# Input Data

The initial data for the forecast flag definition are listed below. These tables should be present in the system before this step for example in STG or in DDS area.

• IN\_PRODUCT (see data requirements)

• IN\_LOCATION (see data requirements)

## HYBRID\_FORECAST\_<target\_var>

Forecast values from VF forecast step that is disaggregated to sku/location level and disaccumulated to daily granularity.

|  |  |
| --- | --- |
| **HYBRID\_FORECAST** | |
| Column Name | Description |
| **product\_lvl\_id<m>** | ID of product group for which specific ML model will be trained, m = **ml\_product\_lvl** |
| **location\_lvl\_id<n>** | ID of location group for which specific ML model will be trained, n = **ml\_location\_lvl** |
| **customer\_lvl\_id<l>** | ID of customer group for which specific ML model will be trained, l = **ml\_customer\_lvl** |
| **distr\_channel\_lvl\_id<k>** | ID of distr\_channel group for which specific ML model will be trained, k = **ml\_distr\_channel\_lvl** |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **PERIOD\_END\_DT** | End date of the period |
| **VF\_FORECAST\_VALUE** | VF forecast value |
| **SEGMENT\_NAME** | Name of segment that was linked to a pair product/location within VF Project (can be missing) |
| **DEMAND\_TYPE** | ‘promo’ or ‘regular’ – type ML model is used to forecast |
| **ASSORTMENT\_TYPE** | new, or old |
| **ML\_FORECAST\_VALUE** | ML forecasted value |
| **HYBRID\_FORECAST\_VALUE** | Hybrid Forecast Value |
| **ENSEMBLE\_FORECAST\_VALUE** | Ensemble forecast value |
| **FORECAST\_SOURCE** | Forecast source |

## PRODUCT LIFE-CYCLE

|  |  |
| --- | --- |
| **PRODUCT LIFE-CYCLE** | |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID |
| **LOCATION\_LVL\_ID** | Location hierarchy element ID |
| **CUSTOMER\_LVL\_ID** | Customer hierarchy element ID |
| **DISTR\_CHANNEL\_LVL\_ID** | Distribution CHANNEL hierarchy element ID |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **PRODUCT\_SUCCESSOR\_ID** | Successor Product ID |
| **RELATION\_SHARE** | Number |
| **PERIOD\_TYPE** | Active', 'Blocked', 'End-of-life' |

## LOCATION LIFE-CYCLE

|  |  |
| --- | --- |
| **LOCATION LIFE-CYCLE** | |
| Column Name | Description |
| **LOCATION\_ID** | Location ID |
| **CUSTOMER\_LVL\_ID** | Customer hierarchy element ID |
| **PRODUCT\_LVL\_ID** | Product hierarchy element ID |
| **DISTR\_CHANNEL\_LVL\_ID** | Distribution CHANNEL hierarchy element ID |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **LOCATION\_SUCCESSOR\_ID** | Successor Location ID |
| **RELATION\_SHARE** | Number |
| **PERIOD\_TYPE** | Closure reason: 'reconstruction', 're-branding' etc. |

## CUSTOMER LIFE-CYCLE

|  |  |
| --- | --- |
| **CUSTOMER LIFE-CYCLE** | |
| Column Name | Description |
| **CUSTOMER\_ID** | Customer ID |
| **LOCATION\_LVL\_ID** | Location hierarchy element ID |
| **PRODUCT\_LVL\_ID** | Product hierarchy element ID |
| **DISTR\_CHANNEL\_LVL\_ID** | Distribution CHANNEL hierarchy element ID |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **CUSTOMER\_SUCCESSOR\_ID** | Successor Customer ID |
| **RELATION\_SHARE** | Number |
| **PERIOD\_TYPE** | active, blocked, end-of-life |

