

A black and white photograph of a man with dark hair, wearing a dark suit jacket over a light-colored shirt. He is seated at a desk, looking down intently at something on the surface. Behind him are vertical window blinds. The background is slightly blurred, showing what appears to be an office environment.

GLOBAL LAYOFFS

SQL Data Cleaning & EDA Project

Presented By **ANJALI BAJAJ**

Project Objectives:

Data Cleaning:

- **Created a Staging Table:** Established a staging table to preserve raw data while performing data cleaning operations.
- **Identified and Removed Duplicates:** Detected and eliminated duplicate records to ensure data uniqueness and integrity.
- **Standardized Data Entries:** Corrected inconsistencies and standardized values in key fields (e.g., industry names, country names) for uniformity.
- **Managed Missing Values:** Addressed and managed missing or NULL values to maintain data quality and usability.
- **Trimmed Unnecessary Data:** Removed irrelevant or redundant columns and rows to streamline the dataset.

Project Objectives:

Exploratory Data Analysis:

- **Trend Identification:** Analyzed the dataset to uncover significant insights, patterns, and outliers related to global layoffs.
- **Analyzed Layoffs by Company:** Examined and ranked companies based on the total number of layoffs and significant single-day layoffs.
- **Geographic and Industry Analysis:** Assessed layoffs by location and industry to understand regional and sector-specific impacts.
- **Temporal Trends:** Conducted a rolling analysis of layoffs over time to detect fluctuations and patterns.
- **Economic Correlation:** Analyzed how funds raised by companies relate to the percentage of layoffs to understand economic impacts.

Created a Staging Table `layoffs_staging`

To work on data cleaning while preserving the raw data.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the **SCHEMAS** section with `world_layoffs` selected. Under `Tables`, there are three entries: `layoffs`, `layoffs_staging`, and `layoffs_staging2`.
- SQL Editor:** The current tab is **Data Cleaning Project***. It contains the following SQL code:

```
8
9 • SELECT *
10 FROM world_layoffs.layoffs;
11
12
13
14 -- first thing we want to do is create a staging +
15
16 • CREATE TABLE world_layoffs.layoffs_staging
17 LIKE world_layoffs.layoffs;
18
19 • INSERT layoffs_staging
20 SELECT *
21 FROM world_layoffs.layoffs;
22
23
24 -- now when we are data cleaning we usually follow
25 -- 1. check for duplicates and remove any
```

The code is numbered from 8 to 25. Lines 14, 15, 24, and 25 are comments indicating steps in the data cleaning process.

Removed Duplicates

By checking them using the `ROW_NUMBER()` function.

The screenshot shows a SQL development environment with two main panes. The top pane displays a SELECT query to identify duplicates based on various company details and a row number. The bottom pane shows the resulting data grid and a subsequent DELETE query to remove rows where the row number is greater than 1.

```
SQL File 9* Data Cleaning Project* x Exploratory Data Analysis Project SQL File 10* SQL File 11* SQL File 12* SQL File 13*
```

```
68
69      -- these are our real duplicates
70
71 •   SELECT *
72 ⚒ FROM (
73     SELECT company, location, industry, total_laid_off, percentage_laid_off, `date`, stage, country,
74     funds_raised_millions,
75     ROW_NUMBER() OVER(
76         PARTITION BY company, location, industry, total_laid_off, percentage_laid_off, `date`, stage, country,
77         funds_raised_millions) AS row_num
78     FROM world_layoffs.layoffs_staging
79     ) duplicates
80     WHERE row_num > 1;
81
```

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_num
Casper	New York City	Retail	HULL	HULL	9/14/2021	Post-IPO	United States	339	2
Cazoo	London	Transportation	750	0.15	6/7/2022	Post-IPO	United Kingdom	2000	2
Hibob	Tel Aviv	HR	70	0.3	3/30/2020	Series A	Israel	45	2
Wildlife Studios	Sao Paulo	Consumer	300	0.2	11/28/2022	Unknown	Brazil	260	2
Yahoo	SF Bay Area	Consumer	1600	0.2	2/9/2023	Acquired	United States	6	2

```
SQL File 9* Data Cleaning Project* x Exploratory Data Analysis Project SQL File 10* SQL File 11* SQL File 12* SQL File 13*
```

```
131
132
133      -- now that we have this we can delete rows were row_num is greater than 2
134
135 •   DELETE
136     FROM world_layoffs.layoffs_staging2
137     WHERE row_num >= 2;
138
139
140
141
```

