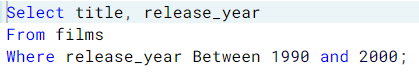
**Introduction to SQL**

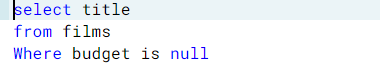
**˖**

‘Between and’ include 1990 and 2000

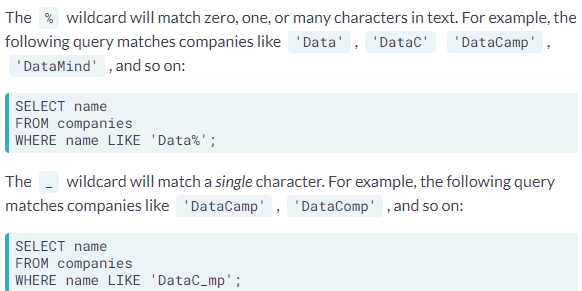
**˖** ‘Where in’ function:

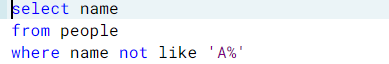
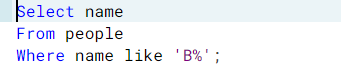


**˖**Null and is Null

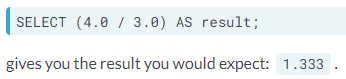
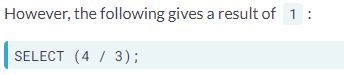
Null means the value is missing, 

**˖**Like and not like

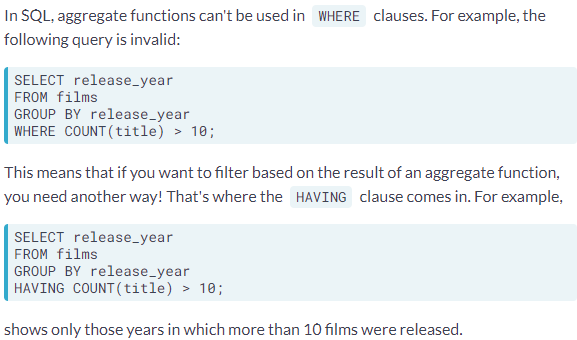




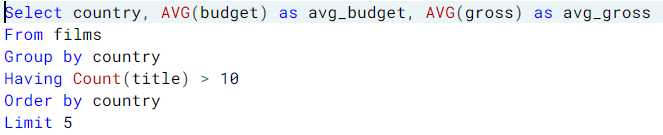
**˖**A note on arithmetic



**˖**Having function



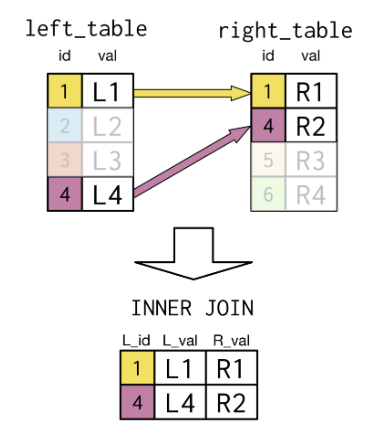
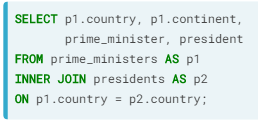
˖Limit function



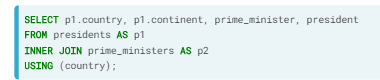
The top 5 value from the table

**Joining Data in SQL**

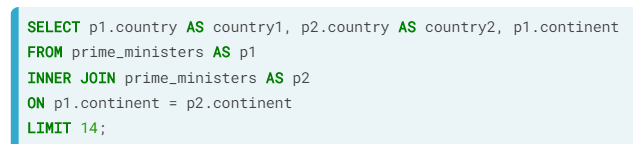
˖Intro to inner join and ‘on’

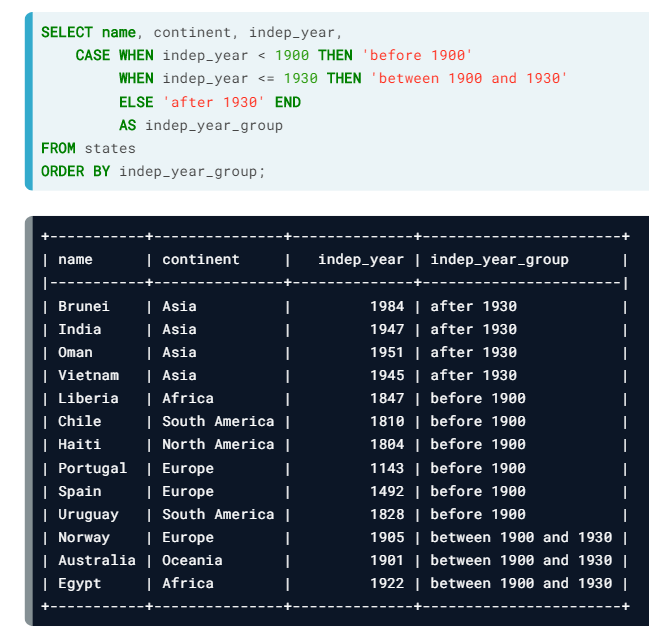
˖Inner join via using



˖Self inner join

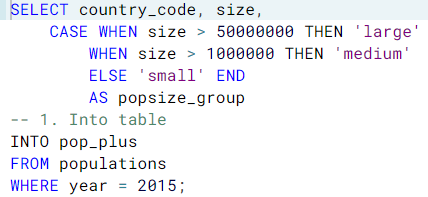


˖Case then function



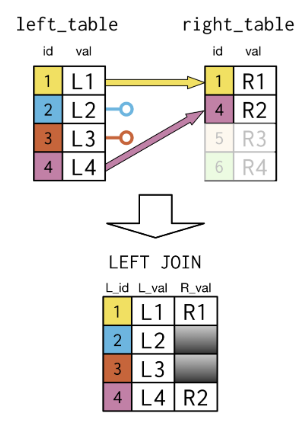
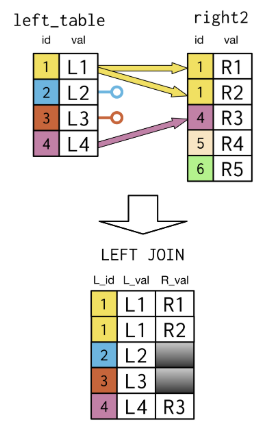
˖’Into’ command

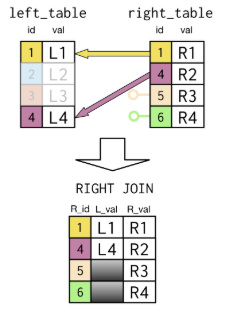
Into command to create a new table with a new name



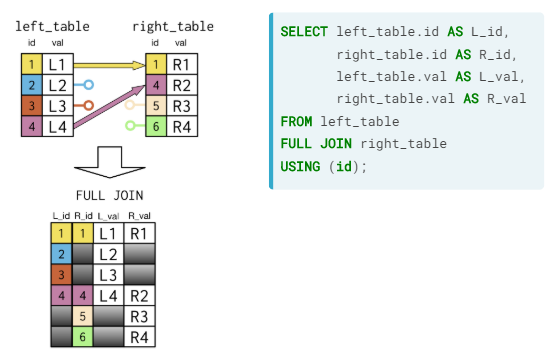
All the content in ‘select’ assembles as a new table called pop\_plus

˖Left join and right join

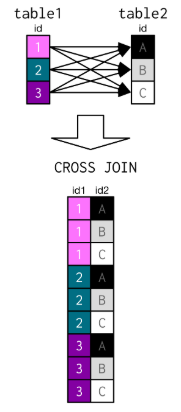
 



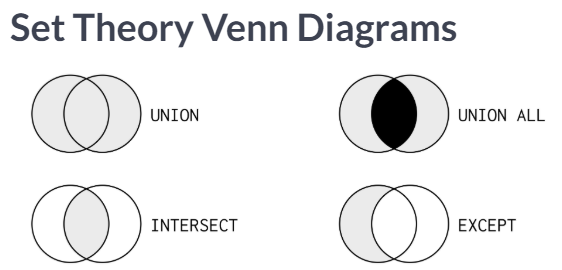
˖Full join

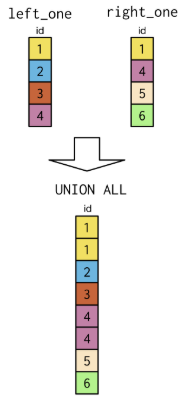
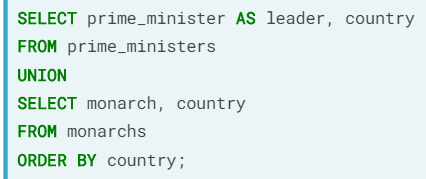


