Two reasons to learn data cleaning with sql while we can also do it in python.

* In some scenarios we must do it.
* SQL is good with large datasets, and it is faster

Data Analysis process

* Gathering Data
* Assessing Data
* Cleaning Data

Here we don’t need to gather

We need to assess and clean

USE session\_35;

SELECT \* FROM session\_35.movies;

SELECT name

FROM movies

WHERE name LIKE '\_\_\_\_\_';

-- 5 spaces name search

SELECT name

FROM movies

WHERE name LIKE 'A\_\_\_\_';

SELECT name

FROM movies

WHERE name LIKE '%man';

SELECT name, UPPER(name), LOWER(name)

FROM movies;

SELECT CONCAT(name, ' - ', director, ' - ', star)

FROM movies;

-- instead of doing this use concat\_ws

SELECT CONCAT(name, ' ', 10)

FROM movies;

-- SQL follows IMPLICIT type conversion

SELECT CONCAT\_WS(' - ', name, director, star)

FROM movies;

-- substr() this will give you the substring of the string

-- In SQL string indexing starts from 1

SELECT name, SUBSTR(name, 1, 5)

FROM movies;

-- starts from 1 till end

SELECT name, SUBSTR(name, 1)

FROM movies;

-- starts from 5 till end

SELECT name, SUBSTR(name, 5)

FROM movies;

-- we can also give -ve

-- this will give last letter

SELECT name, SUBSTR(name, -1)

FROM movies;

-- If you want last 5 letters

-- this will start from last 5th letter

SELECT name, SUBSTR(name, -5)

FROM movies;

-- If you want to start from last 4th character and get 2 letters starting from there then

SELECT name, SUBSTR(name, -4, 2)

FROM movies;

-- replace

-- This is also to show how we can do operations on individual strings and constants, we don't have to always use tables

SELECT REPLACE("Hello World", "World", "India");

SELECT 5 + 6;

SELECT \* FROM (SELECT name, REPLACE(name, 'man', 'woman')

FROM movies) t

WHERE name LIKE '%man%';

-- Reverse

SELECT REVERSE("Hello");

SELECT name FROM movies

WHERE name = REVERSE(name);

-- char\_length and length(including spaces)

-- char\_length only counts the characters

-- while length counts the length of a string in bytes

-- Eg Café

-- Char\_length for this is 4 while length is 5, because it takes 2 bytes to store that é

SELECT name, LENGTH(name), CHAR\_LENGTH(name)

FROM movies

WHERE LENGTH(name) != CHAR\_LENGTH(name) ;

-- insert(str, pos, len newstr)

SELECT INSERT("Hello World", 7, 0, "India");

-- Output Hello IndiaWorld

-- if in place of 0 we give any positive value then it will replace the World

-- It will be like from 7th position rewrite 5 characters

SELECT INSERT("Hello World", 7, 5, "India");

-- Output - Hello India

-- left and write

SELECT name, LEFT(name, 3), RIGHT(name, 3) FROM movies;

-- Output -- The Shining The ing

-- repeat, 3 times

SELECT REPEAT(name, 3)

FROM movies;

-- trim, just like strip in python

SELECT TRIM(" sajjad ");

-- from both sides

SELECT TRIM(BOTH "." FROM "............sajjad...............");

SELECT TRIM(LEADING "." FROM "............sajjad..............");

SELECT TRIM(TRAILING "." FROM "............sajjad..............");

# This will not remove the middile one

SELECT TRIM(BOTH "." FROM "............saj....jad..............");

SELECT LTRIM(" sajjad "), LENGTH(LTRIM(" sajjad "));

SELECT RTRIM(" sajjad "), LENGTH(RTRIM(" sajjad "));

-- substring\_index(Split)

SELECT SUBSTRING\_INDEX("www.google.com", ".", 1);

-- output - www

SELECT SUBSTRING\_INDEX("www.google.com", ".", 2);

