DATA CLEANING

# ~~-- Step 2.1. Irrelevant data - Delete Unused Columns~~

-- Delete Unused Columns

Select \*

From …….

ALTER TABLE …….

DROP COLUMN Manufacturer, Postal.Code

# ~~-- Step 2.2. Duplicate~~

-- Remove Duplicates

Select \*,

ROW\_NUMBER() OVER (

PARTITION BY Order.ID, Product.ID

) – 1 AS duplicate

From ……

Select \*

From ….

Where duplicate > 0

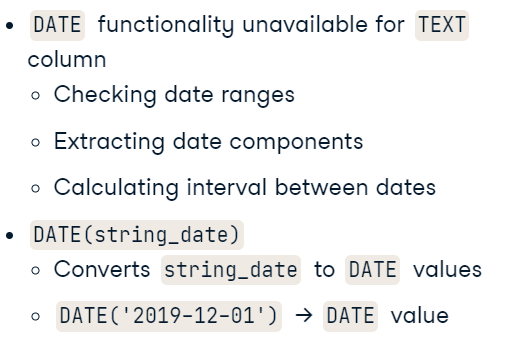
# -- Step 2.3. Type conversion (Number, Date, Unix Timestamp)

* ~~Ban đầu là text type cả => convert / cast các cột tương ứng~~
* Conversion with CASE WHEN

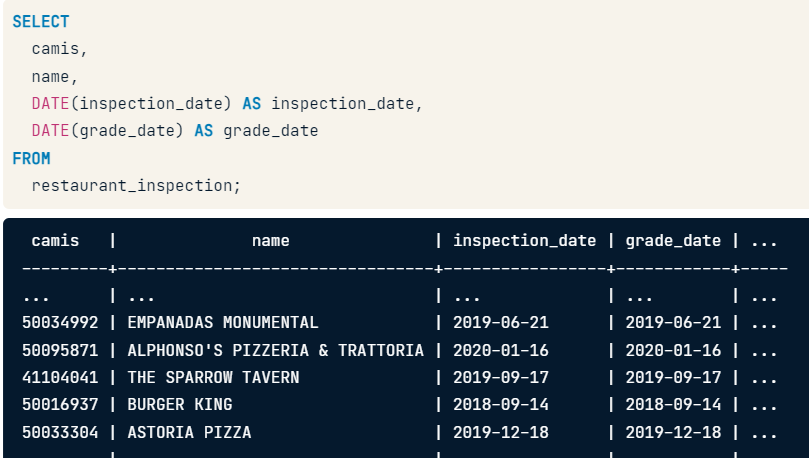
Profit?

* Date:

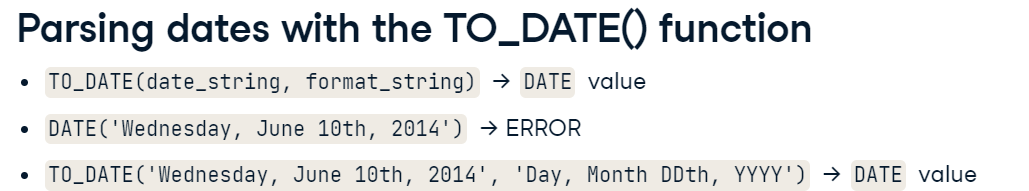
Những functions liên quan date



* date to string:
  + Chuyển đổi thành string => date
    - **Date (string\_date)**