˖Cross join

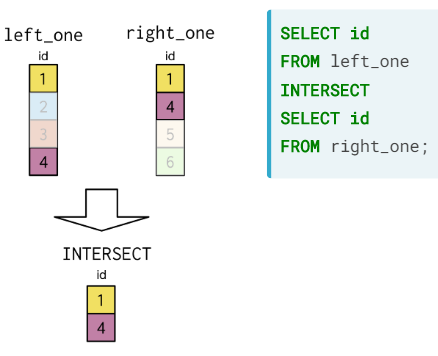


˖ Union

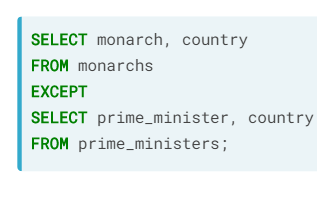
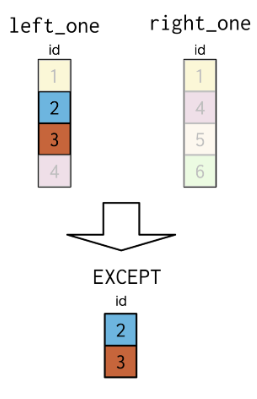


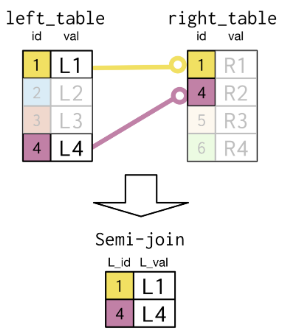
˖ Intersect



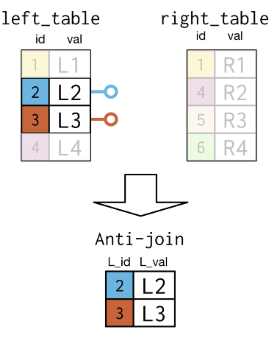
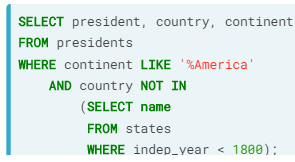
˖Except



˖Semi join(subquery)



˖Anti-join

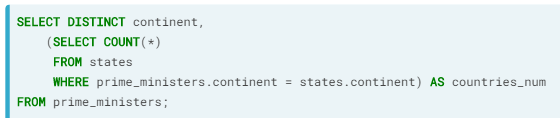
\

˖Subquery (nested queries)

Inside of ‘Where’ clause:

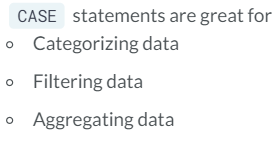


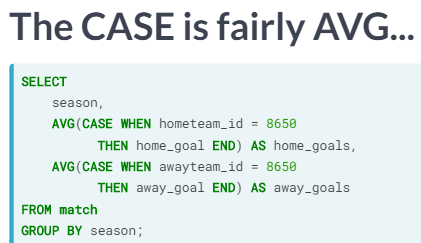
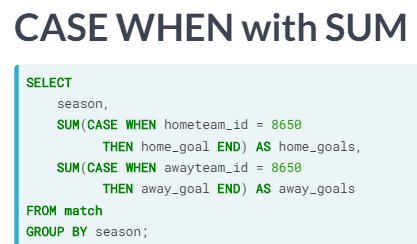
Inside of ‘Select’ clause:



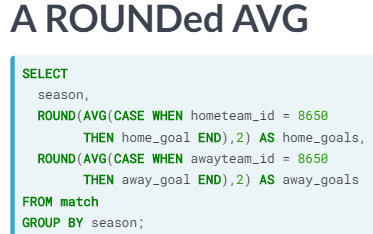
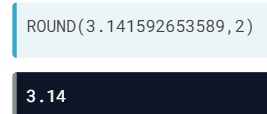
Intermediate SQL

˖Case statement





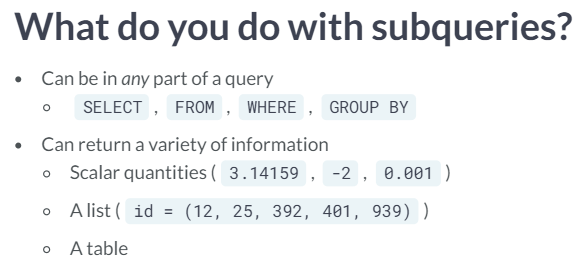
˖Round function

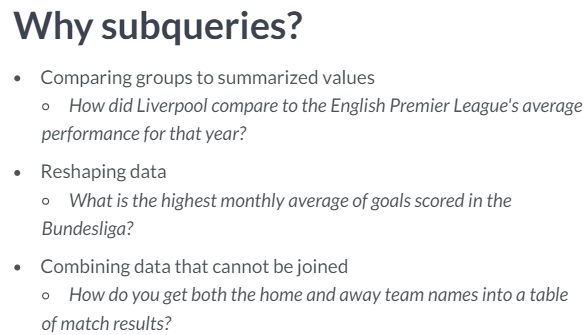




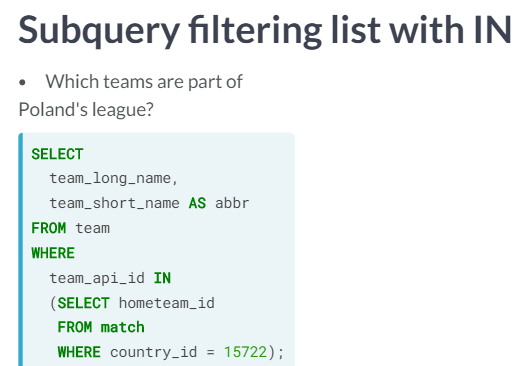
˖Subquery: A query nested in another query

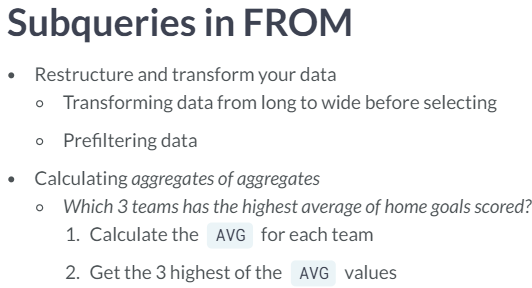
Why is it important? In order to retrieve some info, we have to perform some intermediary transformations before selecting, filtering and calculating data. Subquery is a common way to perform this transformation.

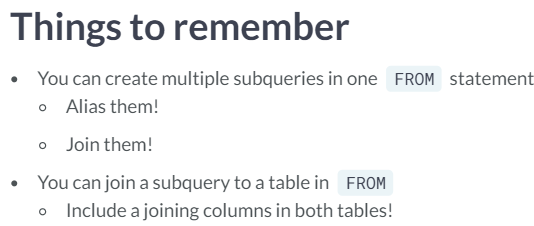
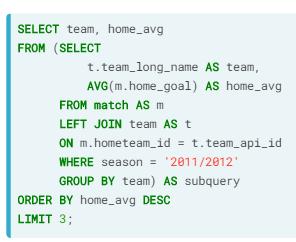


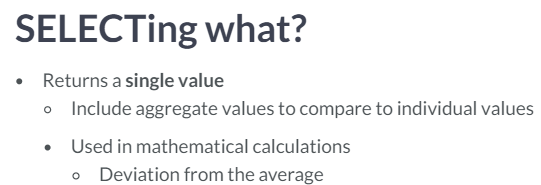








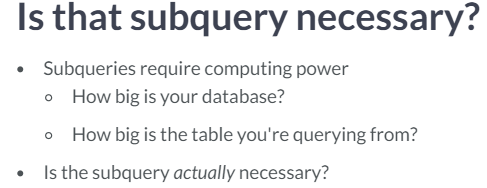




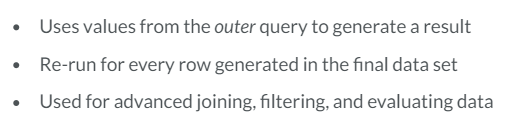


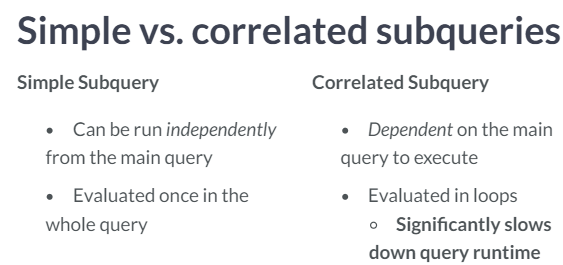


**Things to remember:**

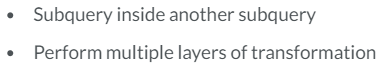
1. Can include many subqueries as you want in ‘Select, From, Where’, but it can be long and tedious.
2. Format queries. Line up ‘select, from, where, group by’
3. Annotate queries (‘/\* \*/’, or inline comments: ‘--')
4. Indent subqueries
5. 

