Data Cleaning Project

Data Cleaning transforms raw data into useful data that can be used in visualizations & products. Clean data allows for meaningful insights.

Data Source: https://github.com/AlexTheAnalyst/MySQL-YouTube-Series/blob/main/layoffs.csv

1. Create the database: world_layoffs
2. Create the table: layoffs
    a. Use import wizard

Column Name	Data Type	Primary Key	Allow Nulls
company	nvarchar(50)	☐	☐
location	nvarchar(50)	☐	☐
industry	nvarchar(50)	☐	☑
total_laid_off	smallint	☐	☑
percentage_laid_off	float	☐	☑
date	date	☐	☐
stage	nvarchar(50)	☐	☐
country	nvarchar(50)	☐	☐
funds_raised_millions	smallint	☐	☑

Azure will automatically put the correct data type for the columns but for the purpose of learning we are going to switch them to their raw format

```
--first look at the data --
select *
from layoffs;
```

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
1	Atlassian	Sydney	Other	500	0.05	3/6/2023	Post-IPO	Australia	210
2	SiriusXM	New York City	Media	475	0.08	3/6/2023	Post-IPO	United States	525
3	Alerzo	Ibadan	Retail	400	0	3/6/2023	Series B	Nigeria	16
4	UpGrad	Mumbai	Education	120	0	3/6/2023	Unknown	India	631
5	Loft	Sao Paulo	Real Estate	340	0.15	3/3/2023	Unknown	Brazil	788
6	Embark Trucks	SF Bay Area	Transportation	230	0.7	3/3/2023	Post-IPO	United States	317
7	Lendi	Sydney	Real Estate	100	0	3/3/2023	Unknown	Australia	59
8	UserTesting	SF Bay Area	Marketing	63	0	3/3/2023	Acquired	United States	152
9	Airbnb	SF Bay Area	*NULL*	30	0	3/3/2023	Post-IPO	United States	6400

Data Cleaning Process

1. Remove Duplicates
2. Standardize the data
3. Null values or blank values
4. Remove columns/rows that aren’t necessary
    a. In industry this can be very risky so we will create another table that will have the revisions

```
-- create another table that will have revisions
select *
into layoffs_staging
from layoffs;

-- has all the data that the table layoffs did
select *
from layoffs_staging;
```

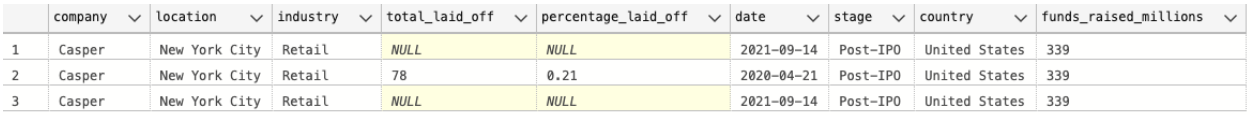
Removing Duplicates

```
-- if the row_num is 2 or above then there are duplicates

with duplicate_cte AS (
  select *,
    row_number() over(
        Partition By company, [location], industry, total_laid_off, percentage_laid_off
        Order by company
    ) as row_num
  from layoffs_staging
)

select *
from duplicate_cte
where row_num > 1;
```

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_num
1	Casper	New York City	Retail	*NULL*	*NULL*	2021-09-14	Post-IPO	United States	339	2
2	Cazoo	London	Transportation	750	0.15	2022-06-07	Post-IPO	United Kingdom	2000	2
3	Hibob	Tel Aviv	HR	70	0.3	2020-03-30	Series A	Israel	45	2
4	Wildlife Studios	Sao Paulo	Consumer	300	0.2	2022-11-28	Unknown	Brazil	260	2
5	Yahoo	SF Bay Area	Consumer	1600	0.2	2023-02-09	Acquired	United States	6	2

```
-- confirm duplicates --
select *
```

```
from layoffs_staging
where company = 'Casper'
```



	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
1	Casper	New York City	Retail	*NULL*	*NULL*	2021-09-14	Post-IPO	United States	339
2	Casper	New York City	Retail	78	0.21	2020-04-21	Post-IPO	United States	339
3	Casper	New York City	Retail	*NULL*	*NULL*	2021-09-14	Post-IPO	United States	339

- There are 2 identical rows within Casper; one needs to be retained & the other deleted

```
-- Remmove all the duplicates at once
with duplicate_cte AS (
  select *,
    row_number() over(
        Partition By company, [location], industry, total_laid_off, percentage_laid_off
        Order by company
    ) as row_num
  from layoffs_staging
)

Delete
from duplicate_cte
where row_num > 1;
```

- When you rerun the CTE & select everything where the row_num > 1 it'll return an empty table

