PROJECT 2

EXPLORATORY DATA ANALYSIS

1. We start with analyzing the data.

```
-- EXPLORATORY DATA ANALYSIS

SELECT MAX(total_laid_off), MAX(percentage_laid_off)
FROM layoffs_staging2;

SELECT *
FROM layoffs_staging2
WHERE percentage_laid_off = 1
ORDER BY total_laid_off DESC;
```

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	fun
Katerra	SF Bay Area	Construction	2434	1	2021-06-01	Unknown	United States	160
Butler Hospitality	New York City	Food	1000	1	2022-07-08	Series B	United States	50
Deliv	SF Bay Area	Retail	669	1	2020-05-13	Series C	United States	80
Jump	New York City	Transportation	500	1	2020-05-07	Acquired	United States	11
SEND	Sydney	Food	300	1	2022-05-04	Seed	Australia	3

1.1. For instance, we can check which company had the **highest amount of total laid offs,** or which of them has the highest amount of **raised funds**.

```
SELECT *
FROM layoffs_staging2
WHERE percentage_laid_off = 1
ORDER BY funds_raised_millions DESC;
```

1.2. Or by **most laid offs per company.**

```
SELECT company, SUM(total_laid_off)
FROM layoffs_staging2
GROUP BY company
ORDER BY 2 DESC;
```

company	SUM(total_laid_off)
Amazon	18150
Google	12000

2. Let's check what is the actual **date range** we have in our database. 3 years, COVID period analyzed.

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

MIN(`date`)	MAX(`date`)
2020-03-11	2023-03-06

2.1. Here we can see the industries affected the most.

```
SELECT industry, SUM(total_laid_off)
FROM layoffs_staging2
GROUP BY industry
ORDER BY 2 DESC;
```

industry	SUM(total_laid_off)
Consumer	45182
Retail	43613
Other	36289
Transportation	33748
Finance	28344

2.2. Or countries.

```
SELECT country, SUM(total_laid_off)
FROM layoffs_staging2
GROUP BY country
ORDER BY 2 DESC;
```

country	SUM(total_laid_off)
United States	256559
India	35993
Netherlands	17220
Sweden	11264

2.3. Or show it **per year.**

```
30 •  SELECT YEAR(`date`), SUM(total_laid_off)
31    FROM layoffs_staging2
32    GROUP BY YEAR(`date`)
33    ORDER BY 2 DESC;
```

Result Grid | Filter Rows: | Export:

YEAR(`date`)	SUM(total_laid_off)
2022	160661
2023	125677
2020	80998
2021	15823
NULL	500

2.4. We can also check this data by **stage** column – means funding level, in which Post-IPO is for the biggest companies (Initial Public Offering).

```
35 •  SELECT stage, SUM(total_laid_off)
36    FROM layoffs_staging2
37    GROUP BY stage
38    ORDER BY 2 DESC;
39
```

Result Grid | Filter Rows: | Export:

stage	SUM(total_laid_off)
Post-IPO	204132
Unknown	40716
Acquired	27576
Series C	20017
Series D	19225
Series B	15311
Series E	12697

3. Let's say we want to see lay offs per **month and year**. We can use SUBSTRING to show the data.

```
44
45 •  SELECT SUBSTRING(`date`,1,7) AS `Month`, SUM(total_laid_of
46    FROM layoffs_staging2
47    WHERE SUBSTRING(`date`,1,7) IS NOT NULL
48    GROUP BY `Month`
49    ORDER BY 1 ASC;
50
51
52
```

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

Month	SUM(total_laid_off)
2020-03	9628
2020-04	26710
2020-05	25804
2020-06	7627
2020-07	7112
2020-08	1969
2020-09	609

Result 8

4. We want to look at **rolling lay offs** here. By that we can see the tendency how the lay offs been changing month to month and see the total (**rolling_total**) amount of it.

```
51 •  WITH Rolling_total AS
52    (
53    SELECT SUBSTRING(`date`,1,7) AS `Month`, SUM(total_laid_off) AS total_off
54    FROM layoffs_staging2
55    WHERE SUBSTRING(`date`,1,7) IS NOT NULL
56    GROUP BY `Month`
57    ORDER BY 1 ASC
58    )
59    SELECT `Month`, total_off,
60    SUM(total_off) OVER(ORDER BY `Month`) AS rolling_total
61    FROM Rolling_total;
62
```

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

Month	total_off	rolling_total
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

Result 11

5. With the next one we can see **which companies laid off how many people per year**.

```
63 •  SELECT company, YEAR(`date`), SUM(total_laid_off)
64    FROM layoffs_staging2
65    GROUP BY company, YEAR(`date`)
66    ORDER BY 1 ASC;
67
```

Result Grid | Filter Rows: | Export: | Wrap Cel

company	YEAR(`date`)	SUM(total_laid_off)
&Open	2022	9
#Paid	2023	19
100 Thieves	2022	12
10X Genomics	2022	100
1stdibs	2020	70
2TM	2022	190
2U	2022	NULL

Result 15

6. To the next – using CTEs we can check on our own what companies were top (had **highest laid offs**) per year.

```
70  WITH Company_year (Company, Years, Total_laid_offs) AS
71  (
72  SELECT company, YEAR(`date`), SUM(total_laid_off)
73  FROM layoffs_staging2
74  GROUP BY company, YEAR(`date`)
75  ORDER BY 3 deSC
76  )
77  SELECT *, DENSE_RANK() OVER(PARTITION BY Years ORDER BY Total_laid_offs DESC) AS Highest_laid_offs
78  FROM Company_year
79  WHERE Years IS NOT NULL
80  ORDER BY Highest_laid_offs;
81
```

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

Company	Years	Total_laid_offs	Highest_laid_offs
Uber	2020	7525	1
Bytedance	2021	3600	1
Meta	2022	11000	1
Google	2023	12000	1
Booking.com	2020	4375	2
Katerra	2021	2434	2
Amazon	2022	10150	2

Result 23