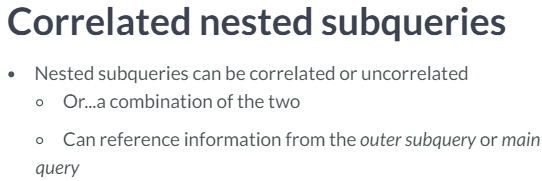
˖Correlated subquery: subquery is dependent by outer query for output





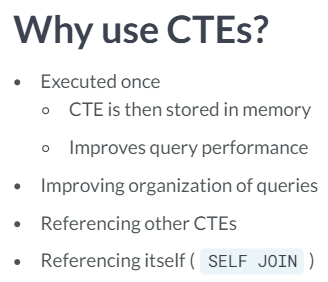
˖Nested subquery





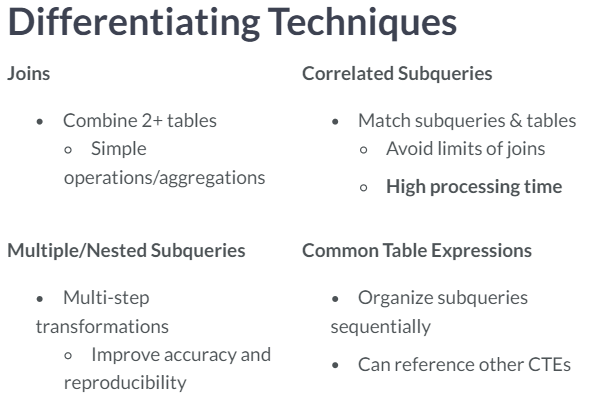
˖Common table expression







˖ Differentiating techniques



Which one to use?

Joins: 2+ tables

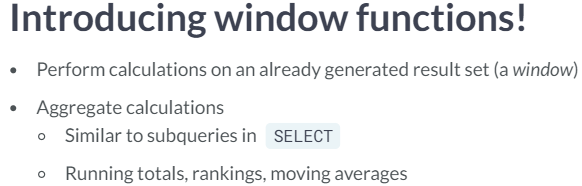
Correlated subqueries: Great for matching data from different columns in one or more tables

Nested subqueries: task that requires multiple steps to transform and prepare before generating the final query

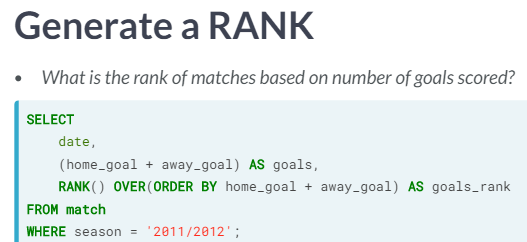
CTE: good for comparing a large number of info, you can extract data from each aspect one after another and combine them to a single query

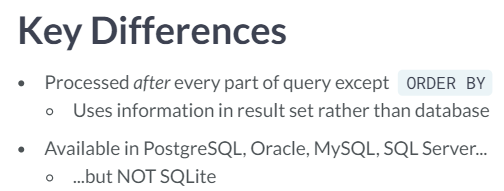
˖Window functions

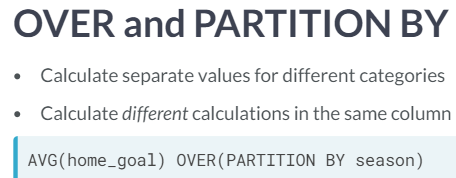
Group by function can only use on non-aggregate functions. To solve the limitation, window function is introduced.

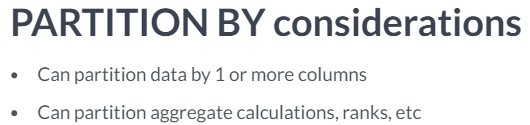








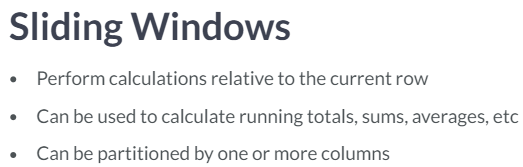


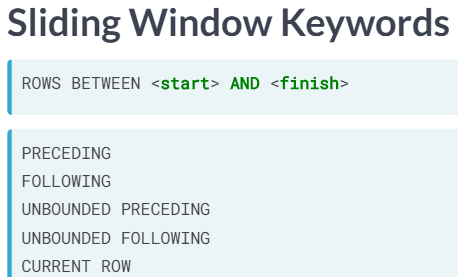


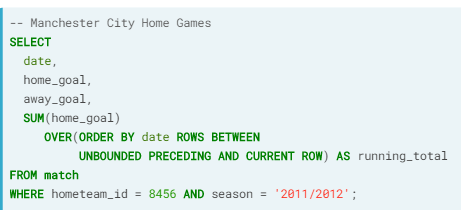
˖Exact function



˖Sliding windows



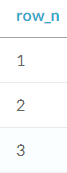
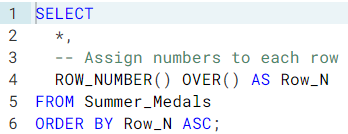




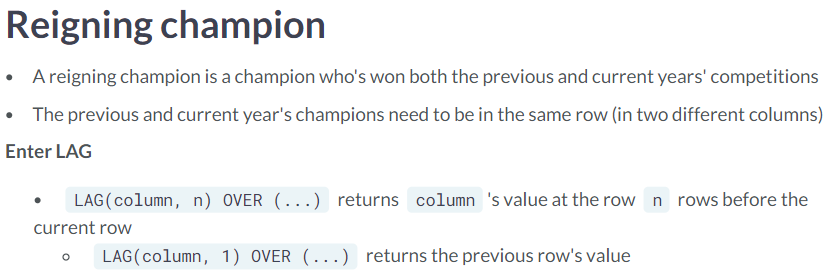


**PostgreSQL Summary Stats and Window Functions**

˖ Row\_number function: assign a number to each row

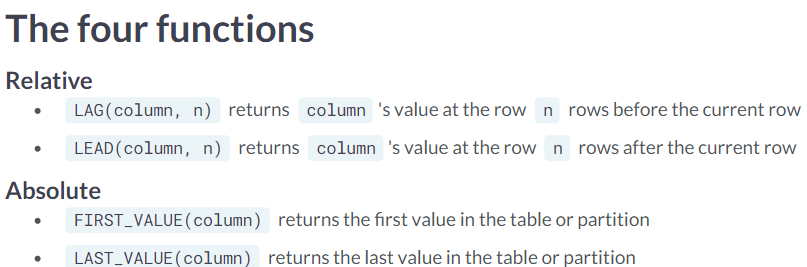


˖Lag function:



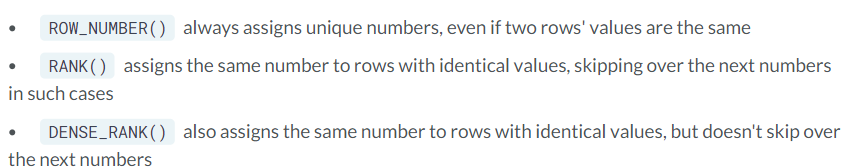


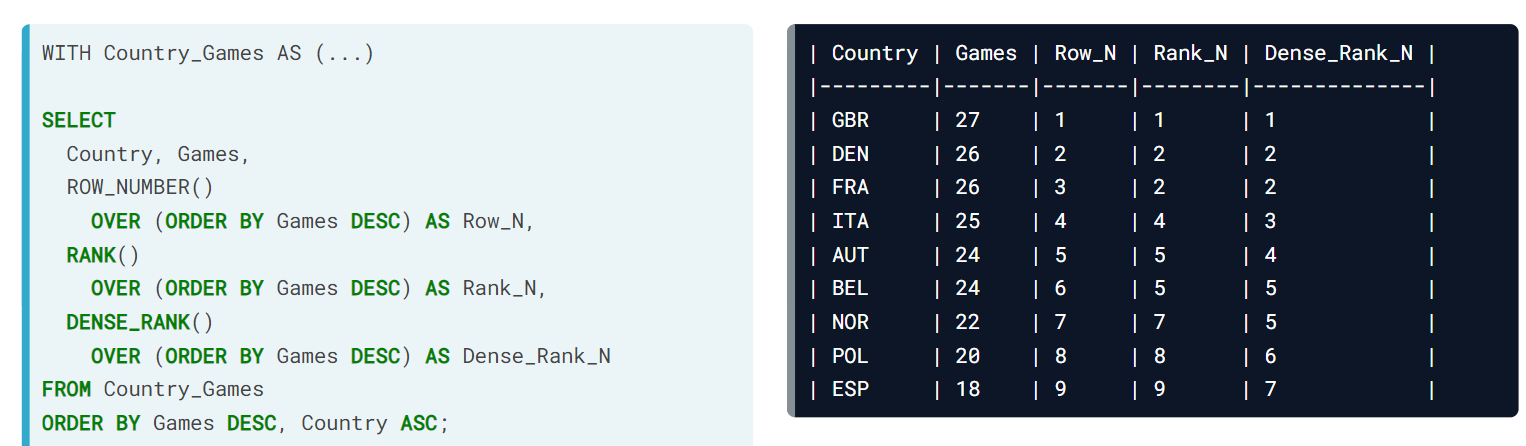
˖Fetch functions

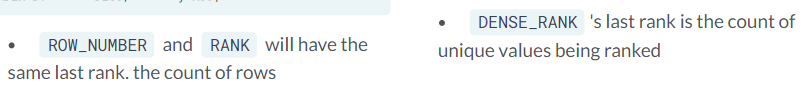




˖Ranking function







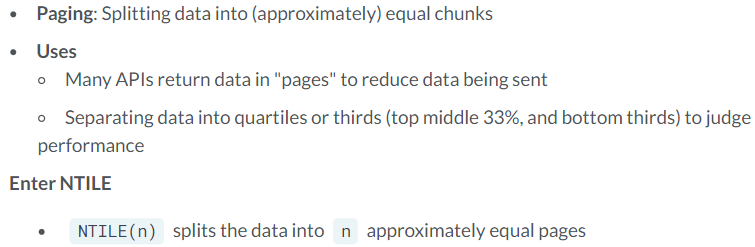


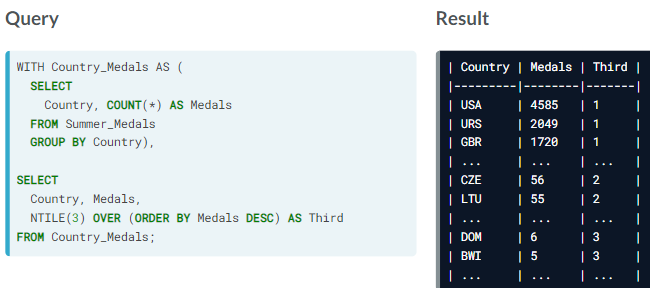


˖Paging

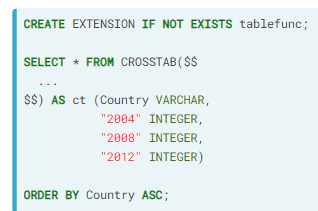
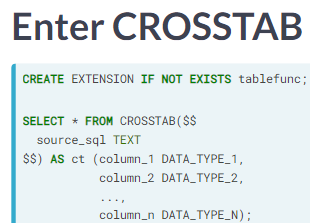
Application programming interface: API

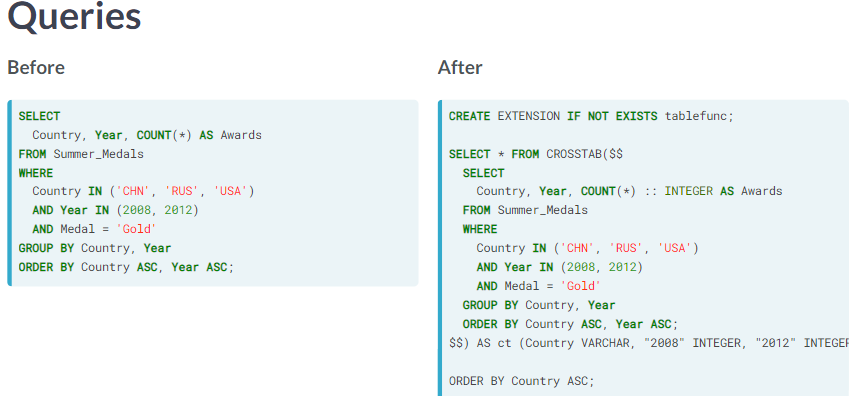
a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.





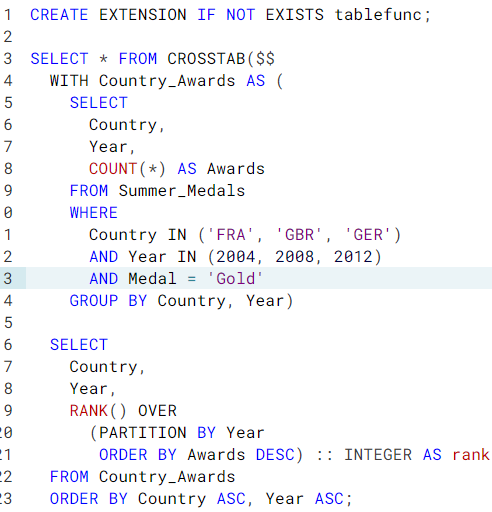
˖Pivoting table: allows you to pivot table by a single column

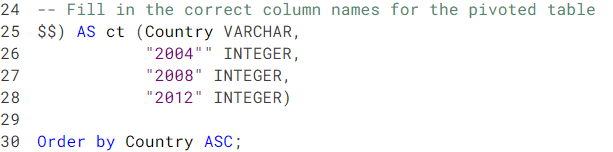






Example:

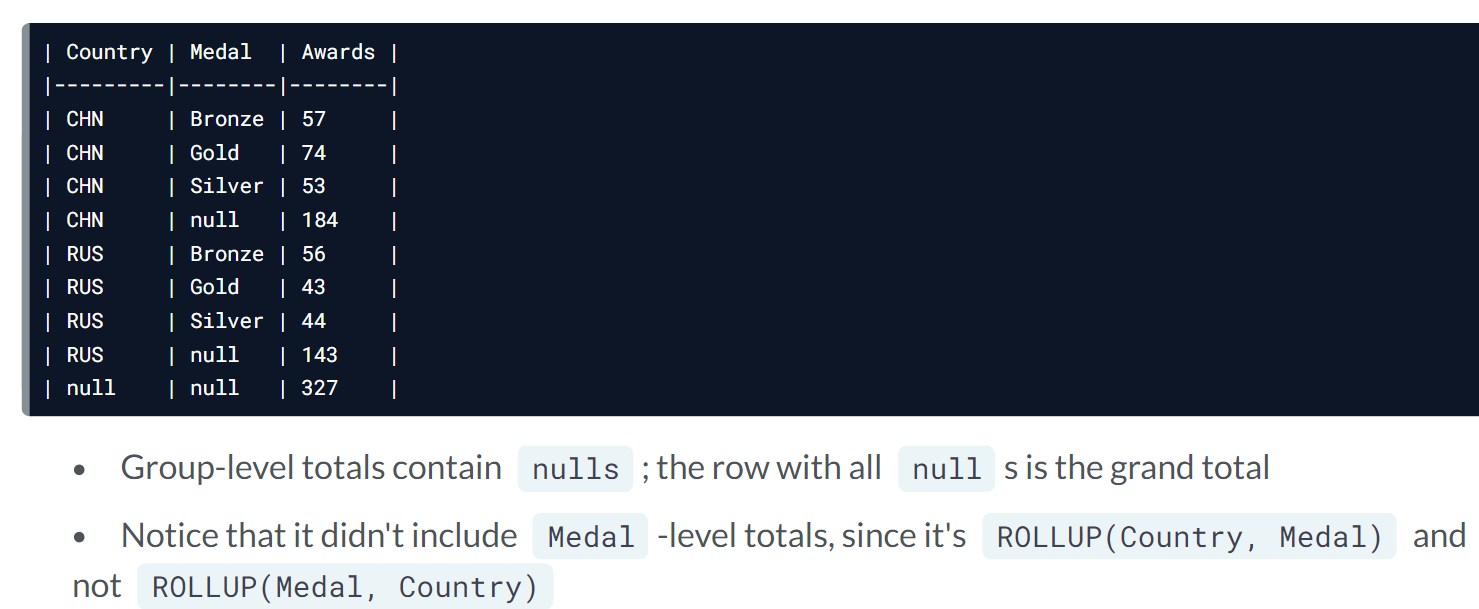




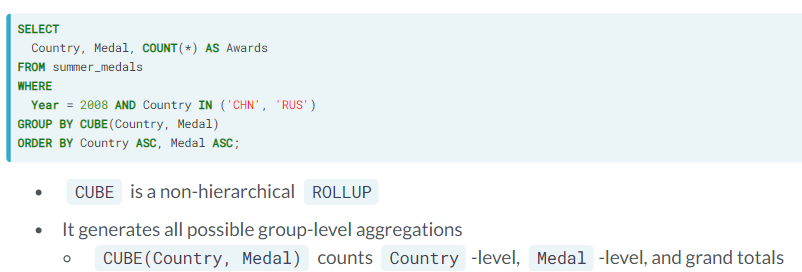
˖Roll up



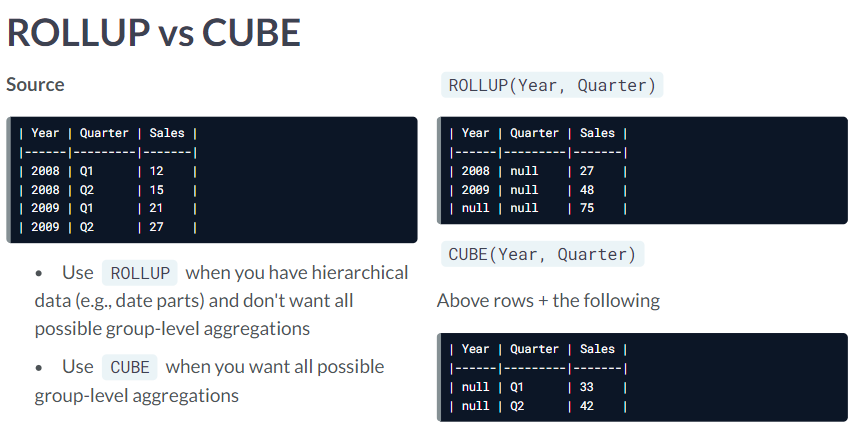




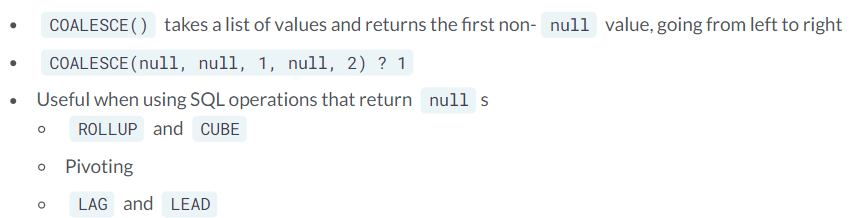
˖Cube

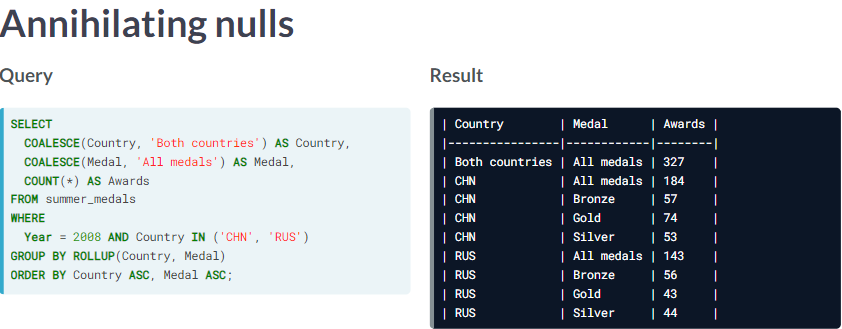




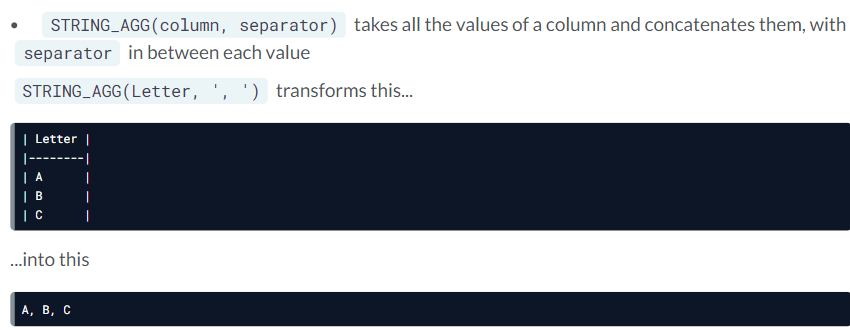


˖Coalesce function: replace ‘nulls’ from a table and name it as a new phase



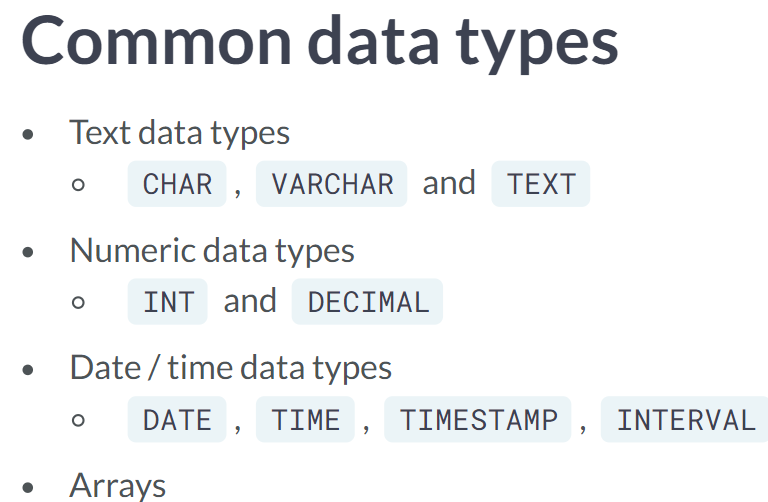


˖String\_AGG function: to compress data

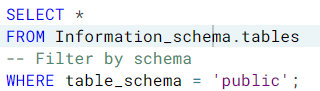


**Functions for Manipulating Data in PostgreSQL**

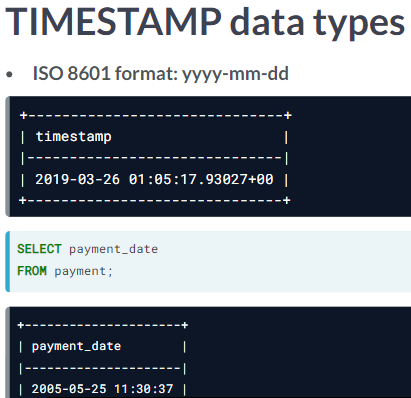
˖Data types







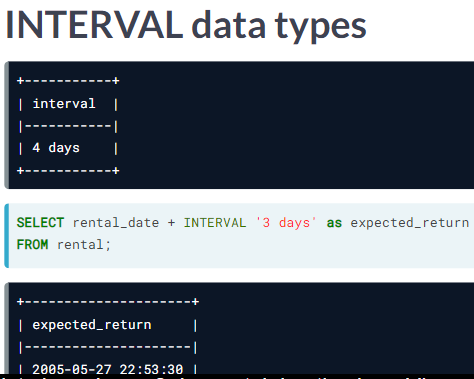
˖Day and time data type



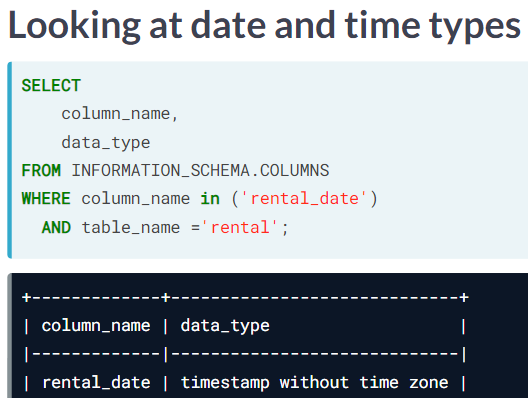
contains a data value and time value with microsecond precision. Used to record an exact a point of time or record was last updated.



