ESS Validat

Single Grant Agreement B2794-2018-ESS-VAL-IMPL

http://www.cros-portal.eu/

Single Grant Agreement Number **553-826006-2018-PT-ESS-VAL-IMPL**

Hybrid Validation Implementation

Deliverable 11.14.2

Manual Translation of the Main Types of Validation Rules from VTL to SQL

Version: 1.0

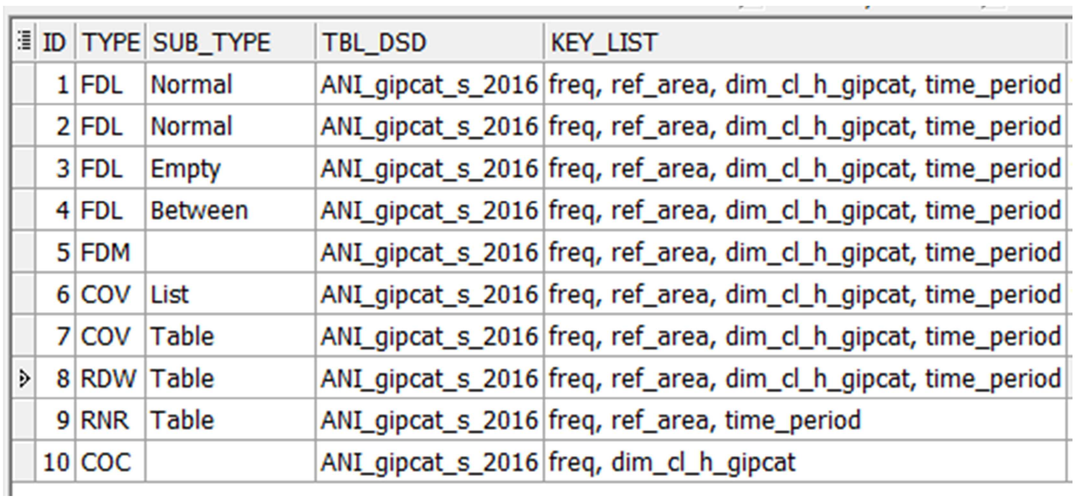
Manual translation of the main types of Rules from VTL to SQL

The current Hybrid Validation Implementation Project[1] (HyVImp) at INE includes implementation of Scenario 1 as described in the Business Architecture for ESS Validation[2]. For storing and executing the rules in SQL we use our Statistical Data Warehouse (SDW) on Oracle databases, where our data is already integrated.

Based on the work that identified the most common types of validation rules using VTL[3] we decided to develop a template. However the rules that we generated automatically in SQL through Eurostat's Interpreter and Sandbox Interface[4] (EISI) using the VTL rules was complex to understand. Our idea was that once the rule is pre-formatted, using it, would be a matter of parameterization, that could be achieved using the metadata of the chosen domain. This proved difficult with the generated SQL see deliverable 10 of task 13 developed in work package 2 (D10_T13_2.docx).

In this report we will explain first the structure of the table to store the SQL rules and then the rules themselves. The procedure that executes the rules generates automatically an execution log similar to what we have seen in EISI.

The table which stores each VTL type of rule for SQL execution is called VTLCTRL, and it stores the type and subtype of rule, according to the VTL main type of rules document (footnote 3) and also the DSD to which it is being applied in the current instance.



ID	TYPE	SUB_TYPE	TBL_DSD	KEY_LIST
1	FDL	Normal	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period
2	FDL	Normal	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period
3	FDL	Empty	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period
4	FDL	Between	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period
5	FDM		ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period
6	COV	List	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period
7	COV	Table	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period
8	RDW	Table	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period
9	RNR	Table	ANI_gipcat_s_2016	freq, ref_area, time_period
10	COC		ANI_gipcat_s_2016	freq, dim_cl_h_gipcat

Figure 1 - VTLCTRL table where SQL rules are stored

Besides the DSD, whose name matches a corresponding table in the current oracle schema it is also indicated the key list. This is the list of elements that form the record key and that will make its identification possible in case of errors.

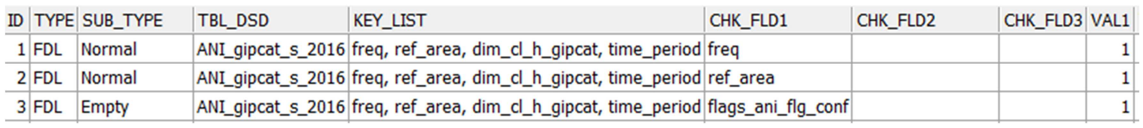
[1] ESS Grant: 553-826006-2018-PT-ESS-VAL-IMPL

[2] https://ec.europa.eu/eurostat/cros/system/files/business_architecture_for_ess_validation_-_final.pdf

[3] https://ec.europa.eu/eurostat/cros/system/files/02c._main_types_of_data_validation_rules_and_fictive_domain_0.pdf

[4] https://github.com/eurostat/VTL

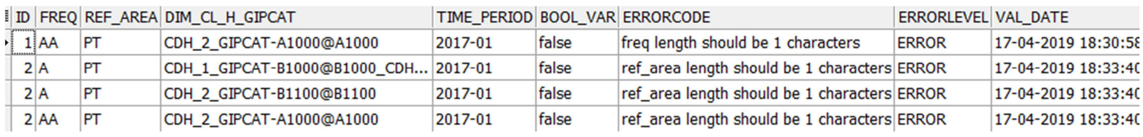
The first columns, described above, document the type of rule for reference and the DSD it applies to. The second set of columns, receive the parameters for the execution of the rule.



