|  |  |  |  |
| --- | --- | --- | --- |
| *VTL-ML operators* | *Current syntax in the Reference Manual 1.1* | *Example* | *Example using the new syntax* |
| Analytic function | lag **(** ds { , offset { , default } } **) over (** { *partitionBy* } { *orderBy* } { *windowingClause*} **)**  *partitionBy* ::= **partition** **by** *c\_p*  { **,** *c\_p* } \*  *orderBy* ::= **order by**  *c\_o*{ **,** *c\_o* } \* { [ **asc** | **desc** ] }  *windowingClause* **::=**  [ **rows** | **range** ]  **between** *windowSpec* **and** *windowSpec*  *windowSpec* **::=**  [ num **preceding** |num **following** | **current row** | **unbounded preceding** | **unbounded following** | lag ( ds, 1, 0 ) over ( order by time\_period )  avg ( ds ) over ( order by time\_period rows between 1 preceding and 1 following )  sum ( ds ) over ( order by time\_period rows between current row and unbounded preceding ) | lag ( ds, offset := 1, default := 0, order\_by := [ time\_period ] )  avg ( ds, order\_by := [ time\_period ] , rows1 := 1, direction1 := preceding, rows2 := 1, direction2 := following ] )  sum ( ds, order\_by := [ time\_period ] , direction1 := current\_row , direction2 := unbounded\_preceding ] ) |
| Aggregate function | sum **(** *ds* **)** { [ **group by** | **along** ] **(** *idComp* {, *idComp* }\***)** } | sum ( ds ) group by ref\_area, partner  sum ( ds ) along time\_period  count ( ds ) | sum ( ds, group\_by := [ ref\_area, partner ] )  sum ( ds, along := [ time\_period ] )  count ( ds ) |
| Time aggregate | sum **(** *ds* ) time\_aggregate (  freq\_from  ,freq\_to  { , *minPeriods* }  { , *timePeriodName* }  { *, timeFormatFrom* }  { *, timeFormatTo* }  **)** | sum ( ds ) time\_aggregate ( "Q", "A" ) | sum ( ds, freq\_to := "Q", freq\_from := "A" ) |
| Time series function | **fill\_time\_series (** *ds, freq,* { , *timePeriodName* *{ , timeFormat* }**)** | fill\_time\_series ( ds, "Q", time\_period ) | fill\_time\_series ( ds, freq := "Q", time\_period\_name := time\_period ) |
| Clause | *clause* := *calc\_clause* | *drop\_clause* | *filter\_clause* | *keep\_clause* | *rename\_clause* | *unfold\_clause* | *fold\_clause*  *calc\_clause* := {role} *compName* **=** *k*  *drop\_clause* ::= **drop** {cmp{ **,** cmp}\*}  *keep\_clause* ::= **keep** {cmp{ **,** cmp}\*}  *filter\_clause* ::= **filter** boolean-expression | dpr  *rename\_clause* ::= **rename** cmp **to** cmp **{ ,** cmp **to** cmp}  *unfold\_clause* ::= **unfold** dim **,** msr **to** elem { **,** elem }  *fold\_clause* ::= **fold** elem { **,** elem } **to** dim **,** msr  *role*:= **identifier** | **measure** | **attribute**  **Question**: what is the type of [ comp1, comp2, comp3 ] | ds [ filter ref\_area = "IT" ]  ds [ m = 1 ]  ds [ keep comp1, comp2, comp3 ]  unfold ref\_area, obs\_value to ["EU25", "BG", "RO"] | ds [ filter ref\_area = "IT" ]  ds [ m := 1 ]  ds [ keep [ comp1, comp2, comp3 ] ]  unfold ( dimension := ref\_area, measure := obs\_value, elements := ["EU25", "BG", "RO"] ) |