-- output - www.google

SELECT SUBSTRING\_INDEX("www.facebook.com", ".", -2);

-- output - facebook.com

-- strcmp

SELECT STRCMP("Delhi", "Mumbai");

-- o/p -> -1

SELECT STRCMP("Mumbai", "Delhi");

-- o/p -> 1

SELECT STRCMP("Delhi", "DELHI");

-- o/p --> 0

-- locate(substr, str, searching\_start\_from\_index)

SELECT LOCATE("w", "hello world");

-- o/p 7

SELECT LOCATE("l", "hello world", 5);

-- o/p 10

-- lpad and rpad

SELECT LPAD('9999999999', 13, '+91');

-- o/p +919999999999

SELECT LPAD('9999999999', 16, '+91');

-- o/p +91+919999999999

SELECT RPAD('sajjad', 16, '@gmail.com');

-- o/p sajjad@gmail.com

SELECT RPAD('sajjad', 19, '@gmail.com');

-- [sajjad@gmail.com@gm](mailto:sajjad@gmail.com@gm)

**Data Cleaning**

1. Create backup

2. Check number of rows

3. Check memory consumption for reference

4. Drop non important cols

5. Drop null values

6. Drop duplicates

7. Clean RAM -> change col data type

8. Clean weight -> change col type

9. ROUND price col and change to integer

10. Change the OpSys col

11. Gpu

12. Cpu

13. Resolution

14. Memory

* Add an index column

ALTER TABLE your\_table\_name

ADD COLUMN index\_column INT;

SET @row\_number = -1; -- Initialize variable to -1 so the first row gets 0

UPDATE your\_table\_name

SET index\_column = (@row\_number := @row\_number + 1)

ORDER BY some\_column; -- Specify how you'd like to order the rows

ALTER TABLE your\_table\_name

MODIFY COLUMN index\_column INT FIRST;

Actual implementation

USE laptop\_data;

SELECT \* FROM laptop\_data.laptop;

-- 1. Create backup

-- create empty table

CREATE TABLE laptops\_backup LIKE laptop;

INSERT INTO laptops\_backup

SELECT \* FROM laptop;

SELECT \* FROM laptops\_backup;

-- 2. Number of rows - 1272 rows

-- 3. check memory consumption for reference

SELECT DATA\_LENGTH/1024 FROM information\_schema.TABLES

WHERE TABLE\_SCHEMA = 'laptop\_data'

AND TABLE\_NAME = 'laptop';

-- 256 kb

-- Updating after going till the end. it has now become 224 kb

SELECT \* FROM laptop;

-- 4. Drop Unnamed

ALTER TABLE laptop DROP COLUMN `Unnamed: 0`;

SELECT \* FROM laptop;

-- 5. dropping null values

DELETE FROM laptop

WHERE `index` IN

(SELECT \* FROM laptop

WHERE Company IS NULL AND TypeName IS NULL AND Inches IS NULL

AND ScreenResolution IS NULL AND Cpu IS NULL AND Ram IS NULL

AND Memory IS NULL AND Gpu IS NULL AND OpSys IS NULL

AND WEIGHT IS NULL AND Price IS NULL);

SELECT \* FROM laptop;

-- 6. Drop Duplicates (Don't Run)

SELECT \* FROM zomato.duplicates;

DELETE FROM zomato.duplicates

WHERE id NOT IN (SELECT MIN(id)

FROM zomato.duplicates

GROUP BY name, gender, age);

-- adding index column

ALTER TABLE laptop

ADD COLUMN `index` INT;

SET @row\_number = -1;

UPDATE laptop

SET `index` = (@row\_number := @row\_number + 1);

ALTER TABLE laptop

MODIFY COLUMN `index` INT FIRST;

SELECT \* FROM laptop;

