF Bay Area	Healthcare	NULL	0.06	2022-07-25	Series E	United States	272	1
100 Thieves	Los Angeles	Retail	NULL	NULL	2023-01-10	Series C	United States	120	1
2U	Washington D.C.	Education	NULL	0.2	2022-07-28	Post-IPO	United States	426	1
5B Solar	Sydney	Energy	NULL	0.25	2022-06-03	Series A	Australia	12	1
80 Acres Farms	Cincinnati	Food	NULL	0.1	2023-01-18	Unknown	United States	275	1
98point6	Seattle	Healthcare	NULL	0.1	2022-07-21	Series E	United States	247	1
Accolade	Seattle	Healthcare	NULL	NULL	2023-03-03	Post-IPO	United States	458	1
ActiveCampaign	Chicago	Marketing	NULL	0.15	2022-10-03	Series C	United States	360	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
Affirm	SF Bay Area	Finance	NULL	0.01	2022-11-03	Post-IPO	United States	1500	1
Air	New York City	Marketing	NULL	0.16	2020-09-16	Series A	United States	18	1

layoffs_stagging2 37

```
select *
from layoffs_stagging2
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
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
AirMap	Los Angeles	Aerospace	NULL	NULL	2020-04-30	Unknown	United States	75	1
Airtasker	Sydney	Consumer	NULL	NULL	2022-07-04	Series C	Australia	26	1
Akerna	Denver	Logistics	NULL	NULL	2020-09-02	Post-IPO	United States	NULL	1
Akerna	Denver	Logistics	NULL	NULL	2022-05-27	Unknown	United States	46	1
Alegion	Austin	Data	NULL	NULL	2020-04-03	Series A	United States	16	1
Alerzo	Ibadan	Retail	NULL	NULL	2022-09-02	Series B	Nigeria	16	1
AllyO	SF Bay Area	HR	NULL	NULL	2020-04-03	Series B	United States	64	1
Almanac	SF Bay Area	Other	NULL	NULL	2022-08-13	Series A	United States	45	1

layoffs_stagging2 38

Output

The query is designed to **find and compare records** with **missing** or empty industry values against those with **non-null industry values** for the same company. This can help in identifying **inconsistencies** or gaps in the data where some entries are incomplete.

Performs the following actions:

1. **Joins the layoffs_stagging2 Table with Itself:** It uses a **self-join**, where t1 and t2 are two aliases for the same table.
2. **Join Condition:** It matches rows from t1 and t2 where the company column is the same in both aliases (t1.company = t2.company).
3. **Filter Conditions:**

- From t1, it selects rows where the **industry column is either NULL** or an empty string.
- From t2, it selects rows where the **industry column is not NULL**.

This results in pairs of records where the industry field is **missing or empty** in one record (t1) but has a value in a **corresponding record (t2)** for the same company.

```
SELECT t1.industry, t2.industry
FROM layoffs_stagging2 t1
JOIN layoffs_stagging2 t2
ON t1.company = t2.company
WHERE (t1.industry IS NULL OR t1.industry = '')
AND t2.industry IS NOT NULL;
```

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

industry	industry
	Travel
	Transportation
	Transportation
	Consumer

The query updates the **layoffs_stagging2** table by setting the industry value to the value from another row in the same table (where the company matches) only if the **industry value is NULL** in the current row and **not NULL in the other row**

```sql
UPDATE layoffs_stagging2 t1
JOIN layoffs_stagging2 t2
ON t1.company = t2.company
SET t1.industry = t2.industry
WHERE t1.industry IS NULL
AND t2.industry IS NOT NULL;
```

It allows you to **filter and retrieve specific rows** from the layoffs_stagging2 table based on a pattern in the company name. By using LIKE **'Bally%'**, you can quickly find and analyze all records where the company name starts with "**Bally**," which can be useful for Identifying and examining data related to specific companies that match the pattern, helping in focused analysis or reporting.

```sql
SELECT *
FROM layoffs_stagging2
WHERE company LIKE 'Bally%';
```

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
Bally's Interactive	Providence	NULL	NULL	0.15	2023-01-18	Post-IPO	United States	946	1

It **filters the data** to include only those rows where the company name matches **'Airbnb'** precisely.

```sql
Select * from layoffs_stagging2
where company = 'Airbnb';
```

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
Airbnb	SF Bay Area		30	NULL	2023-03-03	Post-IPO	United States	6400	1
Airbnb	SF Bay Area	Travel	1900	0.25	2020-05-05	Private Equity	United States	5400	1

Identifying and Removing Records with Missing Data in layoffs_stagging2 Table

This query selects and displays **all rows** from the layoffs_stagging2 table where both total_laid_off and percentage_laid_off columns are **NULL**. It helps in identifying records with **missing data** in these specific columns.