Standardizing Data

Removing spaces before company names

```
update layoffs_staging
set company = trim(company);
```

```
-- view table --
select *
from layoffs_staging
```

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
1	Atlassian	Sydney	Other	500	0.05	2023-03-06	Post-IPO	Australia	210
2	SiriusXM	New York City	Media	475	0.08	2023-03-06	Post-IPO	United States	525
3	Alerzo	Ibadan	Retail	400	*NULL*	2023-03-06	Series B	Nigeria	16
4	UpGrad	Mumbai	Education	120	*NULL*	2023-03-06	Unknown	India	631
5	Loft	Sao Paulo	Real Estate	340	0.15	2023-03-03	Unknown	Brazil	788

```
-- Look into industry column and sort alphabetically
select distinct industry
from layoffs_staging
order by 1;
```

	industry
1	*NULL*
2	Aerospace
3	Construction
4	Consumer
5	Crypto
6	Crypto Currency
7	CryptoCurrency

- Some industries need to be combined (i.e. Crypto, Crypto Currency, CryptoCurrency)

```
-- industries that start with Crypto
select *
from layoffs_staging
where industry like 'Crypto%';
```

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
72	Coinsquare	Toronto	Crypto	30	0.24	2022-07-27	Unknown	Canada	98
73	Immutable	Sydney	Crypto	20	0.06	2022-07-26	Series C	Australia	279
74	Blockchain.com	London	Crypto	150	0.25	2022-07-21	Series D	United Kingdom	490
75	Gemini	New York City	CryptoCurr…	68	0.07	2022-07-18	Unknown	United States	423
76	Unstoppable Domai…	SF Bay Area	Crypto Cur…	42	0.25	2022-07-14	Series B	United States	7
77	OpenSea	New York City	Crypto	*NULL*	0.2	2022-07-14	Series C	United States	427
78	Ignite	SF Bay Area	Crypto	*NULL*	0.5	2022-07-11	Series A	United States	9

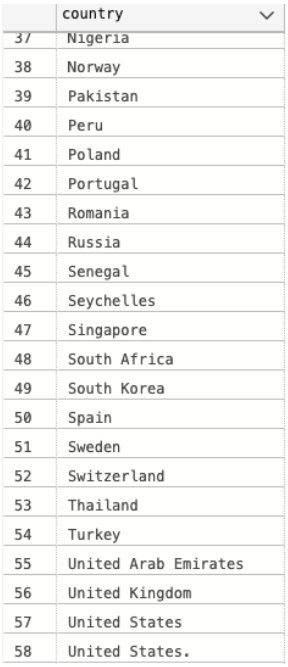
- Majority of them have the industry labeled as Crypto with the exception of a few

```
-- set the industry to Crypto for indutries that start with Crypto
update layoffs_staging
set industry = 'Crypto'
where industry like 'Crypto%';

-- check --
select *
from layoffs_staging
where industry like 'Crypto%';

select distinct industry
from layoffs_staging
order by 1;
```

	industry
1	*NULL*
2	Aerospace
3	Construction
4	Consumer
5	Crypto
6	Data
7	Education
8	Energy

- There’s now only one single line item for Crypto industries instead of 3 different variations

```
-- check the other columns one by one & scan for rows that should be combined
select distinct country
from layoffs_staging
order by 1;
```



	country
37	Nigeria
38	Norway
39	Pakistan
40	Peru
41	Poland
42	Portugal
43	Romania
44	Russia
45	Senegal
46	Seychelles
47	Singapore
48	South Africa
49	South Korea
50	Spain
51	Sweden
52	Switzerland
53	Thailand
54	Turkey
55	United Arab Emirates
56	United Kingdom
57	United States
58	United States.

- Rows with country labeled as ‘United States’ & ‘United States.’ need to be combined

```
-- Zoom into companies based in the United States

select *
from layoffs_staging
where country like 'United States%'
order by 1;
```

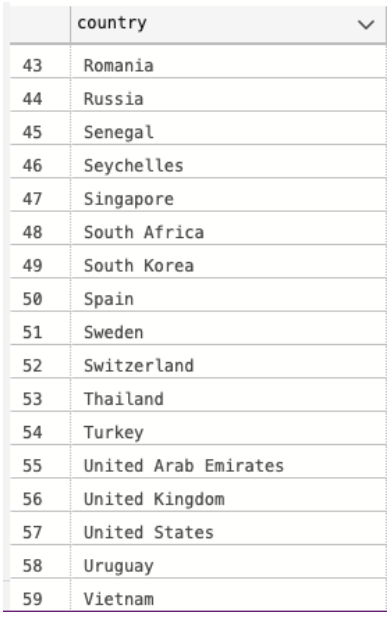
- When you filter the country column it lists 4 that are labeled as ‘United States.’