ID	TYPE	SUB_TYPE	TBL_DSD	KEY_LIST	CHK_FLD1	CHK_FLD2	CHK_FLD3	VAL1
1	FDL	Normal	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period	freq			1
2	FDL	Normal	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period	ref_area			1
3	FDL	Empty	ANI_gipcat_s_2016	freq, ref_area, dim_cl_h_gipcat, time_period	flags_ani_flg_conf			1

Figure 2 - VTLCTRL table showing the check field columns

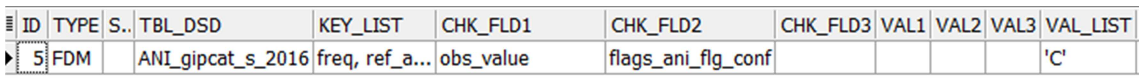
There are 3 columns for check fields, i.e. fields to be checked during rule execution and 3 columns for values, or thresholds to be applied by the rules. In the example in figure 3 there are three FDL type rules, which will check the field length. In each case the field is identified in chk_fld1 and the limit value in val1. In addition to check for the value the third rule also checks for it being empty. The execution log is generated by the execution of validation as explained in the next section and generates the following kind of logs.



ID	FREQ	REF_AREA	DIM_CL_H_GIPCAT	TIME_PERIOD	BOOL_VAR	ERRORCODE	ERRORLEVEL	VAL_DATE
1	AA	PT	CDH_2_GIPCAT-A1000@A1000	2017-01	false	freq length should be 1 characters	ERROR	17-04-2019 18:30:5
2	A	PT	CDH_1_GIPCAT-B1000@B1000_CDH...	2017-01	false	ref_area length should be 1 characters	ERROR	17-04-2019 18:33:4
2	A	PT	CDH_2_GIPCAT-B1100@B1100	2017-01	false	ref_area length should be 1 characters	ERROR	17-04-2019 18:33:4
2	AA	PT	CDH_2_GIPCAT-A1000@A1000	2017-01	false	ref_area length should be 1 characters	ERROR	17-04-2019 18:33:4

Figure 3 - Log table TLOG_ANI_GIPCAT_S_2016_FDL

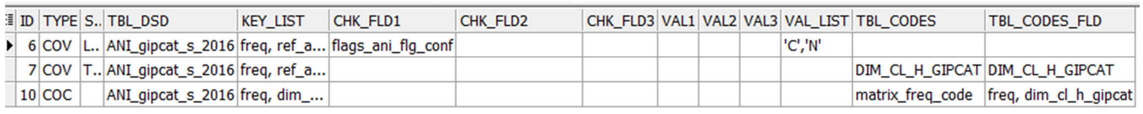
Different rules have distinct requirements, hence the existence of 3 check fields columns, and 3 different values. Other parameters may be required as in the case of the FDM type of rule. Field is mandatory or empty has ordinarily an exception when the flag signals confidentiality. In this case the value or list of values that may be accepted has to be indicated.



ID	TYPE	S..	TBL_DSD	KEY_LIST	CHK_FLD1	CHK_FLD2	CHK_FLD3	VAL1	VAL2	VAL3	VAL_LIST
5	FDM		ANI_gipcat_s_2016	freq, ref_a...	obs_value	flags_ani_flg_conf					'C'

Figure 4 - VTLCTRL table showing the val_list column use

The 3 columns for value do not admit lists of values and for this case we have the column val_list.



ID	TYPE	S..	TBL_DSD	KEY_LIST	CHK_FLD1	CHK_FLD2	CHK_FLD3	VAL1	VAL2	VAL3	VAL_LIST	TBL_CODES	TBL_CODES_FLD
6	COV	L..	ANI_gipcat_s_2016	freq, ref_a...	flags_ani_flg_conf						'C','N'		
7	COV	T..	ANI_gipcat_s_2016	freq, ref_a...								DIM_CL_H_GIPCAT	DIM_CL_H_GIPCAT
10	COC		ANI_gipcat_s_2016	freq, dim_...								matrix_freq_code	freq, dim_cl_h_gipcat

Figure 5 - VTLCTRL table showing the columns for codelists

Finally for checking values we can also check if the values belong to a code list. In that case, if we don't want to insert all the values in the val_list field we can use the fields tbl_codes and tbl_codes_fld together. The table which has the codes must be inserted in tbl_codes and the field(s) that should match these codes must be in tbl_codes_fld.

Naturally not all logs will be equal and there are as many log tables as types of rules. All the rules sharing the same type will be inserted in the correspondent TLOG_<DSD>_TYPE, as shown in figure 3. The rules themselves are described in the next section.

FDL – FielD Length

Fixed length

VTL Rule

```
DATASET:= ANI_gipcat_s_2016;
check  ( length(DATASET#freq) =1
errorcode "Freq length should be 1 character"
errorlevel "ERROR")
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
CASE
   WHEN length(freq)=1 THEN 'true'
   WHEN length(freq)<>1 THEN 'false' END  AS "BOOL_VAR",
   CASE
  WHEN NOT (length(freq) = 1) THEN 'Freq length should be 1 character' END AS
"ERRORCODE",
CASE
  WHEN NOT (length(freq) = 1) THEN 'ERROR' END AS "ERRORLEVEL"
 FROM ANI_gipcat_s_2016
 ORDER BY freq, ref_area, dim_cl_h_gipcat, time_period
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1 := freq;
val1 := 1;

SELECT ' || key_list || ',
                    CASE
                        WHEN length(' ||chk_fld1 || ')=' ||val1 || ' THEN ''true''
                        WHEN length(' ||chk_fld1 || ')<>' ||val1 || ' THEN ''false'' END  AS BOOL_VAR,
                    CASE
                        WHEN NOT (length(' ||chk_fld1 || ') = ' ||val1 || ') THEN ''' ||chk_fld1 || '
length should be ' ||val1 || ' characters''  END AS ERRORCODE,
                    CASE
                        WHEN NOT (length(' ||chk_fld1 || ') = ' ||val1 || ') THEN ''ERROR'' END AS
ERRORLEVEL, sysdate as VAL_DATE
FROM ' ||tbl_dsd
ORDER BY ' ||key_list;
```

Fixed length or Empty

VTL Rule

```
DATASET:= ANI_gipcat_s_2016;
check  ( isnull(DATASET#flags_ani_flg_conf)  or length(nvl(DATASET#flags_ani_flg_conf,0))=1
errorcode "Conf flag should be 1 character or empty"
errorlevel "ERROR")
```