Standardized Data

- Industry - Populated NULLs from non-null values and standardized "Crypto".
- Country - Fixed errors by trimming trailing periods.
- Date - Standardized date format using `STR_TO_DATE()`.

```
SQL File 9* Data Cleaning Project* Exploratory Data Analysis Project SQL File 10*
195
196 -- now we need to populate those nulls if possible
197
198 • UPDATE layoffs_staging2 T1
199 JOIN layoffs_staging2 T2
200   ON T1.company = T2.company
201   SET T1.industry = T2.industry
202 WHERE T1.industry IS NULL
203 AND T2.industry IS NOT NULL;
204
205 -- and if we check it looks like Bally's was the only one without
206
207 • SELECT *
208 FROM world_layoffs.layoffs_staging2
209 WHERE industry IS NULL
210 OR industry = ''
211 ORDER BY industry;
212
213 -- -----
214
```

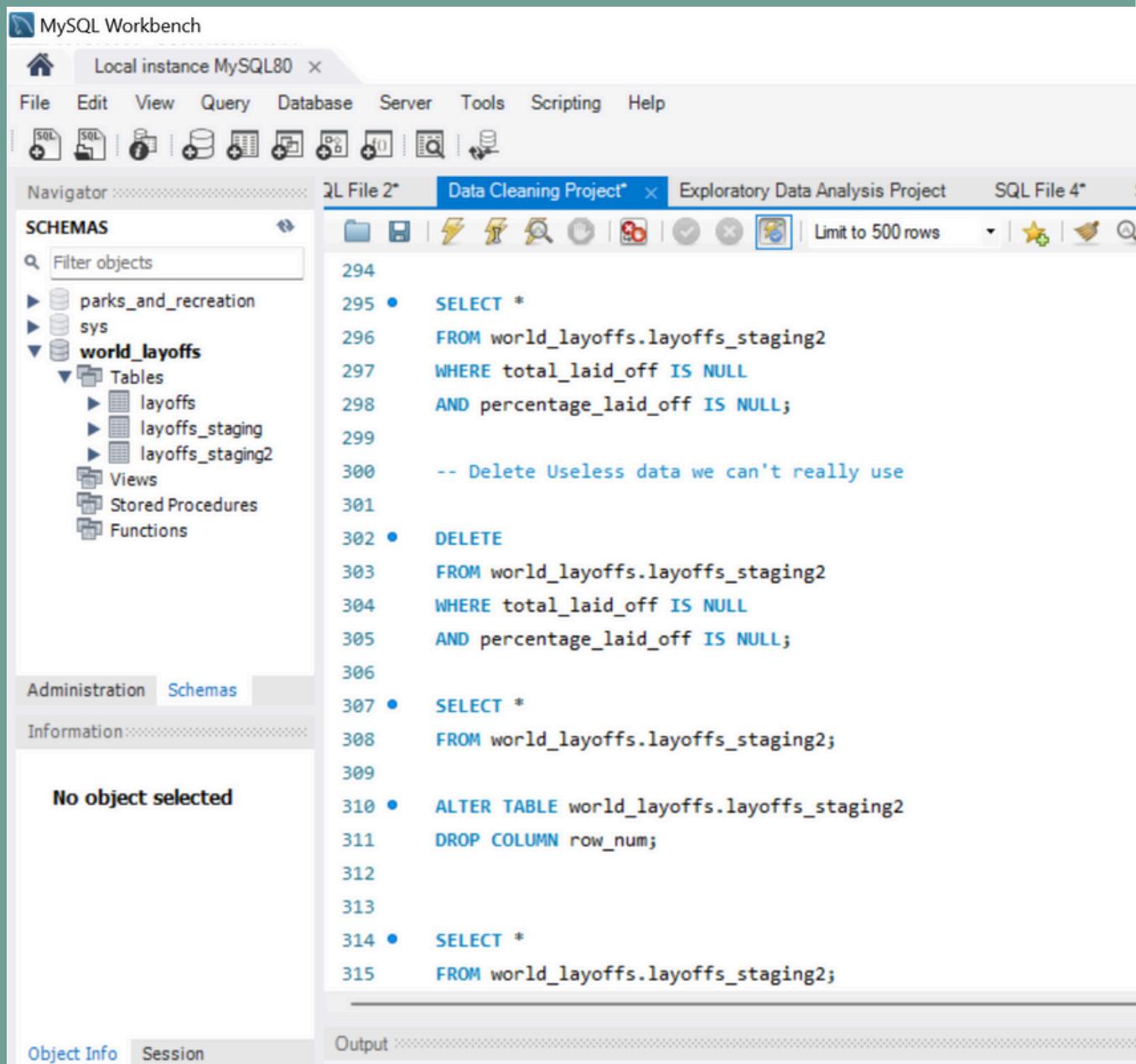
```
SQL File 9* Data Cleaning Project* Exploratory Data Analysis Project SQL File 10*
214
215 -- I also noticed the Crypto has multiple different variations
216
217 • SELECT industry
218 FROM world_layoffs.layoffs_staging2
219 ORDER BY industry;
220
221 • UPDATE world_layoffs.layoffs_staging2
222 SET industry = 'Crypto'
223 WHERE industry IN ('Crypto Currency', 'CryptoCurrency');
224
225 -- now that's taken care of:
226
227 • SELECT industry
228 FROM world_layoffs.layoffs_staging2
229 ORDER BY industry;
230
231 -- -----
```

```
Database Server Tools Scripting Help
SQL File 9* Data Cleaning Project* Exploratory Data Analysis Project SQL File 10*
237
238 -- everything looks good except apparently we have some "United States"
239
240 • SELECT DISTINCT country
241 FROM world_layoffs.layoffs_staging2
242 ORDER BY country;
243
244 • UPDATE world_layoffs.layoffs_staging2
245 SET country = TRIM(TRAILING '.' FROM country);
246
247 -- now if we run this again it is fixed
248
249 • SELECT DISTINCT country
250 FROM world_layoffs.layoffs_staging2
251 ORDER BY country;
252
253
254 -- Let's also fix the date column
255
```

```
Database Server Tools Scripting Help
SQL File 9* Data Cleaning Project* Exploratory Data Analysis Project SQL File 10*
254
255 -- Let's also fix the date column
256
257 • SELECT *
258 FROM world_layoffs.layoffs_staging2;
259
260 -- we can use str to date to update this field
261
262 • UPDATE world_layoffs.layoffs_staging2
263 SET `date` = STR_TO_DATE(`date`, '%m/%d/%Y');
264
265 -- now we can convert the data type properly
266
267 • ALTER TABLE world_layoffs.layoffs_staging2
268 MODIFY COLUMN `date` DATE;
269
270 • SELECT *
```