-- 7. Clean RAM -> change col data type

-- if we use DISTINCT on categorical column we can identify 2 things

-- 1. Whether we have any missing values

-- 2. What are the differnt kinds of values that are present

SELECT DISTINCT Company FROM laptop;

SELECT DISTINCT Typename FROM laptop;

-- converting Inches from DOUBLE to DECIMAL

ALTER TABLE laptop MODIFY COLUMN Inches DECIMAL(10, 1);

SELECT DISTINCT Inches FROM laptop;

-- Converting Ram from str to INT

-- step 1 - remove GB

UPDATE laptop l1

JOIN (SELECT `index`, REPLACE(Ram, "GB", '') AS 'new\_ram' FROM laptop) l2

ON l1.`index` = l2.`index`

SET l1.Ram = l2.new\_ram;

-- convert it to INT

ALTER TABLE laptop MODIFY COLUMN Ram INTEGER;

-- 8. Converting Weight to INT and removing kg from it

-- removing kg

UPDATE laptop l1

JOIN (SELECT `index`, REPLACE(Weight, 'kg', '') AS 'new\_weight' FROM laptop) l2

ON l1.`index` = l2.`index`

SET l1.Weight = l2.new\_weight;

-- Now converting str to INT

ALTER TABLE laptop MODIFY COLUMN Weight FLOAT;

-- debugging the ? in one row

SELECT \* FROM laptop

WHERE Weight REGEXP '[^0-9.]';

UPDATE laptop

SET Weight = NULL

WHERE Weight REGEXP '[^0-9.]';

-- Rerunning

ALTER TABLE laptop MODIFY COLUMN Weight FLOAT;

-- 9. ROUNDing Price and then changing its datatype to INTEGER

UPDATE laptop l1

JOIN (SELECT `index`, ROUND(Price) AS 'new\_price' FROM laptop) l2

ON l1.`index` = l2.`index`

SET l1.Price = l2.new\_price;

ALTER TABLE laptop MODIFY COLUMN Price INTEGER;

-- 10. Combining categories of Operating systems

SELECT DISTINCT OpSys FROM laptop;

-- we are going to combine and sort them in these groups

-- mac

-- windows

-- linux

-- no os

-- Android chrome(others)

SELECT OpSys,

CASE

WHEN OpSys LIKE '%mac%' THEN 'macos'

WHEN OpSys LIKE 'windows%' THEN 'windows'

WHEN OpSys LIKE '%linux%' THEN 'linux'

WHEN OpSys = 'No OS' THEN 'N/A'

ELSE 'other'

END AS 'os\_brand'

FROM laptop;

-- Now converting

UPDATE laptop

SET OpSys = CASE

WHEN OpSys LIKE '%mac%' THEN 'macos'

WHEN OpSys LIKE 'windows%' THEN 'windows'

WHEN OpSys LIKE '%linux%' THEN 'linux'

WHEN OpSys = 'No OS' THEN 'N/A'

ELSE 'other'

END;

-- 11. For Gpu, we will make 2 new columns 'gpu\_brand' and 'gpu\_name'

ALTER TABLE laptop

ADD COLUMN gpu\_brand VARCHAR(255) AFTER Gpu,

ADD COLUMN gpu\_name VARCHAR(255) AFtER gpu\_brand;

SELECT \* FROM laptop;

SELECT Gpu, SUBSTRING\_INDEX(Gpu, ' ', 1) AS 'new\_gpu\_brand' FROM laptop;

UPDATE laptop l1

JOIN (SELECT `index`, SUBSTRING\_INDEX(Gpu, ' ', 1) AS 'new\_gpu\_brand' FROM laptop) l2

ON l2.index = l1.index

SET l1.gpu\_brand = l2.new\_gpu\_brand;

SELECT Gpu, REPLACE(Gpu, Gpu\_brand, ' ') AS 'new\_gpu\_names' FROM laptop;