## FORECAST FLAG

Product/location lifecycle history containing the following fields is used as an input:

|  |  |
| --- | --- |
| **DM.FORECAST\_FLAG** | |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSOMER\_ID** | Customer id |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **STATUS** | One of the following status: active, blocked, out-of-sale |

## RESTORED\_DEMAND\_<target variable type>

Demand information regarding the past till last known day of the history.

|  |  |
| --- | --- |
| **DM.RESTORED\_DEMAND** | |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSOMER\_ID** | Customer id |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_DT** | Date of sales (calendar day) |
| **SALES\_QTY\_R** | Restored demand |
| **PROMO\_FLG** | 1|promo event was active  0|no promo event |
| **SALES\_QTY** | Total sales in units per day (w/o returns) |
| **STOCK\_QTY** | Stock qty (BOP) |
| **DEFICIT\_FLG1** | 1|primary deficit occurred  0|no primary deficit |
| **DEFICIT\_FLG2** | 1|secondary deficit occurred  0|no secondary deficit |

## CONFIG\_PARAMETERS

The following config parameters are used within the step.

### TGT\_VAR\_CONFIG

Csv file

|  |  |  |
| --- | --- | --- |
| **TGT\_VAR\_CONFIG.csv** | | |
| **Column Name** | **Description** | **Example** |
| **tgt\_type** | One of 3 types of the target variable:   * SELLIN – means CPG sales to its customer, * SELLOUT – means CPG’s customer sales to their clients, * POS – means sales in the point of sales, can be relevant for both Retailer and CPG | POS |
| **tgt\_qty\_table** | Name of the table which contains quantity information, one of 3 variants is possibe: IN\_SELL\_IN, IN\_SELL\_OUT, IN\_SALES | IN\_SALES |
| **value\_src** | Name of the target variable from the source table. It should be quantity of sales.  Feasible values: INVOICE\_QTY, SALES\_QTY, SHIPMENT\_QTY, ORDER\_QTY. | SALES\_QTY |
| **act\_flag** | Activity flag, whether this target variable is needed to be forecasted. Feasible values: 0 or 1 | 1 |
| **dr\_scen** | Demand restoration scenario for target variable:   * Scenario 0 (parameter = 0). Demand restoration isn’t needed. * Scenario 1 (parameter = 1). Retail scenario to a greater extent. Demand restoration is performed based on stock data. Demand extending/prolongation for short seasonal products isn’t performed. * Scenario 2 (parameter = 2). CPG scenario to a greater extent. Only demand extending/prolongation for short seasonal products is performed. * Scenario 3 (parameter = 3). Scenario 1 and Scenario 2 are performed sequentially. | 3 |
| **link\_with\_stock** | Flag, whether this target variable is linked with provided stock data. Feasible values: 0 or 1 | 1 |
| **link\_with\_promo** | Flag, whether this target variable is linked with provided promo data. Feasible values: 0 or 1 | 0 |
| **link\_with\_price** | Flag, whether this target variable is linked with provided price data. Feasible values: 0 or 1 | 1 |
| **vf\_product\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 8, which means PRODUCT\_ID | 7 |
| **vf\_location\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 6, which means LOCATION\_ID | 1 |
| **vf\_customer\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 6, which means CUSTOMER\_ID | 5 |
| **vf\_distr\_channel\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 3, which means DISTR\_CHANNEL\_ID | 1 |
| **vf\_time\_lvl** | Accumulation level for ML ABT by time hierarchy, default value is WEEK.2, which means weeks began from Monday | MONTH |
| **ml\_product\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 8, which means PRODUCT\_ID | 7 |
| **ml\_location\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 6, which means LOCATION\_ID | 5 |
| **ml\_customer\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 6, which means CUSTOMER\_ID | 4 |
| **ml\_distr\_channel\_lvl** | Aggregation level for ML ABT by product hierarchy, default value is 3, which means DISTR\_CHANNEL\_ID | 1 |
| **ml\_time\_lvl** | Accumulation level for ML ABT by time hierarchy, default value is WEEK.2, which means weeks began from Monday | WEEK.2 |
| **dag\_product\_lvl** | Aggregation level, that is used at disaggregation step forecast values should be the same at dag before and after disaggregation step if there is at least one active sku  **dag\_product\_lvl<= min(ml\_\*\*\_lvl, vf\_\*\*\_lvl)** |  |
| **dag\_location\_lvl** | Aggregation level, that is used at disaggregation step forecast values should be the same at dag before and after disaggregation step if there is at least one active sku  **dag\_product\_lvl<= min(ml\_\*\*\_lvl, vf\_\*\*\_lvl)** |  |
| **dag\_customer\_lvl** | Aggregation level, that is used at disaggregation step forecast values should be the same at dag before and after disaggregation step if there is at least one active sku  **dag\_product\_lvl<= min(ml\_\*\*\_lvl, vf\_\*\*\_lvl)** |  |
| **dag\_distr\_channel\_lvl** | Aggregation level, that is used at disaggregation step forecast values should be the same at dag before and after disaggregation step if there is at least one active sku  **dag\_product\_lvl<= min(ml\_\*\*\_lvl, vf\_\*\*\_lvl)** |  |
| **out\_product\_lvl** | Aggregation level for output file by product hierarchy, default value is 8, which means PRODUCT\_ID | 8 |
| **out\_location\_lvl** | Aggregation level for output file by product hierarchy, default value is 6, which means LOCATION\_ID |  |
| **out\_customer\_lvl** | Aggregation level for output file by product hierarchy, default value is 6, which means CUSTOMER\_ID |  |
| **out\_distr\_channel\_lvl** | Aggregation level for output file by product hierarchy, default value is 3, which means DISTR\_CHANNEL\_ID |  |
| **out\_time\_lvl** | Accumulation level for ML ABT by time hierarchy, default value is WEEK.2, which means weeks began from Monday | SPLIT-WEEK |