****

* **TO\_DATE function => date value**

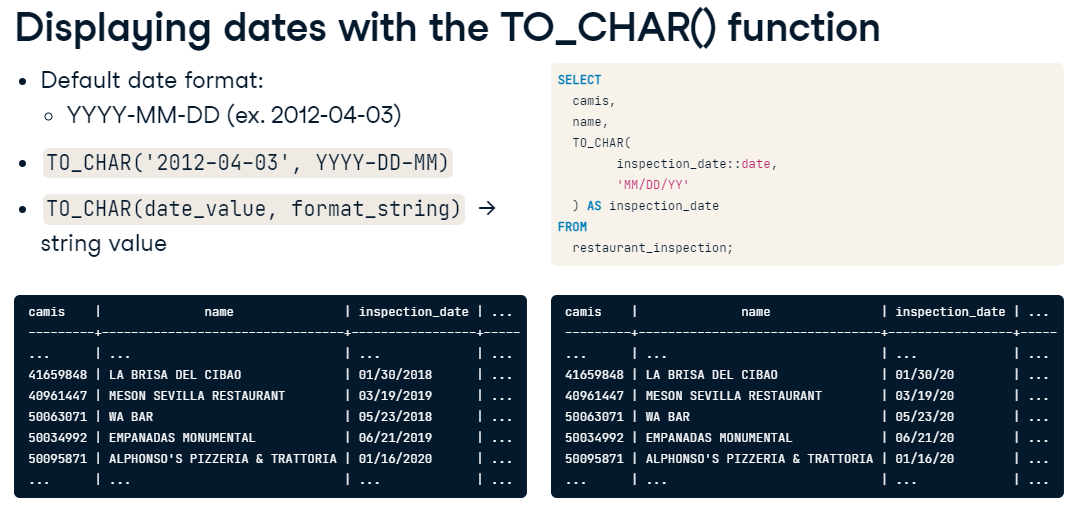


* + Đôi khi: có null

=> NULLIF



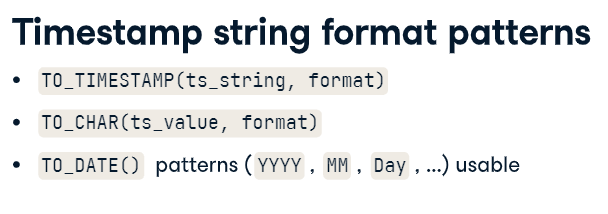
* + Displaying dates with the TO\_CHAR() function => string value