UPDATE laptop l1

JOIN (SELECT `index`, REPLACE(Gpu, Gpu\_brand, ' ') AS 'new\_gpu\_name' FROM laptop) l2

ON l2.index = l1.index

SET l1.gpu\_name = l2.new\_gpu\_name;

-- dropping Gpu

ALTER TABLE laptop DROP COLUMN Gpu;

-- 12. For CPU, we are going to create 3 columns

ALTER TABLE laptop

ADD COLUMN cpu\_brand VARCHAR(255) AFTER Cpu,

ADD COLUMN cpu\_name VARCHAR(255) AFTER cpu\_brand,

ADD COLUMN cpu\_speed DECIMAL(10, 1) AFTER cpu\_name;

-- cpu\_brand

SELECT Cpu, SUBSTRING\_INDEX(Cpu, " ", 1) FROM laptop;

UPDATE laptop l1

JOIN(SELECT `index`, SUBSTRING\_INDEX(Cpu, " ", 1) AS 'new\_cpu\_brand' FROM laptop) l2

ON l1.`index` = l2.`index`

SET l1.cpu\_brand = l2.new\_cpu\_brand;

-- cpu\_speed

SELECT Cpu, CAST(REPLACE(SUBSTRING\_INDEX(Cpu, ' ', -1), 'GHz', '') AS DECIMAL(10, 2)) AS 'new\_cpu\_speed' FROM laptop;

UPDATE laptop l1

JOIN(SELECT `index`, CAST(REPLACE(SUBSTRING\_INDEX(Cpu, ' ', -1), 'GHz', '') AS DECIMAL(10, 2)) AS 'new\_cpu\_speed'

FROM laptop) l2

ON l1.`index` = l2.`index`

SET l1.cpu\_speed = l2.new\_cpu\_speed;

-- cpu\_name

SELECT Cpu,

REPLACE(REPLACE(Cpu, cpu\_brand, ''), SUBSTRING\_INDEX(REPLACE(Cpu, cpu\_brand, ''), ' ', -1), '')

FROM laptop;

UPDATE laptop l1

JOIN(SELECT `index`,

REPLACE(REPLACE(Cpu, cpu\_brand, ''), SUBSTRING\_INDEX(REPLACE(Cpu, cpu\_brand, ''), ' ', -1), '') AS 'new\_cpu\_name'

FROM laptop) l2

ON l1.`index` = l2.`index`

SET l1.cpu\_name = l2.new\_cpu\_name;

SELECT \* FROM laptop;

-- dropping CPU

ALTER TABLE laptop DROP COLUMN Cpu;

USE laptop\_data;

SELECT \* FROM laptop;

-- 13. Splitting Resolution column

SELECT ScreenResolution,

SUBSTRING\_INDEX(ScreenResolution, ' ', -1),

SUBSTRING\_INDEX(SUBSTRING\_INDEX(ScreenResolution, ' ', -1), 'x', 1),

SUBSTRING\_INDEX(SUBSTRING\_INDEX(ScreenResolution, ' ', -1), 'x', -1)

FROM laptop;

ALTER TABLE laptop

ADD COLUMN resolution\_width INTEGER AFTER ScreenResolution,

ADD COLUMN resolution\_height INTEGER AFTER resolution\_width;

UPDATE laptop

SET resolution\_width = SUBSTRING\_INDEX(SUBSTRING\_INDEX(ScreenResolution, ' ', -1), 'x', 1),

resolution\_height = SUBSTRING\_INDEX(SUBSTRING\_INDEX(ScreenResolution, ' ', -1), 'x', -1);

ALTER TABLE laptop

ADD COLUMN touchscreen INTEGER AFTER resolution\_height;

SELECT ScreenResolution, ScreenResolution LIKE '%Touch%'

FROM laptop;

UPDATE laptop

SET touchscreen = ScreenResolution LIKE '%Touch%';

ALTER TABLE laptop

DROP COLUMN ScreenResolution;

SELECT \* FROM laptop;