Are essentially the date and time value of timestamp



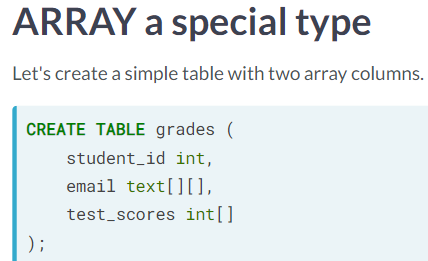
Store data and time data as a period of time in years, months, days, seconds ect. It is useful when you want to do arithmetic on date and time columns



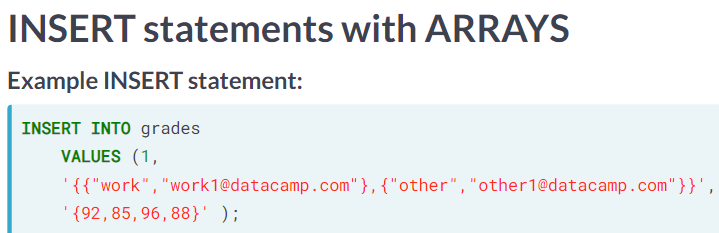
Without timezone is default.

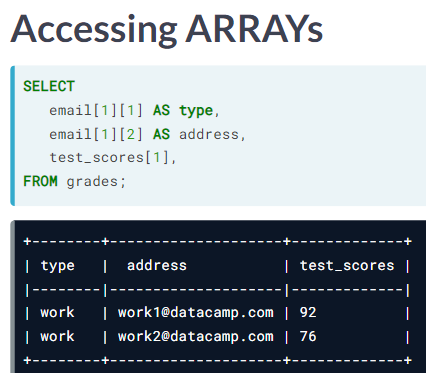
˖Working with arrays



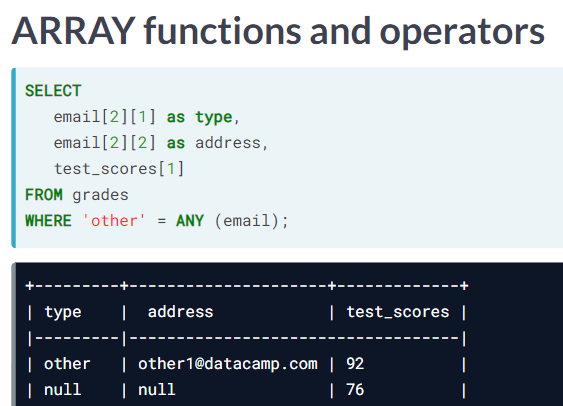
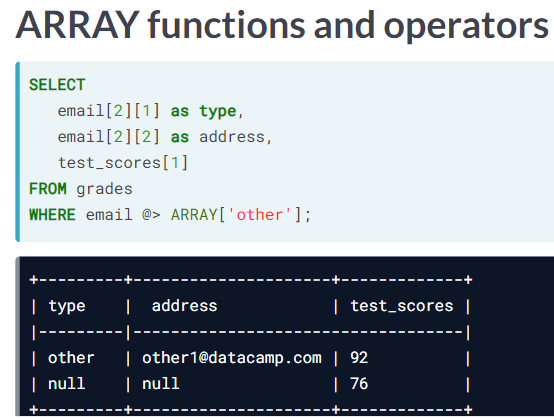


Text[] [], it is a nested array.





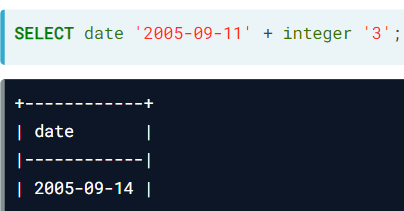
Email[1][1], email[1][2]: those are notation with index. Postgresql index starts with 1 not 0.

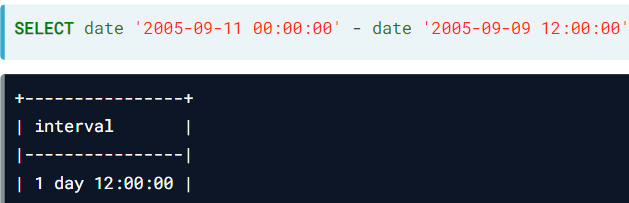
 

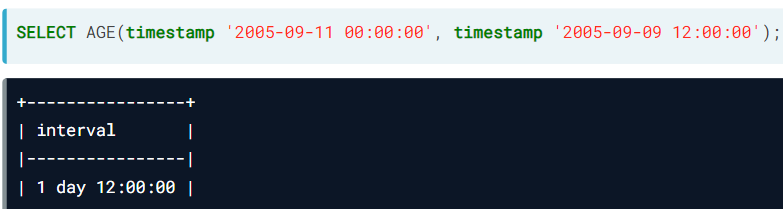
They are substitiute functions.

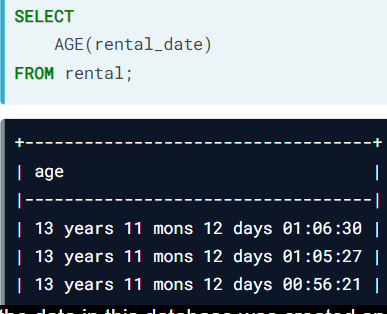
˖Adding and subtracting data/time data



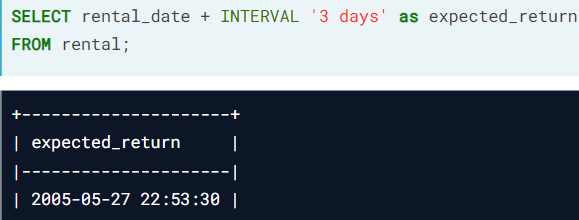


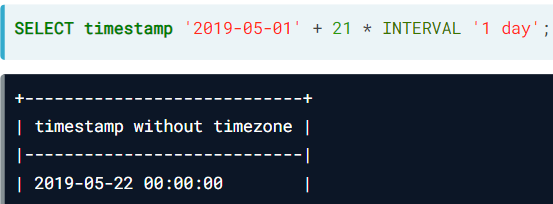


˖Age function: allow us to calculate the difference between two timestamps 

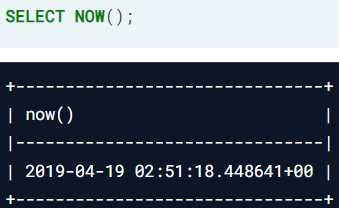


˖Interval clause

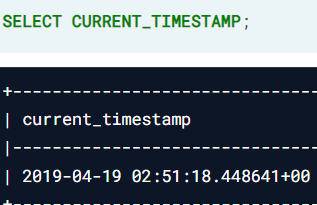
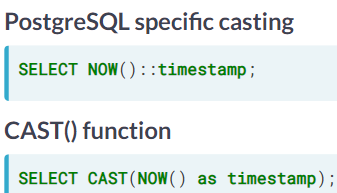


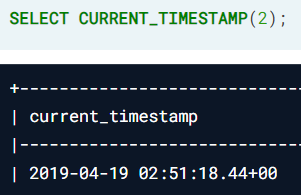


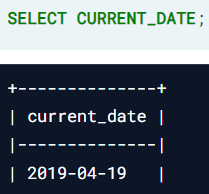
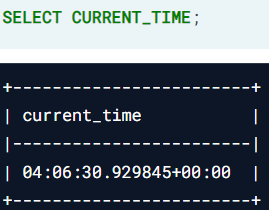
˖ Functions for retrieving current date/time