SQL Rule

```
SELECT freq ,ref_area ,dim_cl_h_gipcat ,time_period,
CASE
        WHEN length(nvl(flags_ani_flg_conf,0))=1 THEN 'true'
        WHEN length(nvl(flags_ani_flg_conf,0))<>1 THEN 'false' END
         AS "BOOL_VAR",
CASE
WHEN NOT (flags_ani_flg_conf IS NULL  or length(nvl(flags_ani_flg_conf,0)) = 1) THEN 'Conf
flag should be 1 character or empty' END AS "ERRORCODE",
CASE
WHEN NOT (flags_ani_flg_conf IS NULL  or length(nvl(flags_ani_flg_conf,0)) = 1) THEN
'ERROR' END AS "ERRORLEVEL"
FROM ANI_GIPCAT_S_2016 a
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1 := flags_ani_flg_conf;
val1 := 1;

SELECT ' || key_list || ',
CASE
       WHEN length(nvl(' ||chk_fld1 || ',0)) =' ||val1 || ' THEN ''true''
       WHEN length(nvl(' ||chk_fld1 || ',0)) <>' ||val1 || ' THEN ''false'' END  AS BOOL_VAR,
CASE
       WHEN NOT (' || chk_fld1 || ' IS NULL OR length(nvl(' ||chk_fld1 || ',0))  = ' ||val1 || ')
THEN ''' ||chk_fld1 || ' length should be ' ||val1 || ' characters or Empty''  END AS ERRORCODE,
CASE
       WHEN NOT (' || chk_fld1 || ' IS NULL OR length(nvl(' ||chk_fld1 || ',0))  = ' ||val1 || ')
THEN ''ERROR'' END AS ERRORLEVEL, sysdate as VAL_DATE
FROM ' ||tbl_dsd
ORDER BY ' ||key_list;
```

Length between two values

VTL Rule

```
DATASET:= ANI_gipcat_s_2016;
check  ( length(nvl(DATASET#dim_cl_h_gipcat,0))>=24 and
length(nvl(DATASET#dim_cl_h_gipcat,0))<=34
errorcode "Class Gipcat lenght should be between 24 and 34"
errorlevel "ERROR")
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
CASE
   WHEN length(dim_cl_h_gipcat) between 24 and 34 THEN 'true'
   else 'false' END  AS "BOOL_VAR",
CASE
  WHEN NOT (length(dim_cl_h_gipcat) between 24 and 34 ) THEN 'Class Gipcat lenght should
be between 24 and 34' END AS "ERRORCODE",
CASE
 WHEN NOT (length(dim_cl_h_gipcat) between 24 and 34 ) THEN 'ERROR' END AS
"ERRORLEVEL"
 FROM ANI_gipcat_s_2016
 ORDER BY freq, ref_area, dim_cl_h_gipcat, time_period
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1 := dim_cl_h_gipcat;
val1 := 24;
val2 := 34;

SELECT ' || key_list || ',
CASE
      WHEN length(' ||chk_fld1 || ') Between ' ||val1 || ' AND ' || val2 || ' THEN ''true''
      ELSE ''false'' END  AS BOOL_VAR,
CASE
       WHEN NOT (length(' ||chk_fld1 || ') Between ' ||val1 || ' AND ' || val2 || ') THEN '''
||chk_fld1 || ' length should be between ' ||val1 || ' and ' ||val2 || ' characters''  END AS
ERRORCODE,
CASE
       WHEN NOT (length(' ||chk_fld1 || ') Between ' ||val1 || ' AND ' || val2 || ') THEN ''ERROR''
END AS ERRORLEVEL, sysdate as VAL_DATE
FROM ' ||tbl_dsd
ORDER BY ' ||key_list;
```

Decimal Length

VTL Rule

```
DATASET:= ANI_gipcat_s_2016;
check((instr(DATASET#obs_value,".")<>0 and length(DATASET#obs_value)-
instr(DATASET#obs_value,".")<2) or instr(DATASET#obs_value,".")=0
errorcode "Observation value must be an integer or have just one decimal"
errorlevel "ERROR")
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
          CASE
             WHEN (obs_value is NULL or instr(obs_value,',')=0 or length(obs_value) -
instr(obs_value,',')<=1 ) THEN 'true' else 'false' END  AS BOOL_VAR,
          CASE
             WHEN (obs_value is not NULL and not instr(obs_value,',')=0 and length(obs_value)
- instr(obs_value,',')>1 ) THEN  'Observation value must be an integer or have just one decimal
' END AS ERRORCODE,
          CASE
             WHEN(obs_value is not NULL and not instr(obs_value,',')=0 and length(obs_value) -
instr(obs_value,',')>1 ) THEN 'ERROR' END AS ERRORLEVEL, sysdate as VAL_DATE
          FROM   ANI_gipcat_s_2016
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1 := obs_value;
val1 := 1;
```

```
SELECT  ' || num ||' as ID,'|| key_list || ',
               CASE
                  WHEN ' ||chk_fld1 || ' IS NULL OR INSTR(' ||chk_fld1 || ',",")=0 OR LENGTH('
||chk_fld1 || ') - INSTR(' ||chk_fld1 || ',",") <= ' ||val1 || ' THEN "true"
                  ELSE "false" END  AS BOOL_VAR,
               CASE
                 WHEN ' ||chk_fld1 || ' IS NOT NULL AND NOT INSTR(' ||chk_fld1 || ',",")=0
AND LENGTH(' ||chk_fld1 || ') - INSTR(' ||chk_fld1 || ',",") > ' ||val1 || ' THEN "Observation
value must be an integer or have just ' ||val1 || ' decimal "  END AS ERRORCODE,
               CASE
                  WHEN ' ||chk_fld1 || ' IS NOT NULL AND NOT INSTR(' ||chk_fld1 || ',",")=0
AND LENGTH(' ||chk_fld1 || ') - INSTR(' ||chk_fld1 || ',",") > ' ||val1 || ' THEN "ERROR" END
AS ERRORLEVEL, sysdate as VAL_DATE
               FROM ' ||tbl_dsd;
               execute immediate v_select;
```