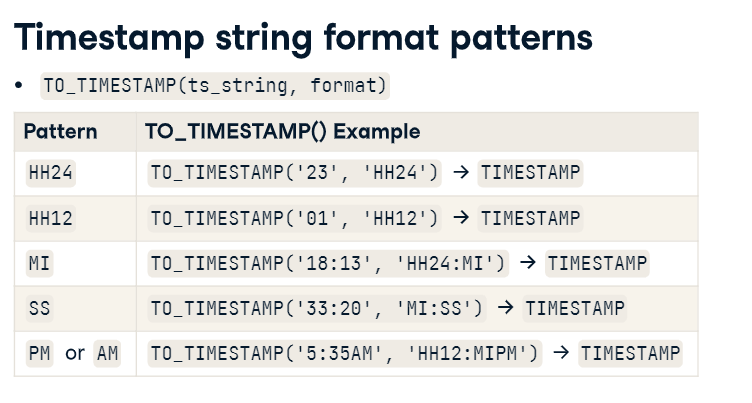


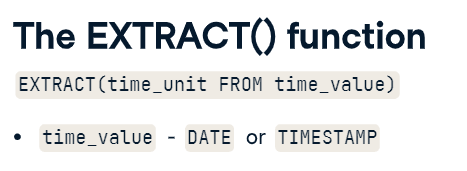
Mở rộng: TO\_DATE

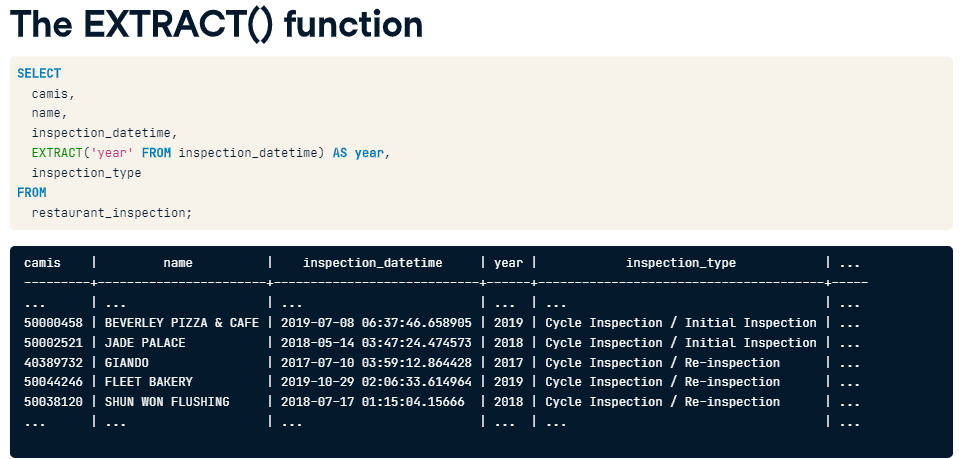
* Timestamp parsing and formatting

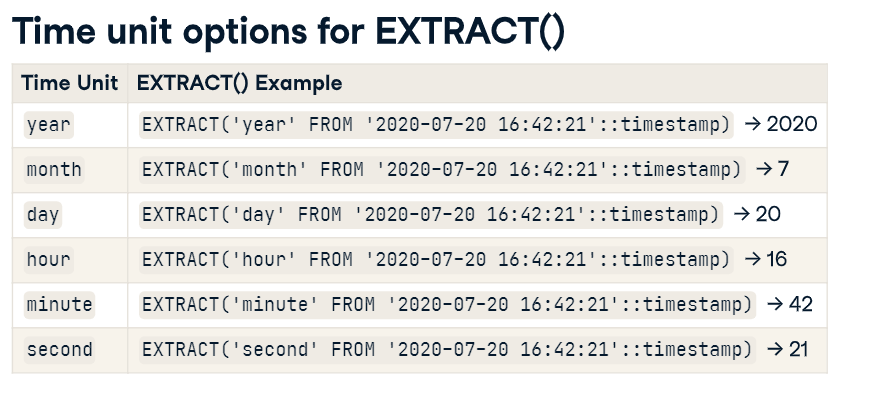












# -- Step 2.4. Break “Adresss” column into Country, Region, City

Select

**PARSENAME(REPLACE**(Adresss, ',', '.') , 1)

,PARSENAME(REPLACE(Adresss, ',', '.') , 2)

,PARSENAME(REPLACE(Adresss, ',', '.') , 3)

From ….

ALTER TABLE ….

Add Country Nvarchar(255);

Update NashvilleHousing

SET Country = PARSENAME(REPLACE(Adresss, ',', '.') , 1)

ALTER TABLE …..

Add Region Nvarchar(255);

Update NashvilleHousing

SET Region = PARSENAME(REPLACE(Adresss, ',', '.') , 2)

ALTER TABLE ….

Add City Nvarchar(255);

Update NashvilleHousing

SET City = PARSENAME(REPLACE(Adresss, ',', '.') , 3)

Select \*

From ….

# ~~-- Step 2.4. Syntax Error (remove white spaces, pad strings, fix types)~~ **~~=> group by to graph a bar plot is the best way~~**

~~Sửa tên Category~~

-- standardize category names

<https://campus.datacamp.com/courses/cleaning-data-in-postgresql-databases/data-cleaning-basics?ex=11>

SELECT DISTINCT category

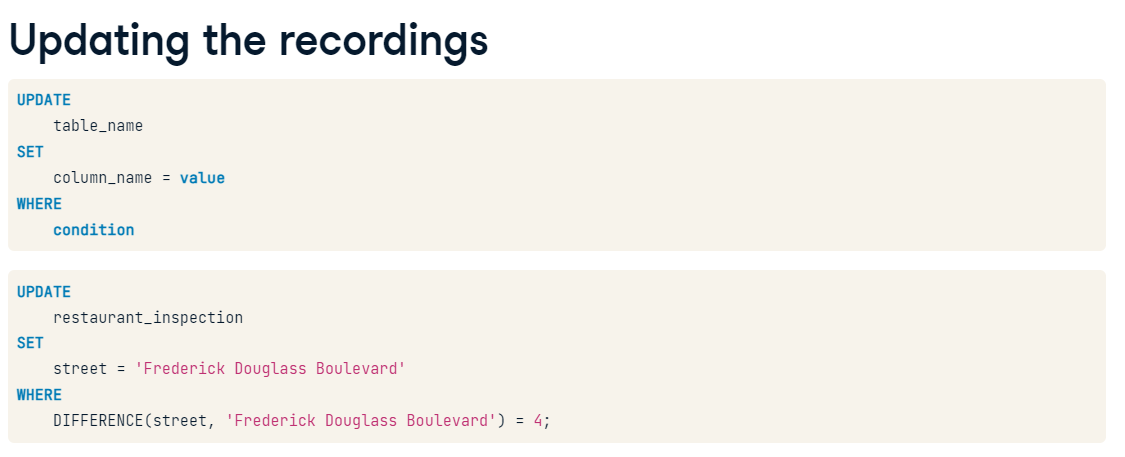
, COUNT (\*)