-- From 12 we have got cpu\_name, we will work on it and categories them

SELECT cpu\_name FROM laptop;

SELECT cpu\_name,

SUBSTRING\_INDEX(TRIM(cpu\_name), ' ', 2)

FROM laptop;

UPDATE laptop

SET cpu\_name = SUBSTRING\_INDEX(TRIM(cpu\_name), ' ', 2);

-- 14 Memory - ROM

-- Breaking into 3 columns

-- 1. type

-- 2. Primary storage

-- 3. Secondary storage

SELECT Memory FROM laptop;

ALTER TABLE laptop

ADD COLUMN memory\_type VARCHAR(255) AFTER Memory,

ADD COLUMN primary\_storage INTEGER AFTER memory\_type,

ADD COLUMN secondary\_storage INTEGER AFTER primary\_storage;

SELECT DISTINCT Memory From laptop;

SELECT Memory,

CASE

WHEN Memory LIKE '%SSD%' AND Memory LIKE '%HDD%' THEN 'Hybrid'

WHEN Memory LIKE '%SSD%' THEN 'SSD'

WHEN Memory LIKE '%HDD%' THEN 'HDD'

WHEN Memory LIKE '%Flash Storage%' THEN 'Flash storage'

WHEN Memory LIKE '%Hybrid%' THEN 'Hybrid'

WHEN Memory LIKE '%Flash Storage%' AND Memory LIKE '%HDD%' THEN 'Hybrid'

ELSE NULL

END AS 'memory\_type'

FROM laptop;

UPDATE laptop

SET memory\_type = CASE

WHEN Memory LIKE '%SSD%' AND Memory LIKE '%HDD%' THEN 'Hybrid'

WHEN Memory LIKE '%SSD%' THEN 'SSD'

WHEN Memory LIKE '%HDD%' THEN 'HDD'

WHEN Memory LIKE '%Flash Storage%' THEN 'Flash storage'

WHEN Memory LIKE '%Hybrid%' THEN 'Hybrid'

WHEN Memory LIKE '%Flash Storage%' AND Memory LIKE '%HDD%' THEN 'Hybrid'

ELSE NULL

END;

-- splitting on +

SELECT Memory,

REGEXP\_SUBSTR(SUBSTRING\_INDEX(Memory, '+', 1), '[-0-9]+'),

REGEXP\_SUBSTR(TRIM(CASE WHEN Memory LIKE '%+%' THEN SUBSTRING\_INDEX(Memory, '+', -1) ELSE 0 END), '[0-9]+')

FROM laptop;

UPDATE laptop

SET primary\_storage = REGEXP\_SUBSTR(SUBSTRING\_INDEX(Memory, '+', 1), '[-0-9]+'),

secondary\_storage = REGEXP\_SUBSTR(TRIM(CASE WHEN Memory LIKE '%+%' THEN SUBSTRING\_INDEX(Memory, '+', -1) ELSE 0 END), '[0-9]+');

SELECT \* FROM laptop;

-- in secondary\_storage and primary\_storage we need to convert 1 and 2 to GB by multiplying by 1024

SELECT primary\_storage,

CASE WHEN primary\_storage <= 2 THEN primary\_storage\*1024 ELSE primary\_storage END,

secondary\_storage,

CASE WHEN secondary\_storage <= 2 THEN secondary\_storage\*1024 ELSE secondary\_storage END

FROM laptop;

UPDATE laptop

SET primary\_storage = CASE WHEN primary\_storage <= 2 THEN primary\_storage\*1024 ELSE primary\_storage END,

secondary\_storage = CASE WHEN secondary\_storage <= 2 THEN secondary\_storage\*1024 ELSE secondary\_storage END;

ALTER TABLE laptop

DROP COLUMN Memory;

-- dropping gpu\_name because they are too much variety

ALTER TABLE laptop

DROP COLUMN gpu\_name;