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
1	Indigo	Boston	Other	NULL	NULL	2023-03-03	Series F	United States.	1200
2	Landing	Birmingham	Real Estate	110	NULL	2022-10-06	Series C	United States.	347
3	Palantir	Denver	Data	75	0.02	2023-02-27	Post-IPO	United States.	3000
4	Twilio	SF Bay Area	Other	1500	0.17	2023-02-13	Post-IPO	United States.	614

```
-- CASE with RIGHT() and LEFT()
-- RIGHT(country, 1) checks if the last character is a period (.)
-- If it is, LEFT(country, LEN(country) - 1) removes the last character
-- Otherwise, the original value of country is returned

select distinct country,
  case
    when right(country, 1) = '.' then left(country, len(country)-1)
    else country
  end as trimmed_country
from layoffs_staging
order by 1;
```

	country	trimmed_country
46	Seychelles	Seychelles
47	Singapore	Singapore
48	South Africa	South Africa
49	South Korea	South Korea
50	Spain	Spain
51	Sweden	Sweden
52	Switzerland	Switzerland
53	Thailand	Thailand
54	Turkey	Turkey
55	United Arab E…	United Arab Emira…
56	United Kingdom	United Kingdom
57	United States	United States
58	United States.	United States

```
-- changing the country column
update layoffs_staging
set country = case
              when right(country, 1) = '.' then left(country, len(country)-1)
              else country
             end
where country like 'United States%';

-- verify
select distinct country
from layoffs_staging
order by 1;
```



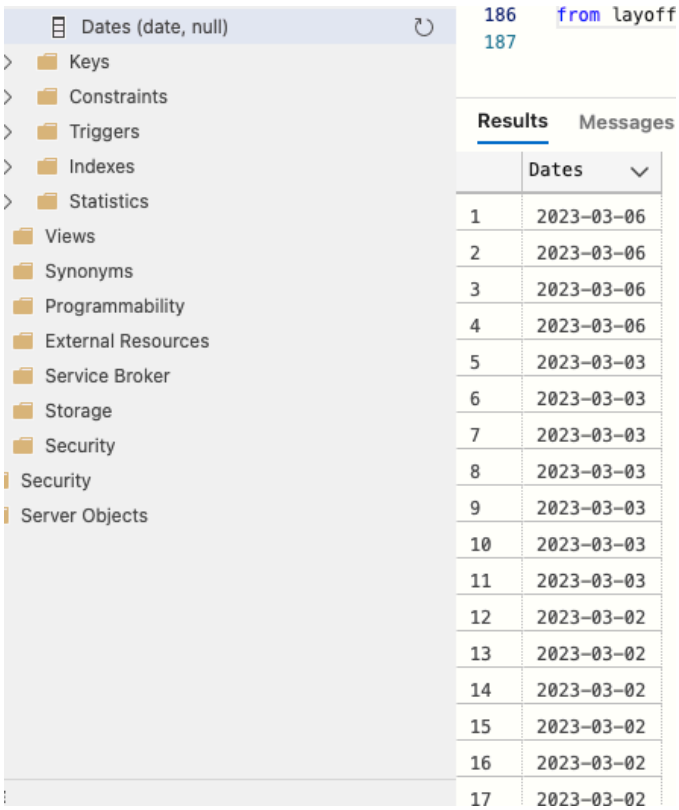
	country
43	Romania
44	Russia
45	Senegal
46	Seychelles
47	Singapore
48	South Africa
49	South Korea
50	Spain
51	Sweden
52	Switzerland
53	Thailand
54	Turkey
55	United Arab Emirates
56	United Kingdom
57	United States
58	Uruguay
59	Vietnam

Now, United states is just 1 row

Making the Date column a date data type. Now when the data was imported it was already in the Date data type format but if it wasn’t this would be the process.

```
-- change the format of the date column to Month/Day/Year
-- store it in column Dates
alter table layoffs_staging
add Dates varchar(50);

update layoffs_staging
set Dates = Convert(VARCHAR, [date], 101);

select *
from layoffs_staging;

-- makes Dates column a Date data type
alter table layoffs_staging
alter column Dates Date;
```

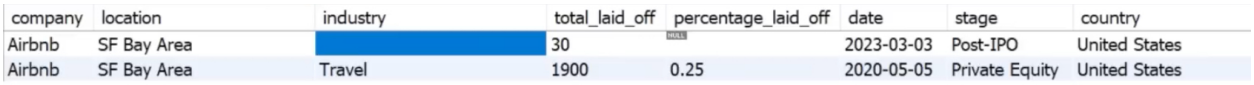
- 101 refers to the Month/Day/Year format
- The end result would be as below