### CONFIG

|  |  |
| --- | --- |
| **INITIAL\_GLOBAL parameters init** | |
| Column Name | Description |
| **IB\_FCTS\_HORIZON** | Forecast Horizon |
| **mpDepHistTimePeriod** | Depth of the history to calculate share of id-level element |

## INITIAL\_GLOBAL parameters

All parameters are listed in initial\_global file.

|  |  |
| --- | --- |
| **INITIAL\_GLOBAL parameters init** | |
| Column Name | Description |
| **IB\_HIST\_END\_DT** | Last known date (i.e. sales and stock information is known) |
| **IB\_FC\_AGG\_END\_DT** | The last date of the Forecast Horizon IB\_HIST\_END\_DT+IB\_FCST\_HORIZ |

## Other Dependencies

# Algorithm Definition

## Common Utility Steps

1. To aggregate a table on al\_product\_lvl/al\_location\_lvl/al\_customer\_lvl/al\_distr\_channel\_lvl level:
   1. Table left join PRODUCT on product\_lvl\_id<m>

left join LOCATION on location\_lvl\_id<n>

left join CUSTOMER on Customer\_lvl\_id<k>

left join DISTR\_CHANNEL on distr\_channel\_lvl\_id<l>

and add columns:

product\_lvl\_id<al\_product\_lvl>,

location\_lvl\_id<al\_location\_lvl>,

customer\_lvl\_id<al\_customer\_lvl>,

distr\_channel\_lvl\_id<al\_distr\_channel\_lvl>

e.g. if al\_product\_lvl = 8 then add column product\_id

and delete previous columns product\_lvl\_id<m>, location\_lvl\_id<n>, Customer\_lvl\_id<k>, distr\_channel\_lvl\_id<l>