FROM …

GROUP BY

ORDER BY category;

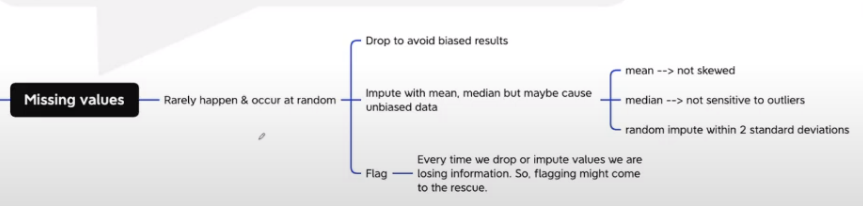
=> dùng update để update



Reference: <https://www.mssqltips.com/sqlservertip/7391/simple-bar-chart-sql-code/>

# -- Step 2.5. Missing values (xem kỹ)

Fill category vào dựa vào product name



Lý do:

* thực ra không có thật
* hệ thống bị lỗi (có những ngày null 80-90%)

Cách xử lý:

* B1: kiểm tra xem gtri NULL **xảy ra thường xuyên không**
  + Nếu xảy ra thường xuyên (ngày nào cũng null) => bình thường (vd hệ thống bình thường, mỗi ngày 1%-2% data null) => accept
  + Lâu lâu mới null. VD trong 1 tháng bỗng vài ngày bị null => cần chú ý lý do. Vd nếu vì lỗi kỹ thuật (máy móc trục trặc ghi nhận => báo bộ phận kỹ thuật sửa)

B2: check Null chiếm bao nhiêu phần trăm trong dữ liệu mình phân tích. Nếu rất nhỏ (1%-5%) & rải đều tất cả các khoảng thời gian => ít & random (không làm bias tập thông tin mình đang phân tích) => drop rows

B3: nếu thiếu nhiều hơn (vd 10%)

* thay thế với gt trung bình (mean), trung vị (median)
  + Mean is most useful when the original data is not skewed, while the median is more robust, not sensitive to outliers, and thus used when data is skewed.
* Random

B4: nếu thiếu quá nhiều => hỏi sếp để đủ rồi làm

Ngoài ra, nếu có căn cứ để chèn thêm thì chèn thêm

Mở rộng trong file tham khảo:

Missing values

Given the fact the missing values are unavoidable leaves us with the question of what to do when we encounter them. Ignoring the missing data is the same as digging holes in a boat; It will sink.

There are three, or perhaps more, ways to deal with them.

— One. Drop.

If the missing values in a column rarely happen and occur at random, then the easiest and most forward solution is to drop observations (rows) that have missing values.

If most of the column’s values are missing, and occur at random, then a typical decision is to drop the whole column.

This is particularly useful when doing statistical analysis, since filling in the missing values may yield unexpected or biased results.

— Two. Impute.

It means to calculate the missing value based on other observations. There are quite a lot of methods to do that.

— First one is using statistical values like mean, median. However, none of these guarantees unbiased data, especially if there are many missing values.

Mean is most useful when the original data is not skewed, while the median is more robust, not sensitive to outliers, and thus used when data is skewed.

In a normally distributed data, one can get all the values that are within 2 standard deviations from the mean. Next, fill in the missing values by generating random numbers between (mean — 2 \* std) & (mean + 2 \* std)

rand = np.random.randint(average\_age - 2\*std\_age, average\_age + 2\*std\_age, size = count\_nan\_age)

dataframe["age"][np.isnan(dataframe["age"])] = rand

— Second. Using a linear regression. Based on the existing data, one can calculate the best fit line between two variables, say, house price vs. size m².

It is worth mentioning that linear regression models are sensitive to outliers.