Addressing NULL Values

Starting with industry column

```
-- look at rows where industry is either NULL or blank
select *
from layoffs_staging
where industry is NULL or industry = ' ';

-- look at each company individually to see if the missing data can be populated
-- Airbnb
select *
from layoffs_staging
where company = 'Airbnb';
```



company	location	industry	total_laid_off	percentage_laid_off	date	stage	country
Airbnb	SF Bay Area		30	NULL	2023-03-03	Post-IPO	United States
Airbnb	SF Bay Area	Travel	1900	0.25	2020-05-05	Private Equity	United States

- The goal is to get similar companies to have the same industry. Here we want to populate the industry for Airbnb to say Travel.

```
-- update the emptry strings in the industry column to be null only
update layoffs_staging
set industry = NULL
where industry = ' ';

-- looking at the industry column of the self join

select t1.industry, t2.industry
from layoffs_staging t1
join layoffs_staging t2
 on t1.company = t2.company
```

```
where t1.industry is NULL
and t2.industry is not null;
```

industry	industry
NULL	Travel
NULL	Transportation
NULL	Transportation
NULL	Consumer

- The purpose of the self-join is to get the empty industry column (t1) to match the populated industry column (t2)

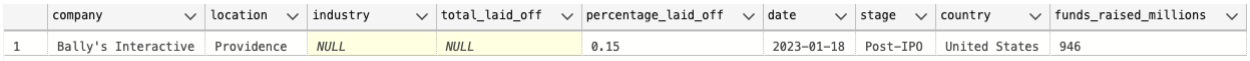
```
-- update the industry information that is null with the data that is populated for tl
update t1
set t1.industry = t2.industry
from layoffs_staging t1
join layoffs_staging t2
  on t1.company = t2.company
where t1.industry is NULL
and t2.industry is not null;

-- check Airbnb
select *
from layoffs_staging
where company = 'Airbnb';
```

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
1	Airbnb	SF Bay Area	Travel	30	*NULL*	2023-03-03	Post-IPO	United States	6400
2	Airbnb	SF Bay Area	Travel	1900	0.25	2020-05-05	Private Equity	United States	5400

- Now, Airbnb has Travel listed as the industry on both rows

```
-- check if null is still in industry column
select *
```

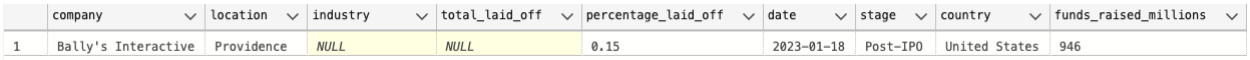
```
from layoffs_staging
where industry is NULL or industry = ' ';
```



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

- Bally’s Interactive is still showing Null

```
-- look into Bally's
select *
from layoffs_staging
where company like 'Bally%';
```



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

- Bally’s only had 1 layoff whereas other companies like Airbnb had multiple and it was possible to populate the missing data. Since Bally’s didn’t have another row that had populated data that could be copied over onto the missing fields the industry column remained NULL

Remove Rows

```
-- Remove rows where total_laid_off & percentaige_laid_off are null
-- Not meaningful b/c the point of the analysis is to look at companies who did ha
-- the following rows are saying these companies didn't have a layoff
select *
from layoffs_staging
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
1	Accolade	Seattle	Healthcare	*NULL*	*NULL*	2023-03-03	Post-IPO	United States	458
2	Indigo	Boston	Other	*NULL*	*NULL*	2023-03-03	Series F	United States	1200
3	Flipkart	Bengaluru	Retail	*NULL*	*NULL*	2023-03-02	Acquired	India	12900
4	Truckstop.com	Boise	Logistics	*NULL*	*NULL*	2023-03-02	Acquired	United States	*NULL*
5	Arch Oncology	St. Louis	Healthcare	*NULL*	*NULL*	2023-02-22	Series C	United States	155
6	Criteo	Paris	Marketing	*NULL*	*NULL*	2023-02-21	Post-IPO	France	61
7	Green Labs	Seoul	Food	*NULL*	*NULL*	2023-02-21	Series C	South Korea	214
8	PeerStreet	Los Angeles	Real Estate	*NULL*	*NULL*	2023-02-21	Series C	United States	121

- It’s questionable whether to delete the above rows because the total laid off & percentage laid off columns are showing these companies didn’t have a lay off as they’re NULL but there’s a date for the lay-off
- For the purposes of the analysis, we will assume that the companies didn’t have a lay off and can be removed from the dataset

```
delete
from layoffs_staging
where total_laid_off is NULL
and percentage_laid_off is NULL;

-- check
select *
from layoffs_staging
where total_laid_off is NULL
and percentage_laid_off is NULL;
```

	company	location	industry	total_laid_off	percentage_lai…	date	stage	country	funds_raised_m…