* 1. Provide aggregation (aggregation method for other columns will be provided separately) table Group by

product\_lvl\_id<al\_product\_lvl>,

location\_lvl\_id<al\_location\_lvl>,

customer\_lvl\_id<al\_customer\_lvl>,

distr\_channel\_lvl\_id<al\_distr\_channel\_lvl>

1. To aggregate a table on al\_time\_lvl:
   1. transform time variable using intnx-function (e.g intnx('week.2', period\_dt, 6));
   2. Provide aggregation (aggregation method for other columns will be provided seprarately) table Group by transformed period\_dt

## Calculate Forecast Share at Final Output Granularity Level

**Inputs:** FORECAST\_FLAG, DEMAND\_RESTORATION\_< tgt variable type, HYBRID\_FORECAST\_<tgt variable type>

**Transformation algorithm:**

1. Select only quadruples product\_id, location\_id, customer\_id, distr\_channel\_id that are active according to Forecast Flag
   1. SELECT \* FROM FORECAST\_FLAG

WHERE PERIOD\_START\_DT <= **IB\_FC\_AGG\_END\_DT**

AND

PERIOD\_END\_DT >= **IB\_HIST\_END\_DT+1**

AND STATUS IS ACTIVE

1. Calculate average tgt\_qty value based on RESTORED\_DEMAND\_<**target\_type**>

/\*Calculate min start\_dt and max end\_dt on aggregated level\*/

SELECT product\_id, location\_id, customer\_id, distr\_channel\_id,

PROMO\_FLG,

BASE = AVERAGE (**TGT\_QTY)**

FROM RESTORED\_DEMAND\_**target\_type**

WHERE DEFICIT\_FLG1 = 0 AND DEFICIT\_FLG2 = 0 and

intnx('day',&IB\_HIST\_END\_DT.,-&mpDepHistTimePeriod.,'b'))

GROUP BY product\_id, location\_id, customer\_id, distr\_channel\_id, PROMO\_FLG

* 1. Join Demand Restored and FF information

2 full join 1 on product\_id, location\_id, customer\_id, distr\_channel\_id

Note: PERIOD\_START\_DT, PERIOD\_END\_DT is missing if quadruple is out-of-date,

Note: AVG\_TGT\_QTY is missing if there is no historical values in RESTORED\_DEMAND.

1. Add out\_lvls and dagl\_lvls to 3:
   1. 3 left join PRODUCT (LOCATION, CUSTOMER, DISTR\_CHANNEL)
   2. Add out\_product\_lvl (out\_location\_lvl, out\_customer\_lvl, out\_distr\_channel\_lvl)
   3. Add dag\_product\_lvl (dag\_location\_lvl, dag\_customer\_lvl, dag\_distr\_channel\_lvl)
2. Calculate MEAN\_BASE\_OUT\_LVL and MEAN\_BASE\_dag\_LVL as:
   1. AVG(BASE) as MEAN\_BASE\_OUT\_LVL, out\_lvls, PROMO\_FLG   
      from 4 group by out\_lvls, PROMO\_FLG
   2. AVG(BASE) as MEAN\_BASE\_dag\_LVL, dag\_lvls, PROMO\_FLG   
      from 4 group by dag\_lvls, PROMO\_FLG
3. product\_id, location\_id, customer\_id, distr\_channel\_id, period\_start\_dt, period\_end\_dt, PROMO\_FLG,  
   BASE=COALESCE(4.BASE, 5a.MEAN\_BASE\_OUT\_LVL, 5b.MEAN\_BASE\_dag\_LVL, 1) ,   
   5a.MEAN\_BASE\_OUT\_LVL,   
   5b.MEAN\_BASE\_dag\_LVL

from 4  
 left join 5a on out\_lvls, PROMO\_FLG   
 left join 5b on dag\_lvls, PROMO\_FLG

1. Add missing promo flags that are present
   1. SELECT \* except PROMO\_FLG, BASE

1-PROMO\_FLG AS PROMO\_FLG

Missing as BASE

FROM 6

Assumption: In case when ml-lvl element has a new promo\_flg value in the future all correspondent out-lvl elements has the same promo\_flg values in the same period.