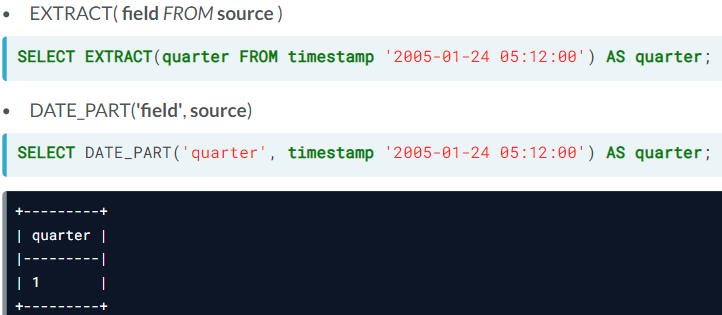
With timezone without timezone

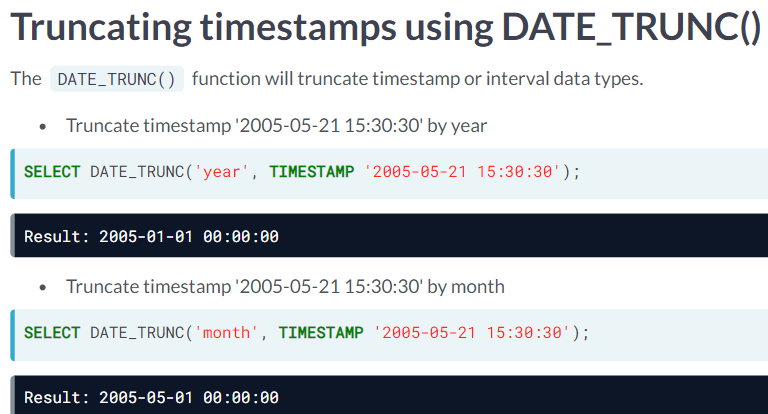


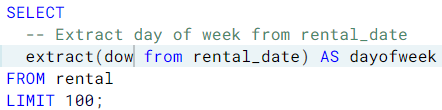
 2 is decimal point precison

˖Extracting and transforming date/ time data

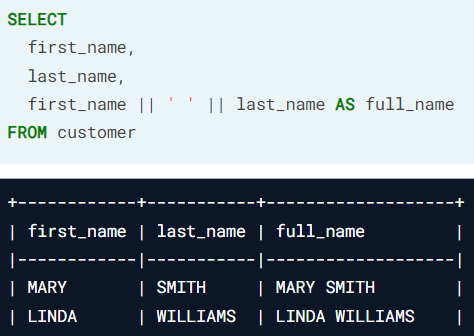
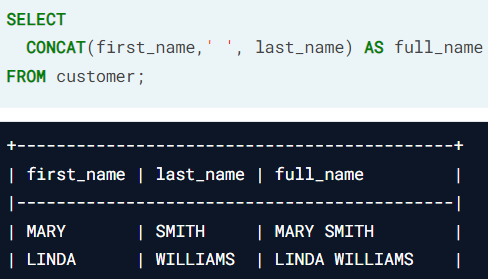


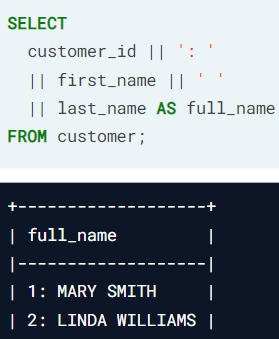




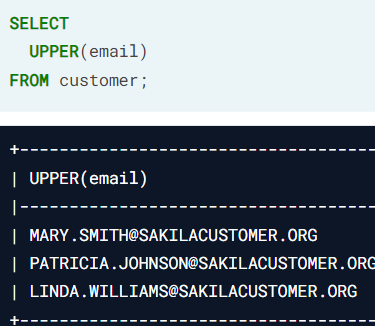
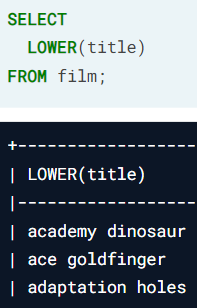
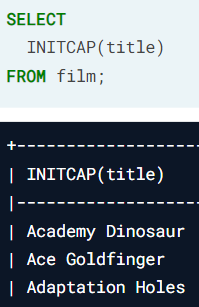
DOW = day of week statement

˖The string concatenation operator

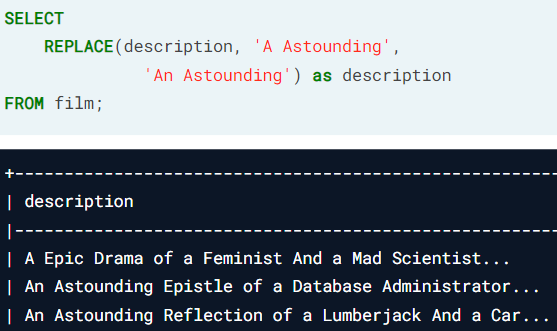
 



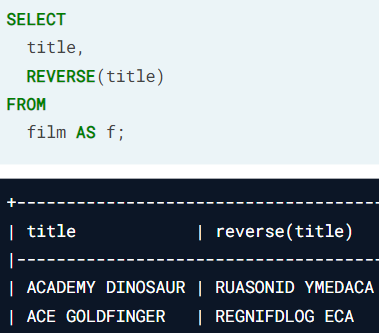
˖Change the case of string

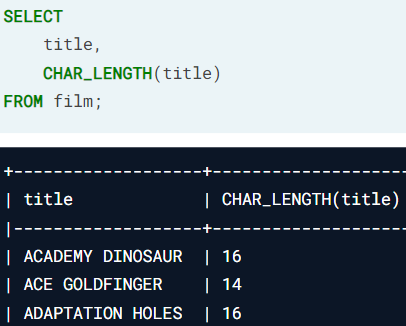
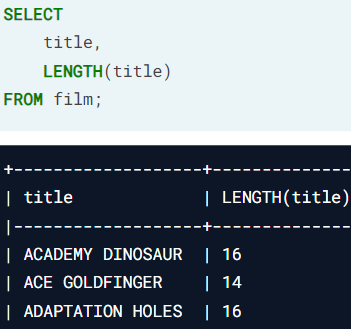
˖Replace characters in a string

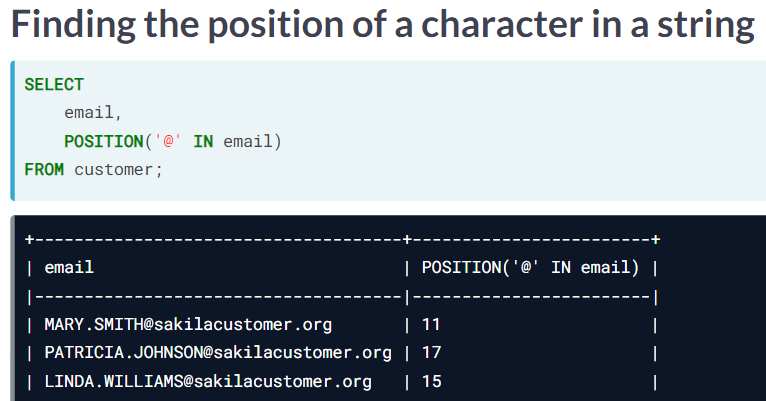


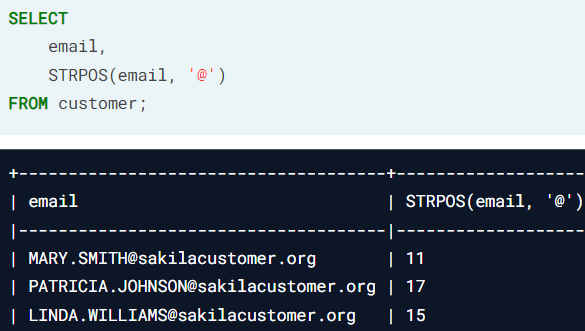
˖Reverse function

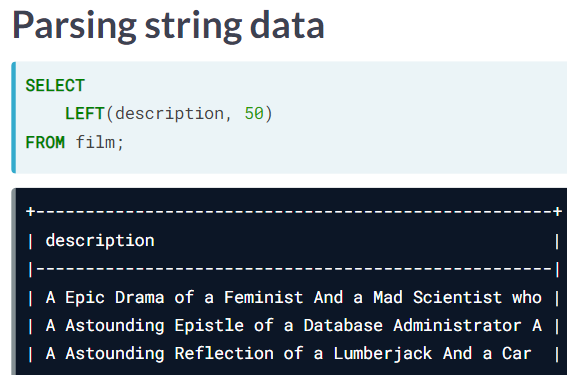


˖ Parsing string and character data

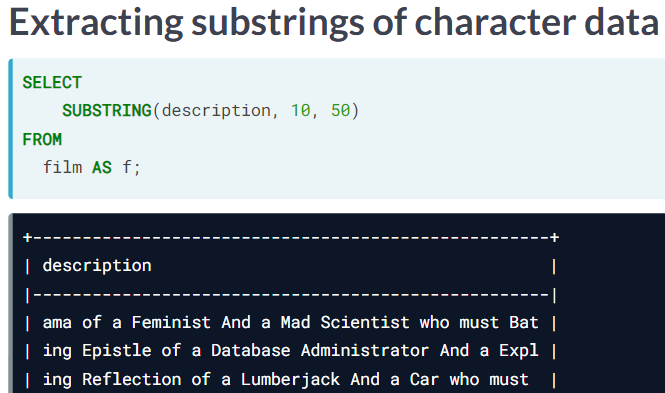


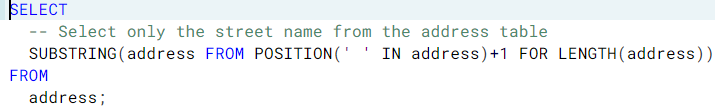




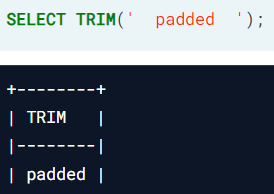
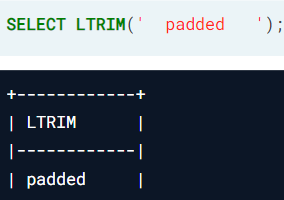
The first 50 characters from the left.