FDM – FielD is Mandatory or empty

VTL Rule

```
DATASET:= ANI_gipcat_s_2016;
check  ( not isnull(DATASET#obs_value) or nvl(DATASET#flags_ani_flg_conf,0) = "C"
errorcode "Animals should not be empty when FLAGS_ANI_FLG_CONF <> C"
errorlevel "ERROR")
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
CASE
    WHEN obs_value IS NULL and nvl(flags_ani_flg_conf,0)<>'C'  THEN 'false'
    ELSE 'true'
END as BOOL_VAR,
CASE
WHEN  obs_value IS NULL and nvl(flags_ani_flg_conf,0)<>'C' THEN 'Animals should not be
empty when FLAGS_ANI_FLG_CONF <> C' END AS "ERRORCODE",
CASE
WHEN  obs_value IS NULL and nvl(flags_ani_flg_conf,0)<>'C' THEN 'ERROR' END AS
"ERRORLEVEL"
FROM ANI_gipcat_s_2016
ORDER BY 1,2,3,4
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1 := obs_value;
chk_fld2 := flags_ani_flg_conf;
val_list := 'C';

SELECT ' || key_list || ',

CASE
    WHEN ' ||chk_fld1 || ' IS NULL AND NVL(' ||chk_fld2 || ',0) NOT IN (' || val_list || ') THEN
''false''
    ELSE ''true'' END  AS BOOL_VAR,
CASE
    WHEN ' ||chk_fld1 || ' IS NULL AND NVL(' ||chk_fld2 || ',0) NOT IN (' || val_list || ') THEN '''
||chk_fld1 || ' should not be empty when  ' ||chk_fld2 ||  ' not in ' ||replace(val_list,'''','') || '''
END AS ERRORCODE,
CASE
     WHEN ' ||chk_fld1 || ' IS NULL AND NVL(' ||chk_fld2 || ',0) NOT IN (' || val_list || ') THEN
''ERROR'' END AS ERRORLEVEL, sysdate as VAL_DATE
FROM ' ||tbl_dsd
ORDER BY ' ||key_list;
```

COV – COdes are Valid

In List

VTL Rule

```
DATASET:= ANI_gipcat_s_2016;
check (isnull(DATASET#flags_ani_flg_conf) or DATASET#flags_ani_flg_conf in {"C", "N"}
errorcode "OBS_CONF is not in the valid list of codes"
errorlevel " ERROR ")
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
CASE
        WHEN flags_ani_flg_conf is null or flags_ani_flg_conf IN ('C','N') THEN 'true' ELSE 'false'
END  AS "BOOL_VAR",
CASE WHEN NOT ( flags_ani_flg_conf is null  or flags_ani_flg_conf IN ('C','N')) THEN
'OBS_CONF is not in the valid list of codes' END AS "ERRORCODE",
CASE WHEN NOT ( flags_ani_flg_conf is null  or flags_ani_flg_conf IN ('C','N')) THEN 'ERROR '
END AS "ERRORLEVEL"
FROM ANI_gipcat_s_2016
ORDER BY 1,2,3,4
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1 := flags_ani_flg_conf;
val_list := 'C', 'N';

SELECT ' || num ||'as ID,'|| key_list || ',
CASE
     WHEN ' ||chk_fld1 || ' IS NULL OR ' ||chk_fld1 || ' IN (' || val_list || ') THEN "true"
     ELSE "false" END  AS BOOL_VAR,
CASE
     WHEN NOT(' ||chk_fld1 || ' IS NULL OR ' ||chk_fld1 || ' IN (' || val_list || ')) THEN "'
||chk_fld1 || ' is not in the valid list of codes" END AS ERRORCODE,
CASE
     WHEN NOT(' ||chk_fld1 || ' IS NULL OR ' ||chk_fld1 || ' IN (' || val_list || ')) THEN "ERROR"
END AS ERRORLEVEL, sysdate as VAL_DATE
FROM ' ||tbl_dsd
ORDER BY ' ||key_list;
```

With codes list

VTL Rule

```
DATASET:= ANI_gipcat_s_2016;
check (DATASET#dim_cl_h_gipcat in dim_cl_h_gipcat
errorcode "Gipcat code is not valid"
errorlevel "ERROR ");
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
CASE
    WHEN dim_cl_h_gipcat  NOT IN (Select bDIM_CL_H_GIPCAT from DIM_CL_H_GIPCAT b)
THEN 'false'
   ELSE 'true' END  AS BOOL_VAR,
CASE
   WHEN dim_cl_h_gipcat  NOT IN (Select DIM_CL_H_GIPCAT from DIM_CL_H_GIPCAT)  THEN
'Gipcat code is not valid' END AS ERRORCODE,
CASE
    WHEN dim_cl_h_gipcat  NOT IN (Select DIM_CL_H_GIPCAT from DIM_CL_H_GIPCAT)  THEN
'ERROR' END AS ERRORLEVEL, sysdate as VAL_DATE
FROM ANI_gipcat_s_2016
ORDER BY 1,2,3,4
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1 := dim_cl_h_gipcat;
tbl_codes := DIM_CL_H_GIPCAT;
tbl_codes_fld := DIM_CL_H_GIPCAT;

SELECT ' || num ||' as ID,'|| key_list || ',
                CASE
                    WHEN ' ||chk_fld1 || ' NOT IN ( SELECT ' || tbl_codes_fld || ' FROM ' ||
tbl_codes || ') THEN ''false''
                   ELSE ''true'' END  AS BOOL_VAR,
                CASE
                   WHEN ' ||chk_fld1 || ' NOT IN ( SELECT ' || tbl_codes_fld || ' FROM ' ||  tbl_codes
|| ') THEN ''' ||chk_fld1 || ' is not valid'' END AS ERRORCODE,
                CASE
                    WHEN ' ||chk_fld1 || ' NOT IN ( SELECT ' || tbl_codes_fld || ' FROM ' ||
tbl_codes || ') THEN ''ERROR'' END AS ERRORLEVEL, sysdate as VAL_DATE
FROM ' ||tbl_dsd
ORDER BY ' ||key_list;
```