— Third. Hot-deck: Copying values from other similar records. This is only useful if you have enough available data. And, it can be applied to numerical and categorical data.

One can take the random approach where we fill in the missing value with a random value. Taking this approach one step further, one can first divide the dataset into two groups (strata), based on some characteristic, say gender, and then fill in the missing values for different genders separately, at random.

In sequential hot-deck imputation, the column containing missing values is sorted according to auxiliary variable(s) so that records that have similar auxiliaries occur sequentially. Next, each missing value is filled in with the value of the first following available record.

What is more interesting is that 𝑘 nearest neighbour imputation, which classifies similar records and put them together, can also be utilized. A missing value is then filled out by finding first the 𝑘 records closest to the record with missing values. Next, a value is chosen from (or computed out of) the 𝑘 nearest neighbours. In the case of computing, statistical methods like mean (as discussed before) can be used.

— Three. Flag.

Some argue that filling in the missing values leads to a loss in information, no matter what imputation method we used.

That’s because saying that the data is missing is informative in itself, and the algorithm should know about it. Otherwise, we’re just reinforcing the pattern already exist by other features.

This is particularly important when the missing data doesn’t happen at random. Take for example a conducted survey where most people from a specific race refuse to answer a certain question.

Missing numeric data can be filled in with say, 0, but has these zeros must be ignored when calculating any statistical value or plotting the distribution.

While categorical data can be filled in with say, “Missing”: A new category which tells that this piece of data is missing.

— Take into consideration …

Missing values are not the same as default values. For instance, zero can be interpreted as either missing or default, but not both.

Missing values are not “unknown”. A conducted research where some people didn’t remember whether they have been bullied or not at the school, should be treated and labelled as unknown and not missing.

Every time we drop or impute values we are losing information. So, flagging might come to the rescue.

-- Handing missing values (populate…)

<https://campus.datacamp.com/courses/cleaning-data-in-postgresql-databases/missing-duplicate-and-invalid-data?ex=5>

# -- Step 2.6. Outliers: thường bỏ

# -- Step 2.7. Normalization & scale data: khá đặc thù về machine learning

-- Step 3: Verifying: After cleaning, the results are inspected to verify correctness.

-- Step 4: Reporting: A report about the changes made and the quality of the currently stored data is recorded.

-- [NOT YET] Converting Data https://campus.datacamp.com/courses/cleaning-data-in-postgresql-databases/data-cleaning-basics?ex=11

-- https://github.com/AlexTheAnalyst/PortfolioProjects/blob/main/Data%20Cleaning%20Portfolio%20Project%20Queries.sql

- [NOT YET] Transforming Data https://campus.datacamp.com/courses/cleaning-data-in-postgresql-databases/data-cleaning-basics?ex=11

---------------------------------------------

-- DATA EXPLORATION

-- Aggregating Time Series Data https://campus.datacamp.com/courses/time-series-analysis-in-sql-server/aggregating-time-series-data?ex=2

-- Answering Time Series Questions with Window Functions ->

-- LỖI Break out a date into year, month & select distinct values & Aggregating Time Series Data & Using CTE, window function and pivot table to analyse promotion effectiveness

WITH promotion\_table AS (

SELECT DISTINCT YEAR ([order\_date]) AS order\_year

, MONTH([order\_date]) AS order\_month

, promo

, SUM (sales) OVER (PARTITION BY YEAR ([order\_date]), MONTH([order\_date]) ORDER BY YEAR ([order\_date]), MONTH([order\_date])) AS sum\_sales

FROM [projects].[dbo].[superstore]

)

SELECT order\_year

, order\_month

, [Bookcases], [Chairs], [Labels], [Storage], [Furnishings], [Art], [Phones], [Binders],[Appliances], [Paper], [Accessories], [Envelopes], [Fasteners], [Machines], [Copiers]

FROM

( SELECT order\_year, order\_month, promo, sum\_sales

FROM promotion\_table) AS new\_table

PIVOT

( SUM (sum\_sales))

FOR promo IN ([Bookcases], [Chairs], [Labels], [Storage], [Furnishings], [Art], [Phones], [Binders],[Appliances], [Paper], [Accessories], [Envelopes], [Fasteners], [Machines], [Copiers]) AS new\_table2