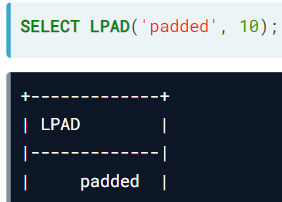
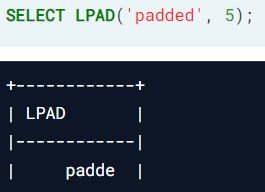
RIGHT (description, 50) is similar. The first 50 characters from the right.



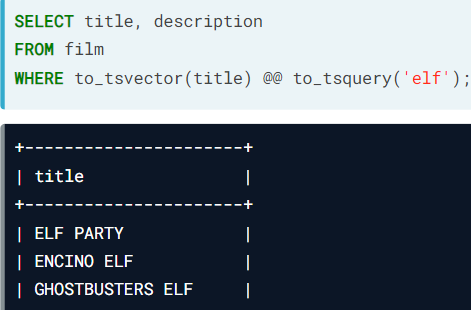


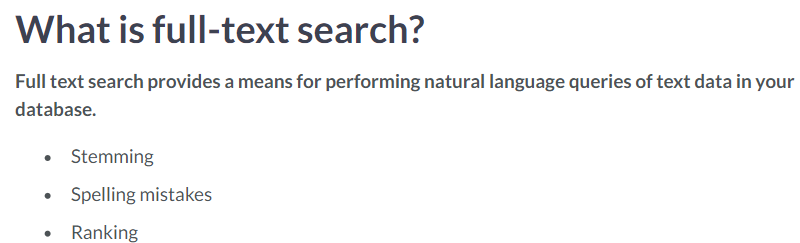
˖ Truncating and padding string data

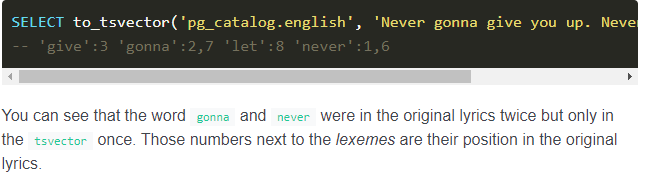
 

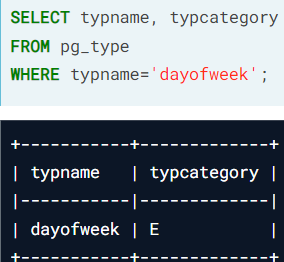
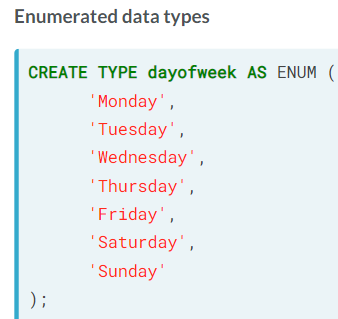
˖Full text search

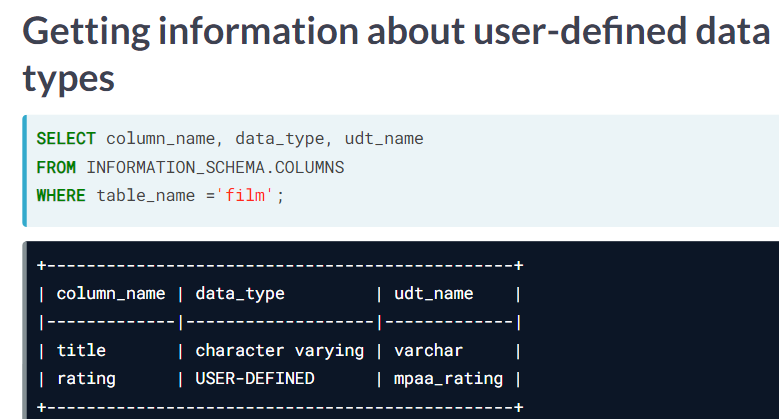
 case insensitive



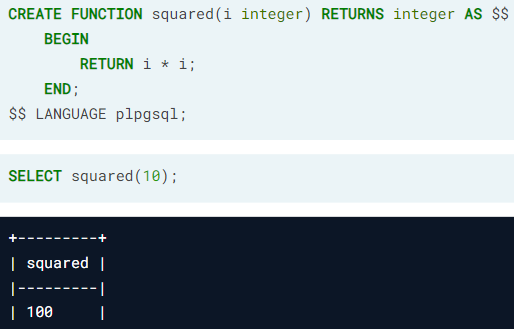
convert the description column to tsvector data type which is used for full text search. A tsvector value merges different variants of the same word and removes duplicates to create a sorted list of distinct words called lexemes. 

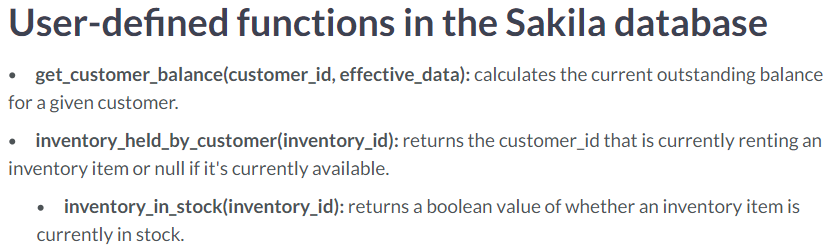
˖User-defined datatypes





˖User-defined functions





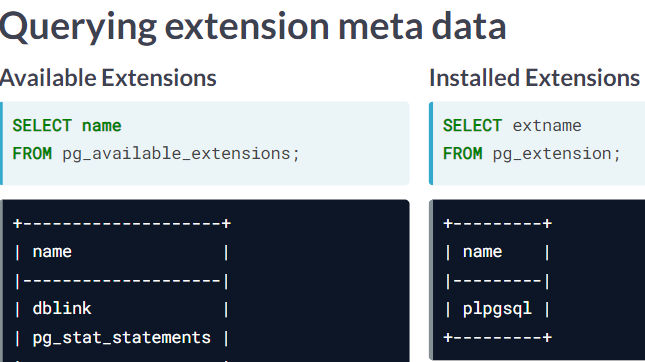
˖Intro to postgresql extensions

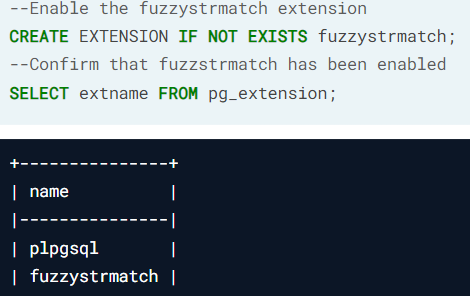
Commonly used extensions:

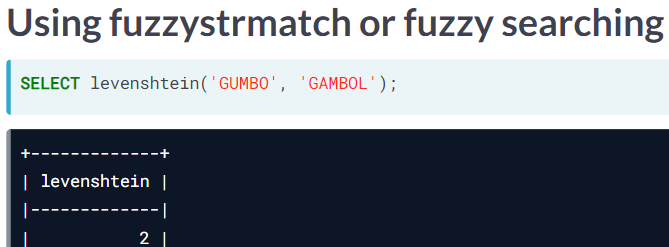
1, PostGIS: adds support for allowing location queries to run in sql

2, PostPIC: allows image processing within database

3, Fuzzystrmatch and pg\_trgm: provide functions that extend full text search capabilities by finding similarities between strings







Levenshtein calculates the levenstein distance between two strings which is the number of edits required for the strings to be perfectly match



Similarity number should be 0 to 1. 0 means no match, 1 means perfectly match