RWD – Records are Without Duplicate Id-keys

This check is implemented in a structural validation service or data loader but can be implemented also in VTL (additional check or using a logical key instead of structural key)

VTL Rule

```
ds:= ANI_gipcat_s_2016;
key_num := count (ds group by time_period, dim_cl_h_gipcat,freq,ref_area);
check (key_num = 1
errorcode "Duplicate keys found"
errorlevel "ERROR")
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
CASE
   WHEN B.key_num IS NULL THEN 'true' else 'false' END  AS BOOL_VAR,
CASE
   WHEN not(B.key_num IS NULL) THEN 'Duplicate keys found' END AS ERRORCODE,
CASE
   WHEN not(B.key_num IS NULL) THEN 'ERROR' END AS ERRORLEVEL, sysdate as VAL_DATE,
sysdate as VAL_DATE
   from (SELECT  freq||ref_area|| dim_cl_h_gipcat||time_period KEY, A.* FROM
ANI_gipcat_s_2016 a) A,
   (SELECT freq||ref_area||dim_cl_h_gipcat||time_period KEY, COUNT(1) key_num FROM
ANI_gipcat_s_2016 A group by  freq||ref_area||dim_cl_h_gipcat||time_period having
count(1)>1) b
where a.KEY=b.KEY(+)
ORDER BY 1,2,3,4;
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;


SELECT ' || num ||' as ID,'|| key_list || ',
         CASE
            WHEN B.key_num IS NULL THEN ''true'' else ''false'' END  AS BOOL_VAR,
         CASE
            WHEN not(B.key_num IS NULL) THEN ''Duplicate keys found'' END AS ERRORCODE,
         CASE
            WHEN not(B.key_num IS NULL) THEN ''ERROR'' END AS ERRORLEVEL, sysdate as
VAL_DATE
         FROM (SELECT ' || REPLACE(key_list,',','||') || ' AS KEY, A.* FROM '  ||tbl_dsd || ' A) A,
         (SELECT ' || REPLACE(key_list,',','||') || ' as KEY, COUNT(1) key_num FROM  '
||tbl_dsd || '
         GROUP BY ' || REPLACE(key_list,',','||') || ' HAVING COUNT(1)>1) B
         WHERE A.KEY=B.KEY(+)';
```

REP – Records Expected are Provided

VTL Rule

```
ds:= ANI_gipcat_s_2016;
time_1s := ds  [ sub time_period = "2017-01" ] [ filter ref_area = "PT" ];
time_2s := ds  [ sub time_period = "2017-02" ] [ filter ref_area = "PT" ];
check ( exists_in (time_1s, time_2s) and exists_in (time_2s, time_1s) not valid
errorcode "Both semesters should be provided"
errorlevel "Error");
```

or

```
ds:= ANI_gipcat_s_2016;
time_1s := ds  [ sub time_period = "2017-01" ] [ filter ref_area = "PT" ];
time_2s := ds  [ sub time_period = "2017-02" ] [ filter ref_area = "PT" ];
check ( exists_in (time_1s, time_2s) not valid
errorcode "If 1st semester is provided, the 2nd should also be"
errorlevel "Error");

check ( exists_in (time_1s, time_2s) and exists_in (time_2s, time_1s) not valid
errorcode "If 2nd semester is provided, the 1fs should also be"
errorlevel "Error");
```

```
ds:= ANI_gipcat_s_2016;
sub_gipcat := ds  [ sub dim_cl_h_gipcat = "CDH_2_GIPCAT-B1100@B1100" ] [ filter ref_area =
"PT" ];
tot_gipcat := ds  [ sub dim_cl_h_gipcat = "CDH_1_GIPCAT-B1000@B1000" ] [ filter ref_area =
"PT" ];
check (  exists_in (  sub_gipcat, tot_gipcat)  not valid
errorcode "If B1100 is provided so B1000 should be too"
errorlevel "Error");
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
            CASE
                WHEN B.val is null THEN 'false' else 'true' END  AS BOOL_VAR,
                        CASE
                WHEN B.val is null  THEN 'If 2017-01 is provided 2017-02 should be provided too.'
END AS ERRORCODE,
            CASE
                WHEN B.val is null  THEN  'ERROR' END AS ERRORLEVEL, sysdate as VAL_DATE
            FROM
            (SELECT freq||ref_area||dim_cl_h_gipcat AS KEY, A.* FROM ANI_gipcat_s_2016 A
WHERE time_period= '2017-01' and ref_area='PT') A,
            (SELECT freq||ref_area||dim_cl_h_gipcat as KEY, 1 val  FROM  ANI_gipcat_s_2016
B WHERE time_period= '2017-02' and ref_area='PT') B
         WHERE A.KEY=B.KEY(+)
UNION
SELECT ID, freq, ref_area, dim_cl_h_gipcat, time_period,
         CASE
            WHEN A.val is null THEN 'false' else 'true' END  AS BOOL_VAR,
                  CASE
            WHEN A.val is null  THEN 'If 2017-02 is provided 2017-01 should be provided too.'
END AS ERRORCODE,
         CASE
            WHEN A.val is null  THEN  'ERROR' END AS ERRORLEVEL, sysdate as VAL_DATE
         FROM
            (SELECT freq||ref_area||dim_cl_h_gipcat AS KEY, 1 val FROM ANI_gipcat_s_2016 A
WHERE time_period= '2017-01' and ref_area='PT') A,
            (SELECT freq||ref_area||dim_cl_h_gipcat as KEY, B.*   FROM  ANI_gipcat_s_2016 B
WHERE time_period= '2017-02' and ref_area='PT') B
         WHERE A.KEY(+)=B.KEY
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1:= time_period;
chk_fld2:= ref_area;
val1:= 2017-01;
val2:= 2017-02;
val3:= PT;
```

```
SELECT ' || num ||' as ID, '|| key_list || ',
         CASE
            WHEN (B.val IS NULL) THEN ''false'' else ''true'' END  AS BOOL_VAR,
         CASE
            WHEN (B.val IS NULL) THEN  ''If '|| val1 || ' is provided  '|| val2 || ' should be
provided too '' END AS ERRORCODE,
         CASE
            WHEN (B.val IS NULL) THEN ''ERROR'' END AS ERRORLEVEL, sysdate as VAL_DATE
         FROM
            (SELECT ' || REPLACE(key_list,',','||') || ' AS KEY, A.* FROM '  ||tbl_dsd || ' A
            WHERE '  ||chk_fld1 || '= '''  ||val1 || ''' and ' || chk_fld2|| '=''' || val3 ||''') A,
           (SELECT ' || REPLACE(key_list,',','||') || ' AS KEY, 1 val FROM '  ||tbl_dsd || ' B
            WHERE '  ||chk_fld1 || '= '''  ||val2 || ''' and ' || chk_fld2|| '=''' || val3 ||''') B
         WHERE A.KEY=B.KEY(+)
UNION
SELECT ' || num ||' as ID, '|| key_list || ',
         CASE
            WHEN (A.val IS NULL) THEN ''false'' else ''true'' END  AS BOOL_VAR,
         CASE
            WHEN (A.val IS NULL) THEN  ''If '|| val2 || ' is provided  '|| val1 || ' should be
provided too '' END AS ERRORCODE,
         CASE
```

```
    WHEN (A.val IS NULL) THEN ''ERROR'' END AS ERRORLEVEL, sysdate as VAL_DATE
FROM
    (SELECT ' || REPLACE(key_list,',','||') || ' AS KEY, 1 val FROM '  ||tbl_dsd || ' A
    WHERE '  ||chk_fld1 || '= '''  ||val1 || ''' and ' || chk_fld2|| '=''' || val3 ||''') A,
   (SELECT ' || REPLACE(key_list,',','||') || ' AS KEY, B.* FROM '  ||tbl_dsd || ' B
    WHERE '  ||chk_fld1 || '= '''  ||val2 || ''' and ' || chk_fld2|| '=''' || val3 ||''') B
WHERE A.KEY(+)=B.KEY';
```

