```
create database layoff_world;

use layoffs_world;


-- Exploratory Data Analysis


-- REMOVE DUPLICATES

-- STANDARDIZE THE DATA spellings etc

-- Null values Blank values

-- remove columns rows not necessary


CREATE TABLE layoffs_staging

LIKE layoffs;


insert layoffs_staging

select *

from layoffs;


-- Using window funtion row_num to check first and then remove any possible duplicate


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_staging

)

DELETE

FROM duplicate_cte
```

```
WHERE row_num >1;


drop table layoffs_staging2;

create TABLE `layoffs_staging2` (

`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

) ENGINE=InnoDB default CHARSET=utf8mb4 collate=utf8mb4_0900_ai_ci;


select *

from layoffs_staging2

where row_num >1;


Insert into layoffs_staging2

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_staging;


DELETE

FROM layoffs_staging2
```

```
WHERE row_num > 1;


select *

from layoffs_staging2;


-- Standardizing data


SELECT company, trim(company)

FROM layoffs_staging2;


UPDATE layoffs_staging2

SET company = trim(company);


SELECT *

FROM layoffs_staging2

WHERE industry LIKE 'Crypto%';


UPDATE layoffs_staging2

SET industry = 'Crypto'

WHERE industry LIKE 'Crypto%';


select distinct location

FROM layoffs_staging2

order by 1;


-- In the Country column , USA. how to deal with it using Trailing

select distinct country

FROM layoffs_staging2

order by 1;
```

```
select distinct(country), trim(country)
from layoffs_staging2
where country like 'United States%'
order by 1;

select distinct(country), trim(trailing '.' from country)
from layoffs_staging2
order by 1;

UPDATE layoffs_staging2
SET country = trim(trailing '.' from country)
WHERE country like 'United States%';

SELECT *
FROM layoffs_staging2;

-- Formating Date from text

SELECT `date`,
str_to_date(`date`, '%m/%d/%Y')
from layoffs_staging2;

UPDATE layoffs_staging2
SET `date` = str_to_date(`date`, '%m/%d/%Y');



ALTER TABLE layoffs_staging2
MODIFY COLUMN `date` DATE;
```

```
-- NULL AND BLANKS

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

SELECT *
FROM layoffs_staging2
WHERE industry is NULL
or industry = '';

-- I got 2 data for 'Airbnb' one of which has not an industry. Lets use a Self-Join to apply the changes

SELECT *
FROM layoffs_staging2
WHERE company = 'Airbnb';

UPDATE layoffs_staging2
SET industry = null
where industry = '';

select *
from layoffs_staging2 a
join layoffs_staging2 b
            on a.company = b.company
      AND a.location = b.location
WHERE a.industry IS NULL
```

```
AND b.industry IS NOT NULL;



UPDATE layoffs_staging2 a
JOIN layoffs_staging2 b
        on a.company = b.company
SET a.INDUSTRY = b.industry
WHERE a.industry IS NULL
AND b.industry IS NOT NULL;

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

SELECT * FROM layoffs_staging2;

-- DROPING TA ROW NUM

ALTER TABLE layoffs_staging2
DROP COLUMN row_num;

-- 02 - 05 - 2024

SELECT * FROM layoffs_staging2;

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

```
SELECT * FROM layoffs_staging2

where percentage_laid_off = 1

order by funds_raised_millions desc;


select YEAR(`date`), SUM(total_laid_off)

FROM layoffs_staging2

GROUP BY YEAR(`date`)

ORDER BY 1 DESC;


SELECT MIN(`date`),MAX(`date`)

FROM layoffs_staging2;



-- isws ta post-ipo einai ta big companies google ktlp


select stage, SUM(total_laid_off)

FROM layoffs_staging2

GROUP BY stage

ORDER BY 2 DESC;


-- Lets check the progression of layoffs yearly and by company

-- Firstly i will extract the Month from the date column


SELECT SUBSTRING(`date`,1,7) AS `MONTH`, SUM(total_laid_off)

FROM layoffs_staging2

WHERE SUBSTRING(`date`,1,7) IS NOT NULL

GROUP BY `MONTH`

ORDER BY 1 ASC;
```

```sql
-- Partitioning by the month while getting a more clear view of the data

WITH rolling_total AS
(
SELECT SUBSTRING(`date`,1,7) AS `MONTH`, SUM(total_laid_off) AS total_off
FROM layoffs_staging2
WHERE SUBSTRING(`date`,1,7) IS NOT NULL
GROUP BY `MONTH`
ORDER BY 1 ASC
)
select `month`, total_Off
,sum(total_off)    OVER(ORDER BY `MONTH`) AS rolling_total
from rolling_total;

SELECT company, YEAR(`date`), SUM(total_laid_off)
FROM layoffs_staging2
GROUP BY company, `date`
ORDER BY 1 DESC;

-- Ranking the years with the most laid offs

SELECT company, YEAR(`date`), SUM(total_laid_off)
FROM layoffs_staging2
GROUP BY company, `date`
ORDER BY 1 DESC;

WITH Company_Year (company,years,total_laid_off) AS
(
SELECT company, YEAR(`date`), SUM(total_laid_off)
FROM layoffs_staging2

GROUP BY company, YEAR(`date`)

ORDER BY 1 DESC

), Company_year_rank AS

(SELECT *, DENSE_RANK() OVER ( PARTITION BY years ORDER BY total_laid_off DESC) AS Ranking

FROM Company_Year

WHERE years IS NOT NULL

)

SELECT *

FROM Company_year_rank

WHERE Ranking <=5;
```