* 1. SELECT product\_id, location\_id, customer\_id, distr\_channel\_id, PROMO\_FLG, period\_start\_dt, period\_end\_dt,

COALESCE (MIN(BASE), 0)) as BASE

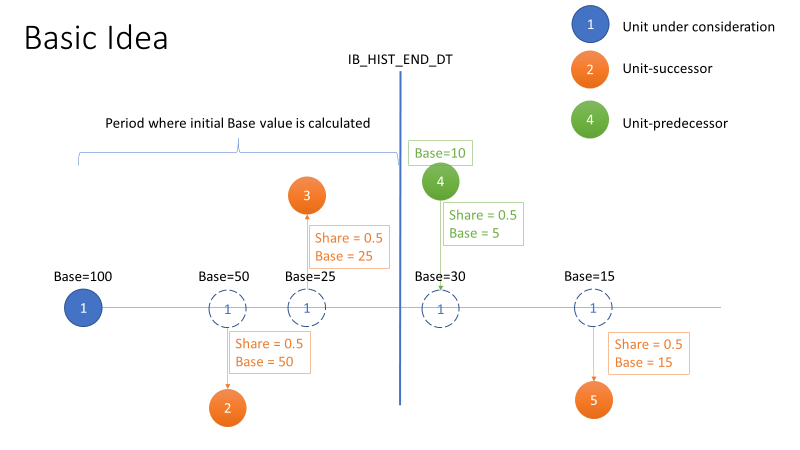
FROM 6. UNION 7.a

GROUP BY product\_id, location\_id, customer\_id, distr\_channel\_id, PROMO\_FLG, period\_start\_dt, period\_end\_dt

Output: As a result of this step, a table of the following structure is constructed, *T1.*

|  |  |
| --- | --- |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSOMER\_ID** | Customer id |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **PROMO\_FLG** | 1|promo event was active  0|no promo event |
| **BASE** | Average tgt variable value |

## Considering Life Cycles information



**Idea of the step:** calculate transformation matrix for each time moment whenthere is a base volume change according to lifecycle information

ASSUMPTION: if there are several transfers within a day: from product1 to product 2 and from location 1 to location 2, then the correct order of these transformations is:

* + - * 1. distr\_channel dimension transfers (from distr\_channel 1 to distr\_channel 2);
        2. customer dimension transfers (from customer 1 to customer 2);
        3. location dimension transfers (from location 1 to location 2);
        4. product dimension transfers (from product 1 to product 2).

**Inputs:** T1

**Transformation algorithm:**

1. Expand all lifecycle matrix to PRODUCT\_ID/LOCATION\_ID/CUSTOMER\_ID/DISTR\_CHANNEL\_ID level, use only those rows where PERIOD\_START\_DT>**IB\_HIST\_END\_DT.**
2. Union all lifecycle tables

After this step the following table should be present

|  |  |
| --- | --- |
| Column Name | Description |
| **PRODUCT\_ID\_FROM** | Product ID predecessor |
| **LOCATION\_ID\_FROM** | Location ID predecessor |
| **CUSOMER\_ID\_FROM** | Customer id predecessor |
| **DISTR\_CHANNEL\_ID\_FROM** | Distribution Channel ID predecessor |
| **PRODUCT\_ID\_TO** | Product ID successor |
| **LOCATION\_ID\_TO** | Location ID successor |
| **CUSOMER\_ID\_TO** | Customer id successor |
| **DISTR\_CHANNEL\_ID\_TO** | Distribution Channel ID successor |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **SOURCE** | 1. Product, 2- Location, 3- Customer, 4 – Distr-Channel LifeCycle |
| **SHARE** | relation share |