RTS – Records are all present for Time Series

For this function to work the dataset requires a time_period atribute. Although we fulfilled the requirement and executed the code without errors we were not able to receive any results. We are still consulting VTL experts to discover how and if this needs to be changed.

```
ds:=ANI_gipcat_s_2016;
fill_time_series ( ds, all )
```

RNR – Records' Number is in Range

VTL Rule

```
ds:= ANI_gipcat_s_2016;
ds1:= count( ds group by time_period);
check(ds1>=2 and ds1<=6
errorcode  "Nº of records for each period should be between 2 and 6"
errorlevel "ERROR")
```

SQL Rule

```
  select freq, ref_area, time_period,
  case
  when int_var between 2 and 6 then 'true'
  else 'false'   END  AS "BOOL_VAR",
  CASE
  WHEN NOT (int_var between 2 and 6) THEN 'Nº of records for each period should be between
2 and 6' END AS "ERRORCODE",
    CASE
  WHEN NOT (int_var between 2 and 6) THEN 'ERROR' END AS "ERRORLEVEL"
  from
  (select distinct freq||ref_area||time_period key, freq, ref_area, time_period from
ANI_gipcat_s_2016 a ) a,
 (select freq||ref_area||time_period key, count(1) int_var from ANI_gipcat_s_2016 group by
freq||ref_area||time_period) b
 where a.key=b.key(+)
order by 1,2,3
```

SQL Rule with Parameters

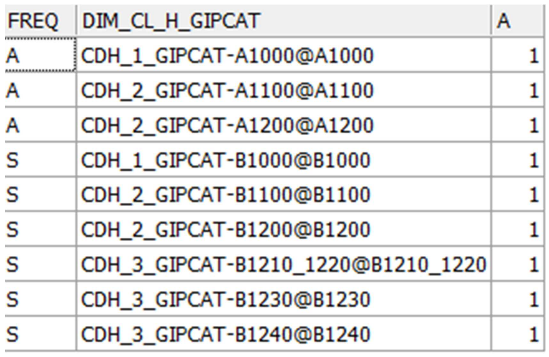
```
Key_list := freq, ref_area, time_period;
tbl_dsd := ANI_gipcat_s_2016;
val1 := 2;
val2 := 6;

SELECT ' || num || ' as ID,'|| key_list || ',
            CASE
                WHEN B.int_var between ' || val1|| ' AND ' || val2 || '  THEN "true" else "false" END
AS BOOL_VAR,
            CASE
                WHEN not(B.int_var between ' || val1|| ' AND ' || val2 || ') THEN "Nº of records for
each period should be between ' || val1|| ' AND ' || val2 || ' " END AS ERRORCODE,
            CASE
                WHEN not(B.int_var between ' || val1|| ' AND ' || val2 || ') THEN "ERROR" END AS
ERRORLEVEL, sysdate as VAL_DATE
            FROM (SELECT DISTINCT ' || REPLACE(key_list,',','||') || ' AS KEY, '|| key_list || ' FROM
'  ||tbl_dsd || ' A) A,
            (SELECT ' || REPLACE(key_list,',','||') || ' as KEY, COUNT(1) int_var FROM  '  ||tbl_dsd
|| '
            GROUP BY ' || REPLACE(key_list,',','||') || ' ) B
            WHERE A.KEY=B.KEY(+)';
```

COC – Codes are Consistent

VTL Rule

```
ds:= ANI_gipcat_s_2016;
comb := count(ds group by freq, dim_cl_h_gipcat);
check (not exists_in (comb, matrix_freq_code,all)
errorcode "Combination of Freq, DIM_CL_H_GIPCAT not
possible"
errorlevel "Error");
```