```sql
SELECT *
FROM layoffs_stagging2
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
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
AirMap	Los Angeles	Aerospace	NULL	NULL	2020-04-30	Unknown	United States	75	1
Airtasker	Sydney	Consumer	NULL	NULL	2022-07-04	Series C	Australia	26	1
Akerna	Denver	Logistics	NULL	NULL	2020-09-02	Post-IPO	United States	NULL	1
Akerna	Denver	Logistics	NULL	NULL	2022-05-27	Unknown	United States	46	1
Alegion	Austin	Data	NULL	NULL	2020-04-03	Series A	United States	16	1
Alerzo	Ibadan	Retail	NULL	NULL	2022-09-02	Series B	Nigeria	16	1
AllyO	SF Bay Area	HR	NULL	NULL	2020-04-03	Series B	United States	64	1

layoffs_stagging2 43

This query **deletes all rows** from the layoffs_stagging2 table where both total_laid_off and percentage_laid_off columns are **NULL**. It is used to **remove records** that have **no useful data** in these columns, effectively cleaning up the table by removing entries that **can't be ued for anlysis**.

Venkatakrishn

```
-- Delete Useless data we can't really use
DELETE FROM layoffs_stagging2
WHERE total_laid_off IS NULL
AND percentage_laid_off IS NULL;

SELECT *
FROM layoffs_stagging2;
```

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

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_nu
Included Health	SF Bay Area	Healthcare	NULL	0.06	2022-07-25	Series E	United States	272	1
&Open	Dublin	Marketing	9	0.09	2022-11-17	Series A	Ireland	35	1
#Paid	Toronto	Marketing	19	0.17	2023-01-27	Series B	Canada	21	1
100 Thieves	Los Angeles	Consumer	12	NULL	2022-07-13	Series C	United States	120	1
10X Genomics	SF Bay Area	Healthcare	100	0.08	2022-08-04	Post-IPO	United States	242	1
1stdibs	New York City	Retail	70	0.17	2020-04-02	Series D	United States	253	1
2TM	Sao Paulo	Crypto	90	0.12	2022-06-01	Unknown	Brazil	250	1
2TM	Sao Paulo	Crypto	100	0.15	2022-09-01	Unknown	Brazil	250	1
2U	Washington D.C.	Education	NULL	0.2	2022-07-28	Post-IPO	United States	426	1
54gene	Washington D.C.	Healthcare	95	0.3	2022-08-29	Series B	United States	44	1
5B Solar	Sydney	Energy	NULL	0.25	2022-06-03	Series A	Australia	12	1
6sense	SF Bay Area	Sales	150	0.1	2022-10-12	Series E	United States	426	1

layoffs_stagging2 44

```
UPDATE layoffs_stagging2
SET total_laid_off = 0
WHERE total_laid_off IS NULL;

UPDATE layoffs_stagging2
SET percentage_laid_off = 0
WHERE percentage_laid_off IS NULL;

SELECT *
FROM layoffs_stagging2;
```

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

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
Included Health	SF Bay Area	Healthcare	0	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	0	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
1stdibs	New York City	Retail	70	0.17	2020-04-02	Series D	United States	253
2TM	Sao Paulo	Crypto	90	0.12	2022-06-01	Unknown	Brazil	250
2TM	Sao Paulo	Crypto	100	0.15	2022-09-01	Unknown	Brazil	250
2U	Washington D.C.	Education	0	0.2	2022-07-28	Post-IPO	United States	426
54gene	Washington D.C.	Healthcare	95	0.3	2022-08-29	Series B	United States	44
5B Solar	Sydney	Energy	0	0.25	2022-06-03	Series A	Australia	12