Removed:

- Rows with NULL values in both `total_laid_off` and `percentage_laid_off` columns, as they were deemed unusable.
- Column `row_num` after completing the deduplication process.



The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** MySQL Workbench - Local instance MySQL80
- Toolbar:** Standard MySQL Workbench toolbar with icons for Home, File, Edit, View, Query, Database, Server, Tools, Scripting, Help, and various file operations.
- Navigator:** Shows the database schema. Under the SCHEMAS tab, the 'world_layoffs' schema is selected, displaying its Tables (layoffs, layoffs_staging, layoffs_staging2), Views, Stored Procedures, and Functions.
- Information:** Shows 'No object selected'.
- Object Info:** Tab selected at the bottom left.
- Session:** Tab selected at the bottom left.
- Output:** Tab selected at the bottom right.
- SQL Editor:** The main area contains the following SQL code:

```
294
295 •  SELECT *
296   FROM world_layoffs.layoffs_staging2
297  WHERE total_laid_off IS NULL
298    AND percentage_laid_off IS NULL;
299
300  -- Delete Useless data we can't really use
301
302 •  DELETE
303   FROM world_layoffs.layoffs_staging2
304  WHERE total_laid_off IS NULL
305    AND percentage_laid_off IS NULL;
306
307 •  SELECT *
308   FROM world_layoffs.layoffs_staging2;
309
310 •  ALTER TABLE world_layoffs.layoffs_staging2
311   DROP COLUMN row_num;
312
313
314 •  SELECT *
315   FROM world_layoffs.layoffs_staging2;
```

Descriptive Analysis

Queried the maximum values of `total_laid_off` and `percentage_laid_off` to identify the highest layoffs and their impact.