FREQ	DIM_CL_H_GIPCAT	A
A	CDH_1_GIPCAT-A1000@A1000	1
A	CDH_2_GIPCAT-A1100@A1100	1
A	CDH_2_GIPCAT-A1200@A1200	1
S	CDH_1_GIPCAT-B1000@B1000	1
S	CDH_2_GIPCAT-B1100@B1100	1
S	CDH_2_GIPCAT-B1200@B1200	1
S	CDH_3_GIPCAT-B1210_1220@B1210_1220	1
S	CDH_3_GIPCAT-B1230@B1230	1
S	CDH_3_GIPCAT-B1240@B1240	1

SQL Rule

```
SELECT freq, dim_cl_h_gipcat,
CASE
    WHEN freq||dim_cl_h_gipcat  NOT IN (Select  freq||dim_cl_h_gipcat from matrix_freq_code
b) THEN 'false'
   ELSE 'true' END  AS BOOL_VAR,
CASE
   WHEN freq||dim_cl_h_gipcat  NOT IN (Select freq||dim_cl_h_gipcat from matrix_freq_code)
THEN 'Combination of Freq, DIM_CL_H_GIPCAT not possible' END AS ERRORCODE,
CASE
    WHEN freq||dim_cl_h_gipcat  NOT IN (Select  freq||dim_cl_h_gipcat from matrix_freq_code)
THEN 'ERROR' END AS ERRORLEVEL, sysdate as VAL_DATE
FROM ANI_gipcat_s_2016
ORDER BY 1,2;
```

SQL Rule with Parameters

```
Key_list := freq, dim_cl_h_gipcat;
tbl_dsd := ANI_gipcat_s_2016;
tbl_codes:= matrix_freq_code;
tbl_codes_fld:= freq, dim_cl_h_gipcat;

SELECT ' || num || ' as ID,'|| key_list || ',
            CASE
                WHEN ' || REPLACE(key_list,',','||') || ' NOT IN (Select   ' || REPLACE(,',','||') || '
from ' || tbl_codes || ' b) THEN ''false'' END  AS BOOL_VAR,
            CASE
                WHEN ' || REPLACE(key_list,',','||') || ' NOT IN (Select   ' ||
REPLACE(tbl_codes_fld,',','||') || '  from ' || tbl_codes || ' b) THEN ''Combination of Freq,
DIM_CL_H_GIPCAT not possible '' END AS ERRORCODE,
            CASE
                 WHEN ' || REPLACE(key_list,',','||') || ' NOT IN (Select   ' ||
REPLACE(tbl_codes_fld,',','||') || '  from ' ||tbl_codes || ' b) THEN ''ERROR'' END AS
ERRORLEVEL, sysdate as VAL_DATE
            FROM  '  ||tbl_dsd;
```

VIR – Values are In Range

VTL Rule

```
ds:= ANI_gipcat_s_2016 [filter dim_cl_h_gipcat="CDH_1_GIPCAT-B1000@B1000" and
ref_area="PT"];
check ( ds<130 or  ds>290
errorcode "Values of B1000 for Portugal must be between 130 and 290 "
errorlevel "Warning")
```

SQL Rule

```
SELECT freq, ref_area, dim_cl_h_gipcat, time_period,
CASE
   WHEN obs_value between 130 and 290 THEN 'true'
   else 'false' END  AS "BOOL_VAR",
CASE
  WHEN NOT (obs_value between 130 and 290 ) THEN 'Values of CDH_1_GIPCAT-
B1000@B1000 for PT should be between 130 and 290' END AS "ERRORCODE",
CASE
 WHEN NOT (obs_value between  130 and 290 ) THEN 'ERROR' END AS "ERRORLEVEL"
 FROM ANI_gipcat_s_2016 where dim_cl_h_gipcat = 'CDH_1_GIPCAT-B1000@B1000' and
ref_area = 'PT'
 ORDER BY freq, ref_area, dim_cl_h_gipcat, time_period
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, dim_cl_h_gipcat, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1:= obs_value;
chk_fld2:= dim_cl_h_gipcat;
chk_fld3:= ref_area;
val1:= 130;
val2:= 290;
val3:= PT;
VAL_LIST:= CDH_1_GIPCAT-B1000@B1000;
```

```
SELECT ' || num || ' as ID,'|| key_list || ',
            CASE
                WHEN ' || chk_fld1|| ' BETWEEN ' || val1 || ' and ' ||  val2|| ' THEN ''true'' ELSE
''false'' END  AS BOOL_VAR,
            CASE
                WHEN NOT(' || chk_fld1|| ' BETWEEN ' || val1 || ' and ' ||  val2|| ') THEN ''Values of
'  || VAL_LIST ||  ' for '   || val3 ||   ' should be between '   || val1 ||   ' and '   || val2 ||  ''' END
AS ERRORCODE,
            CASE
                                 WHEN NOT(' || chk_fld1|| ' BETWEEN ' || val1 || ' and ' ||  val2|| ')
THEN ''ERROR'' END AS ERRORLEVEL, sysdate as VAL_DATE
```

```
          FROM  '  ||tbl_dsd || ' WHERE '  ||chk_fld2 || ' = '''  ||VAL_LIST || ''' AND '  ||chk_fld3
|| ' = '''  ||val3 || '''';
```

VCO – Values are Consistent

VTL Rule

```
ds_t:= ANI_gipcat_s_2016 [sub dim_cl_h_gipcat="CDH_1_GIPCAT-B1000@B1000"];
ds_t1:= ANI_gipcat_s_2016 [sub dim_cl_h_gipcat="CDH_2_GIPCAT-B1200@B1200"];
check ( ds_t1<ds_t*0.7
errorcode "Values of B1200 should not be less than 70% of total"
errorlevel "Warning")
```

