PROJECT

SQL DATA CLEANING

Project description:

This project focuses on the application of SQL-based data cleaning techniques to ensure the accuracy, consistency, and completeness of raw datasets. Data cleaning is an essential phase in the data preparation process, aimed at enhancing data quality by addressing issues such as missing values, duplicate records, incorrect data formats, and inconsistent entries.

Data cleaning process is crucial because dirty or flawed data can lead to incorrect insights, poor decision-making, and inefficiencies in downstream analytics.

This project utilized SQL queries to address common data quality issues like duplicate records, incorrect data types, missing or null values, and inconsistent formatting.

Approach:

- Remove duplicate entries.
- Standardize values in categorical fields.
- Correct inconsistent data formats (date or numerical fields).
- Identify and handle missing or null values.
- Remove columns and rows that are not necessary.
- Ensure data integrity for future analysis and reporting.

Tech-stack used:

I am using MySQL Workbench 8.0.38-winx64 CE and it is ideal for this project as it provides a user-friendly interface for running SQL queries, to clean and transform data.

1. Importing the data:

The data set(https://github.com/AlexTheAnalyst/MySQL-YouTube-Series/blob/main/layoffs.csv) has to be imported.

Create a database called world_layoffs

Import the tables into this database.

```
create database world_layoffs;
use world_layoffs;
```

Create a copy of this table to work with.

```
create table layoffs_staging
like layoffs;

insert into layoffs_staging
select distinct * from layoffs;
```

2.Remove the duplicates:

Removing duplicates is a key aspect of data cleaning, and various methods can be applied.

It can be done using temporary tables or using distinct.

Firstly, **identify the duplicates:**

```
with CTE as(
select *,
row_number() over(
partition by company, location, industry, total_laid_off, percentage_laid_off, `date`, stage, country, fu
as row_num
from layoffs )

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

Output:

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_num
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
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

The output table shows the rows that have duplicates in them.

So,

Using temporary table:

Using distinct we copy only the unique values in the new table

```
create table layoffs_staging
like layoffs;

insert into layoffs_staging
select distinct * from layoffs;
```

Now,

company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_

No duplicates present in the table.

3. Standardizing the data:

```
select* from layoffs_staging;

-- if we have space we need to trim that

select  company, trim(company)
from layoffs_staging;
```

Output, showing spaces present

company	trim(company)
E Inc.	E Inc.
Included Health	Included Health
#Paid	#Paid
&Open	&Open
100 Thieves	100 Thieves
100 Thieves	100 Thieves
10X Genomics	10X Genomics
1stdibs	1stdibs

```
-- now we need ot update the table

update layoffs_staging
set company = trim(company);

select  company, trim(company)
from layoffs_staging;
```

Output,

company	trim(company)
Included Health	Included Health
Incredible He...	Incredible He...
InDebted	InDebted
Indigo	Indigo
Industrious	Industrious
Infarm	Infarm
Infarm	Infarm

4.Looking for errors:

A. The industry column in the table contains some errors that need to be fixed.

```
select industry
from layoffs_staging;
```

Output

industry
NULL
Aerospace
Construction
Consumer
Crypto
Crypto Currency
CryptoCurrency
Data
Education
Energy
Fin-Tech

The industry column has rows for crypto and cryptocurrency assuming which are the same so we need to make it the same name.

```
update layoffs_staging
set industry = 'Crypto'
where industry like 'crypto%';

 select distinct industry
 from layoffs_staging;
```

0utput:

industry
NULL
Aerospace
Construction
Consumer
Crypto
Data
Education
Energy

B. The location column of the table has some errors.

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

Output:

country
Sweden
Switzerland
Thailand
Turkey
United Arab Emirates
United Kingdom
United States
United States.
Uruguay

The above table shows that United states are in two rows one with error so we will remove that to make it one row,

Query:

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

Output:

country	trim(trailing '.' from country)
Sweden	Sweden
Switzerland	Switzerland
Thailand	Thailand
Turkey	Turkey
United Arab Emirates	United Arab Emirates
United Kingdom	United Kingdom
United States	United States
United States.	United States
Uruguay	Uruguay
Vietnam	Vietnam

The above table shows that the error has been removed so we need to update this change.

```
update layoffs_staging
set country = trim(trailing  '.' from country)
where country like 'united states%';
```

5. Correct inconsistent data formats:

date
3/6/2023
3/6/2023
3/6/2023
3/6/2023
3/3/2023
3/3/2023

```
-- now as date is in text format we need to change it into datetime format  so  str_to_date can be used

select date ,
str_to_date(`date`, '%m/%d/%Y') AS Date
from layoffs_staging;

update layoffs_staging
set date = str_to_date(`date`, '%m/%d/%Y');

-- but we also need to alter it in the table to as it is presnt as text there so we need to use alter modify funct

alter table layoffs_staging
modify `date` date;
```

Output:

	date
▶	2023-03-06
	2023-03-06
	2023-03-06
	2023-03-06
	2023-03-03
	2023-03-03
	2023-03-03

6. Identify and handle missing or null values:

Identify nulls and blanks

```
select *
from layoffs_staging
where industry is null
or industry = '';
```

Output:

company	location	industry	total_laid_off	percentage_laid_off	date	stage
Airbnb	SF Bay Area		30	NULL	3/3/2023	Post-IPO
Bally's Interactive	Providence	NULL	NULL	0.15	1/18/2023	Post-IPO
Juul	SF Bay Area		400	0.3	11/10/2022	Unknown
Carvana	Phoenix		2500	0.12	5/10/2022	Post-IPO

```
-- so lets look one by one so we look for airbnd and look
-- if we can populate it rather than deleting the complte row

select *
from layoffs_staging
where company = 'airbnb';
```

Output:

company	location	industry	total_laid_off
Airbnb	SF Bay Area		30
Airbnb	SF Bay Area	Travel	1900

```
-- so we can see airbnb is a travel company and we can fill it with travel

-- lets self join the table

select *
from layoffs_staging t1
join layoffs_staging t2
on t1.company = t2.company
where (t1.industry is null or t1.industry = '') and
t2.industry is not null;
```

```sql
-- we need to change blanks into nulls first then update

update layoffs_staging
set industry = null
where industry = '';

update layoffs_staging t1
join layoffs_staging t2
on t1.company = t2.company
set t1.industry = t2.industry
where t1.industry is null and
t2.industry is not null;



-- now lets see if other industry is null

select *
from layoffs_staging
where industry is null
or industry = '';

-- so as ballys is still null we look up for it

select*
from layoffs_staging
where company like 'bally%';
```

So, there is only one ballys so we cannot populate it.

Also the percentage funds cannot be populated.

```sql
select *
from layoffs_staging
where total_laid_off is null
and percentage_laid_off is null;

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

We do not have information regarding these columns so we can not populate them so we delete these values.

The dataset has been cleaned and can be used for further analysis !

Result:

In this data cleaning project, I used SQL to identify and resolve inconsistencies within the dataset. By applying various SQL functions and techniques such as filtering null values, correcting data types, removing duplicates, and standardizing formats, I ensured that the data was structured, accurate, and ready for further analysis. This process highlighted the critical role of data cleaning in improving data quality, which ultimately enhances the reliability of the insights drawn from the data. Overall, this project solidified my understanding of SQL's powerful capabilities in preparing data for advanced analytics.