PSEUDO-SQL Cheat Sheet

by Gaston via cheatography.com/24027/cs/47539/

General format

General idea	PSEUDO-SQL (PQL in this context) is a formal data processing language with non-formal data definitions	
formal	In the scope of this language we speak of "formal" when the keywords and syntax is exactly defined like in a programming language or real SQL	
non-formal	In the scope of this document non-formal elements, if thosw elements are not, or not fully defined and are to be stated in such a wayx that they make sense in plain language. So we may for example reference a full-address in a statement where as in a real database this would be split into several fields	
Keywords & Syntax	The language keywords (i.e. language tokens) and syntax are formally defined. But informal statements may be introduced by the user	
Basic PQL statment	The basic PQL statement differs from the standard SQL in that the destination is usually only a name for the result of the statement. The destination and source may have the same name, stating that the source record set is to be updated. In general the result of this statement is a new destination record set which would override any existing one with same name	SELECT destination FROM source ...
Record sets	A record set is a list of records which is not formally defined. I.e. it is not mandatory that all records explicitely have the same columns available (see Missing columns).	
Initial record set(s)	Initial record sets are those which are used as source without being created by any previous PQL statement. In most cases it will not be necessary to formally introduce these as they should be informally comprehensible.	

General format (cont)

Identifiers	Identifiers are names which are composed of alphanumeric character, underscore and/or minus but no space. Identifiers shall also start with a letter or underscore	
Informal columns	Column names should be informal, and thus meaning full by their name. Columns are virtual and can represent any simple or complex concept (like the employee-office-number-and-address)	
Formal columns	Sometimes informal column identifiers may get too cumbersome to describe its full usage and scope. In these cases columns may be formally defined using an informal description. The recordset name is optional in case the column is to be globally available	DEFINE [<recordset>.]<column> AS <desciption>
	This definition simply reuses a previous column definition and may optionally alter its definition via another informal description	DEFINE <recordset>.<column> = [<recordset>].column> BUT <description>
Missing columns	As described in the record sets topic, records in a record set are not expected to each have the same set of columns. This is handled by a set of formal rules below	
	RULE 1: tests	Missing columns being assumed empty they behave like empty columns in any test
	RULE 2: assignment	Any value assigned to an non-existent column virtually creates that column in the given record. This rule generally allows to add new columns to any recordsetz
WHERE condition	Just like in plain SQL, PSQ allows to filter records via the WHERE keyword. The conditions are PQL specific however	

General format (cont)

JOINS	Joins being pure SQL database technical statments, they are not used in PQL. If at any time select data from several record sets are required as source, a record set should be previously created by a PQL statement making it easier comprehensible

Conditions

'condition' or ('condition')	Informal conditions are to be written in single quotes. Additional parenthesis may be added for better readability where required
NOT | AND | OR	Boolean operators in order of precedence (highest to lowest)
(condition)	Conditions can be grouped in parenthesizes to manage proper precedence