SQL Rule

```
SELECT A.freq, A.ref_area, A.time_period,
CASE
WHEN (A.obs_value<0.7*B.obs_value) THEN 'false'
ELSE 'true' END  AS "BOOL_VAR",
CASE
WHEN (A.obs_value<0.7*B.obs_value) THEN 'Values of B1200 should not be less than 70% of
total'
END "ERRORCODE",
CASE
WHEN (A.obs_value<0.7*B.obs_value) THEN 'Warning'
END "ERRORLEVEL", sysdate as VAL_DATE
FROM
(SELECT freq||ref_area||time_period key, A.*  FROM valdata.ANI_gipcat_s_2016 a WHERE
dim_cl_h_gipcat='CDH_1_GIPCAT-B1000@B1000') A,
(SELECT freq||ref_area||time_period key, obs_value FROM valdata.ANI_gipcat_s_2016 WHERE
dim_cl_h_gipcat='CDH_2_GIPCAT-B1200@B1200') B
WHERE (a.key=B.key);
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1:= obs_value;
chk_fld2:= dim_cl_h_gipcat;
val1:= 70;
val2:= CDH_1_GIPCAT-B1000@B1000;
val3:= CDH_2_GIPCAT-B2000@B2000;
```

```
SELECT ' || num ||' as ID, '|| key_list || ',
            CASE
                WHEN A.'|| chk_fld1|| ' < '|| to_number(val1) || '*B.'|| chk_fld1|| '/100 THEN
''false'' else ''true'' END  AS BOOL_VAR,
                        CASE
                WHEN A.'|| chk_fld1|| ' < '|| to_number(val1) || '*B.'|| chk_fld1|| '/100 THEN
''Values of '|| val3 || ' should not be less then '|| to_number(val1) || '% of ' || val2 ||'.'' END AS
ERRORCODE,
            CASE
```

```
                WHEN A.'|| chk_fld1|| ' < '|| to_number(val1) || '*B.'|| chk_fld1|| '/100 THEN
''ERROR'' END AS ERRORLEVEL, sysdate as VAL_DATE
            FROM
                (SELECT ' || REPLACE(key_list,',','||') || ' AS KEY, A.* FROM '  ||tbl_dsd || ' A
WHERE '  ||chk_fld2 || '= '''  ||val2 || ''') A,
                (SELECT ' || REPLACE(key_list,',','||') || ' as KEY,'|| chk_fld1|| ' FROM  '  ||tbl_dsd ||
' B WHERE '  ||chk_fld2 || '= '''  ||val3 || ''') B
            WHERE A.KEY=B.KEY(+)';
```

VAD – Values for Aggregates are consistent with Details

VTL Rule

```
ds:= ANI_gipcat_s_2016;
tot:= sum(ds [ filter dim_cl_h_gipcat = "CDH_1_GIPCAT-B1000@B1000"] group by
time_period,freq,ref_area) ;
detail:= sum(ds [ filter dim_cl_h_gipcat in {"CDH_2_GIPCAT-B1100@B1100", "CDH_2_GIPCAT-
B1200@B1200"}] group by time_period,freq,ref_area);
check ( tot>detail-2 and tot<detail+2
errorcode "Total bovines should be equal to 'Calves' and young cattle plus 'Adult cattle' more
or less 1"
errorlevel "ERROR");
```

SQL Rule

```
SELECT freq,ref_area,time_period ,
CASE
WHEN (A.v < b.v+2 and A.v > b.v-2) THEN 'true'
ELSE 'false' END  AS "BOOL_VAR",
CASE
WHEN not(A.v < b.v+2 and A.v > b.v-2) THEN 'B1000 should be equal to B1100 and B1200
more or less 1'
END "ERRORCODE",
CASE
WHEN not (A.v < b.v+2 and A.v > b.v-2) THEN 'Warning'
END "ERRORLEVEL", sysdate as VAL_DATE
FROM
(SELECT freq||ref_area||time_period key, sum(obs_value)v, freq,ref_area,time_period FROM
valdata.ANI_gipcat_s_2016 a WHERE dim_cl_h_gipcat='CDH_1_GIPCAT-B1000@B1000' group
by freq||ref_area||time_period, freq,ref_area,time_period) A,
(SELECT freq||ref_area||time_period key, sum(obs_value) v FROM valdata.ANI_gipcat_s_2016
WHERE dim_cl_h_gipcat in ('CDH_2_GIPCAT-B1100@B1100','CDH_2_GIPCAT-B1200@B1200')
group by freq||ref_area||time_period) B
WHERE (a.key=B.key);
```

SQL Rule with Parameters

```
Key_list := freq, ref_area, time_period;
tbl_dsd := ANI_gipcat_s_2016;
chk_fld1:= obs_value;
chk_fld2:= dim_cl_h_gipcat;
val1:= 2;
val2:= CDH_1_GIPCAT-B1000@B1000;
val_list:= 'CDH_2_GIPCAT-B1100@B1100',' CDH_2_GIPCAT-B1200@B1200';

SELECT  ' || num ||' as ID, '|| key_list || ',
          CASE
             WHEN (A.ov < b.ov + '|| val1|| ' and a.ov > b.ov - '|| val1 || ') THEN ''true'' else
''false'' END  AS BOOL_VAR,
                    CASE
```

```
               WHEN NOT (A.ov < b.ov + '|| val1|| ' and a.ov > b.ov - '|| val1 || ') THEN ''Value of
'|| val2 || ' should be equal to '|| replace(val_List,'''','') || ' more or less 1 '' END AS
ERRORCODE,
          CASE
               WHEN NOT (A.ov < b.ov + '|| val1|| ' and a.ov > b.ov - '|| val1 || ') THEN ''ERROR''
END AS ERRORLEVEL, sysdate as VAL_DATE
          FROM
               (SELECT ' || REPLACE(key_list,',','||') || ' AS KEY, sum('|| chk_fld1|| ') as ov, '||
key_list||
               ' FROM '  ||tbl_dsd || ' A WHERE '  ||chk_fld2 || '= '''  ||val2 || ''' group by ' ||
REPLACE(key_list,',','||') || ' , '|| key_list|| ') A,
               (SELECT ' || REPLACE(key_list,',','||') || ' as KEY, sum(' || chk_fld1|| ') as ov
                 FROM  ' ||tbl_dsd || ' B WHERE '  ||chk_fld2 || ' in ( '  ||val_List || ') group by ' ||
REPLACE(key_list,',','||') || ') B
          WHERE A.KEY=B.KEY(+)';
```

VNO – Values are Not Outliers in Time Series

Not applicable