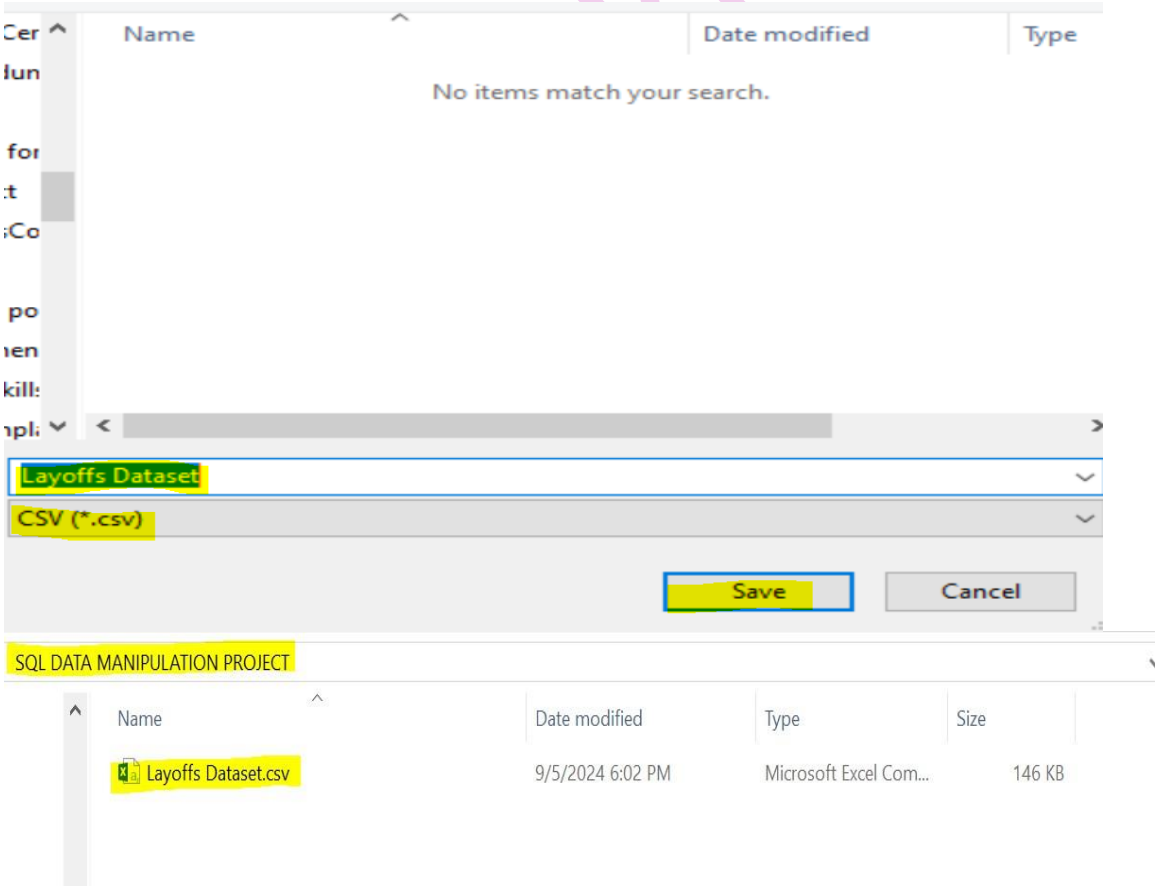
1. ***Exporting Cleaned Data:***

Export the cleaned and **standardized dataset** into a **CSV file** and share it with the **client.**

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

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
Included Health	SF Bay Area	Healthcare	0	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	0	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
1stdibs	New York City	Retail	70	0.17	2020-04-02	Series D	United States	253
2TM	Sao Paulo	Crypto	90	0.12	2022-06-01	Unknown	Brazil	250
2TM	Sao Paulo	Crypto	100	0.15	2022-09-01	Unknown	Brazil	250
2U	Washington D.C.	Education	0	0.2	2022-07-28	Post-IPO	United States	426
54gene	Washington D.C.	Healthcare	95	0.3	2022-08-29	Series B	United States	44
5B Solar	Sydney	Energy	0	0.25	2022-06-03	Series A	Australia	12
6sense	SF Bay Area	Sales	150	0.1	2022-10-12	Series E	United States	426
80 Acres Farms	Cincinnati	Food	0	0.1	2023-01-18	Unknown	United States	275
8x8	SF Bay Area	Support	155	0.07	2023-01-18	Post-IPO	United States	253
8x8	SF Bay Area	Support	200	0.09	2022-10-04	Post-IPO	United States	253
98point6	Seattle	Healthcare	0	0.1	2022-07-21	Series E	United States	247
99	Sao Paulo	Transportation	75	0.02	2022-09-20	Acquired	Brazil	244

layoffs_stagging2 45



Tools:

- **MySQL Workbench** for queries and code.
- CSV format for the final **data delivery**.

Deliverables

- A cleaned, standardized layoffs dataset in **CSV format**.
- **Documentation** explaining the steps taken and any **assumptions made** during the cleaning process.

Acknowledgement:

"I would like to express my heartfelt gratitude to **Alex The Analyst** for creating and sharing such valuable projects and resources. His detailed explanations and well-structured tutorials have been immensely helpful in expanding my understanding of data analytics and project workflows. By following his project, I was able to gain hands-on experience and successfully complete it.

Thank you, Alex, for your generosity and dedication to helping others in the data community. Your guidance has truly made a significant impact on my learning journey"

Venkatakrishna