The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** MySQL Workbench - Local instance MySQL80
- Toolbar:** Standard MySQL Workbench toolbar with icons for File, Edit, View, Query, Database, Server, Tools, Scripting, Help, and various connection and export options.
- Navigator:** Shows the database schema. Under the "world_layoffs" schema, there are three tables: layoffs, layoffs_staging, and layoffs_staging2. There are also Views, Stored Procedures, and Functions listed.
- Query Editor:** Displays two SQL queries numbered 17 and 24.

```
17
18 •   SELECT MAX(total_laid_off)
19     FROM world_layoffs.layoffs_staging2;
20
21
22     -- Looking at Percentage to see how big these layoffs were
23
24 •   SELECT MAX(percentage_laid_off), MIN(percentage_laid_off)
25     FROM world_layoffs.layoffs_staging2
26     WHERE percentage_laid_off IS NOT NULL;
27
28
29     -- Which companies had 1 is basically 100 percent of the company
30
```
- Result Grid:** Shows the results of the first query:

	MAX(total_laid_off)	MAX(percentage_laid_off)
▶	12000	1
- Status Bar:** Shows "No object selected" and "Object Info" tab is active.

Layoffs by Company

- Identified companies with the highest percentage of layoffs (close to 100%).
- Ordered companies by `funds_raised_millions` to see the financial impact of the layoffs.

The screenshot shows a SQL development environment with a toolbar at the top and several tabs below it. The main area displays a block of SQL code with line numbers. The code is used to find companies with 100% layoffs and then orders them by funds raised.

```
File 2* Data Cleaning Project* Exploratory Data Analysis Proj... × SQL File 4* SQL File 5* SQL File 5* SQL File 5*
```

```
29 -- Which companies had 1 is basically 100 percent of the company laid off
30
31 • SELECT *
32   FROM world_layoffs.layoffs_staging2
33   WHERE percentage_laid_off = 1;
34   -- these are mostly startups it looks like who all went out of business during this time
35
36   -- if we order by funds_raised_millions we can see how big some of these companies were
37
38 • SELECT *
39   FROM world_layoffs.layoffs_staging2
40   WHERE percentage_laid_off = 1
41   ORDER BY funds_raised_millions DESC;
42
```

Below the code, there is a result grid showing the data for the companies identified. The columns include company, location, industry, total_laid_off, percentage_laid, date, stage, country, and funds_raised_millions. The data shows several companies from the US and UK, with one entry from Australia.

company	location	industry	total_laid_off	percentage_laid	date	stage	country	funds_raised_millions
Britishvolt	London	Transportation	206	1	2023-01-17	Unknown	United Kingdom	2400
Quibi	Los Angeles	Media	NULL	1	2020-10-21	Private Equity	United States	1800
Deliveroo Australia	Melbourne	Food	120	1	2022-11-15	Post-IPO	Australia	1700
Katerra	SF Bay Area	Construction	2434	1	2021-06-01	Unknown	United States	1600
BlockFi	New York City	Crypto	NULL	1	2022-11-28	Series E	United States	1000
Aura Financial	SF Bay Area	Finance	NULL	1	2021-01-11	Unknown	United States	584

Aggregate Analysis

- Calculated top 5 total layoffs by company, location, country, year, industry, and company stage.
- Summarized total layoffs by grouping data by various attributes.

The screenshot shows the MySQL Workbench interface. On the left, there's a sidebar with tabs for 'Projects', 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'Tables' tab is selected, showing a list of tables including 'world_layoffs', 'layoffs_staging', and 'layoffs_staging2'. Below the sidebar, there are tabs for 'Session' and 'Schemas', with 'Session' currently active.

The main area has several tabs at the top: 'Data Cleaning Project*', 'Exploratory Data Analysis Proj...', 'SQL File 4*', 'SQL File 5*', and 'SQL File 6*'. The 'Exploratory Data Analysis Proj...' tab is active.

The query editor window contains the following SQL code:

```
54 •    SELECT company, total_laid_off
55      FROM world_layoffs.layoffs_staging2
56      ORDER BY 2 DESC
57      LIMIT 5;
58      -- now that's just on a single day
59
60      -- Companies with the most Total Layoffs
61
62 •    SELECT company, location, country, YEAR(date), industry, stage,
63        SUM(total_laid_off)
64      FROM world_layoffs.layoffs_staging2
65      GROUP BY company, location, country, YEAR(date), industry, stage
66      ORDER BY 4 DESC, 7 DESC
67      LIMIT 5;
```

Below the code, the 'Result Grid' shows the following data:

company	location	country	YEAR(date)	industry	stage	SUM(total_laid_off)
Google	SF Bay Area	United States	2023	Consumer	Post-IPO	12000
Microsoft	Seattle	United States	2023	Other	Post-IPO	10000
Ericsson	Stockholm	Sweden	2023	Other	Post-IPO	8500
Amazon	Seattle	United States	2023	Retail	Post-IPO	8000
Salesforce	SF Bay Area	United States	2023	Sales	Post-IPO	8000

Rank Analysis:

Ranked companies based on yearly layoffs using Common Table Expressions (CTEs) and tracked changes over time.

The screenshot shows a SQL development environment with two main panes. The left pane is a Navigator window titled "Data Cleaning Project" showing the schema structure. It includes sections for "SCHEMAS", "Tables", "Views", "Stored Procedures", and "Functions". The "world_layoffs" schema is expanded, showing tables like "layoffs", "layoffs_staging", and "layoffs_staging2". The right pane is the main workspace with a SQL editor tab titled "Exploratory Data Analysis Proj...". The code in the editor is a Common Table Expression (CTE) query:

```
108    -- I want to look at
109
110    WITH Company_Year AS
111    (
112        SELECT company, YEAR(`date`) AS years, SUM(total_laid_off) AS total_laid_off
113        FROM world_layoffs.layoffs_staging2
114        GROUP BY company, YEAR(`date`)
115    )
116    , Company_Year_Rank AS
117    (
118        SELECT company, years, total_laid_off, DENSE_RANK() OVER(
119            PARTITION BY years ORDER BY total_laid_off DESC) AS ranking
120            FROM Company_Year
121    )
122    SELECT company, years, total_laid_off, ranking
123    FROM Company_Year_Rank
124    WHERE ranking <= 3
125    AND years IS NOT NULL
126    ORDER BY years ASC, total_laid_off DESC;
```

Below the editor is a "Result Grid" pane showing the query results as a table:

	company	years	total_laid_off	ranking
▶	Uber	2020	7525	1
	Booking.com	2020	4375	2
	Groupon	2020	2800	3
	Bytedance	2021	3600	1
	Katerra	2021	2434	2
	Zillow	2021	2000	3
	Meta	2022	11000	1
	Amazon	2022	10150	2
	Cisco	2022	4100	3
	Google	2023	12000	1
	Microsoft	2023	10000	2
	Ericsson	2023	8500	3

The grid has 11 rows, labeled "Result 11".

Rolling Total Analysis:

Calculated the cumulative total of layoffs per month using a CTE to observe trends and fluctuations over time.

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator, which includes the Schemas section with 'world_layoffs' expanded to show 'Tables' (layoffs, layoffs_staging, layoffs_staging2), 'Views', 'Stored Procedures', and 'Functions'. The bottom-left pane shows 'No object selected'. The main center pane contains a query editor with the following SQL script:

```
130 -- Rolling Total of Layoffs Per Month
131
132 • SELECT SUBSTRING(`date`,1,7) AS dates, SUM(total_laid_off) AS total_laid_off
133     FROM world_layoffs.layoffs_staging2
134     GROUP BY dates
135     ORDER BY dates ASC;
136
137 -- now use it in a CTE so we can query off of it
138
139 • WITH DATE_CTE AS
140 (
141     SELECT SUBSTRING(`date`,1,7) AS dates, SUM(total_laid_off) AS total_laid_off
142     FROM world_layoffs.layoffs_staging2
143     GROUP BY dates
144     ORDER BY dates ASC
145 )
146     SELECT dates, total_laid_off, SUM(total_laid_off) OVER(
147     ORDER BY dates ASC) AS rolling_total_layoffs
148     FROM DATE_CTE
149     WHERE dates IS NOT NULL
150     ORDER BY dates ASC;
```

The bottom-right pane shows the 'Result Grid' with the following data:

dates	total_laid_off	rolling_total_layoffs
2020-03	9628	9628
2020-04	26710	36338
2020-05	25804	62142
2020-06	7627	69769
2020-07	7112	76881
2020-08	1969	78850
2020-09	609	79459
2020-10	450	79909
2020-11	237	80146
2020-12	852	80998
2021-01	6813	87811
2021-02	868	88679
2021-03	47	88726
2021-04	261	88987
2021-06	2434	91421
2021-07	80	91501
2021-08	1867	93368

Result 14 ×