1. Select time moments when any transformation take place from 2:

Union (SELECT DISTINCT PERIOD\_START\_DT as TR\_DT) and (SELECT DISTINCT PERIOD\_END\_DT as TR\_DT) FROM 2

1. Copy T1 to temp\_t1 and add column rows\_to\_transform\_flg
   1. SELECT \* apart from period\_start\_dt, period\_end\_dt

, COALESCE(period\_start\_dt, IB\_HIST\_END\_DT) as period\_start\_dt

, COALESCE(period\_end\_dt, IB\_HIST\_END\_DT) as period\_end\_dt

, rows\_to\_transform\_flg = 0

FROM T1

Assumption: there is no transformations related with out-of-date products

* 1. Replace missing period\_start\_dt and period\_end\_dt to IB\_HIST\_END\_DT

1. Order dates from 3 and calculate transformation matrix and updated base volumes for each timestamp , for each time moment from 3:
   1. Update Base values in temp\_t1 based on ended lifecycle for source : (Consider subsequently DISTR\_CHANNEL lifecycle, Consider CUSTOMER lifecycle, Consider LOCATION lifecycle, Consider PRODUCT lifecycle. i.e. = 4, 3, 2, 1):
      1. Select rows from 2 where PERIOD\_END\_DT = and SOURCE =
      2. Mark all rows in temp\_t1 that must be changed due to lifecycle changes at the moment :

Select temp\_t1.\*,

rows\_to\_transform\_flg = CASE WHEN i.PERIOD\_START\_DT is not missing THEN 1 ELSE 0 END

i.PERIOD\_START\_DT as TRANS\_START\_DT

FROM temp\_t1

LEFT JOIN i. on

PRODUCT\_ID = i.PRODUCT\_ID\_FROM,

LOCATION\_ID = i.LOCATION\_ID\_FROM,

CUSTOMER\_ID = i.CUSTOMER\_ID\_FROM,

DISTR\_CHANNEL\_ID = i.DISTR\_CHANNEL\_ID\_FROM,

i.PERIOD\_END\_DT BETWEEN temp\_t1.PERIOD\_START\_DT and temp\_t1.PERIOD\_END\_DT

* + 1. Prepare correct base value since

/\*select date of the earliest transformation that is ended at the moment \*/

SELECT MIN (TRANS\_START\_DT) as TRANS\_START\_DT, PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID

FROM ii.

WHERE rows\_to\_transform\_flg = 1

GROUP BY PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID

/\*put the base value that was before ended transformation as a base value after the moment \*/

SELECT PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID, PROMO\_FLG

as PERIOD\_START\_DT,

MAX(PERIOD\_END\_DT) as PERIOD\_END\_DT, /\*fake maximum\*/

COALESCE (MAX(

IFN(iii.1.TRANS\_START\_DT is not missing; ii.BASE; missing)

) /\*fake maximum\*/

, max(T1.BASE ) /\*fake maximum\*/

) As BASE

0 as rows\_to\_transform\_flg

FROM ii.

WHERE rows\_to\_transform\_flg = 1

LEFT JOIN iii.1 ON PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID and

intnx(‘day’, TRANS\_START\_DT, -1) BETWEEN PERIOD\_START\_DT and PERIOD\_END\_DT

LEFT JOIN T1 ON PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID, PROMO\_FLG

GROUP BY PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID, PROMO\_FLG

* + 1. Update base value period\_end\_dt for all related to transformation rows:

Select PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID, PROMO\_FLG,

MAX(PERIOD\_START\_DT) as PERIOD\_START\_DT /\*fake maximum\*/,

as PERIOD\_END\_DT

0 as rows\_to\_transform\_flg

FROM ii

WHERE rows\_to\_transform\_flg = 1

GROUP BY PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID, PROMO\_FLG

* + 1. Update temp\_T1:

SELECT\*FROM ii WHERE rows\_to\_transform\_flg = 0

UNION SELECT \* FROM iii

UNION SELECT \* FROM iv

* 1. Update base values in temp\_t1 based on started lifecycle for source : (Consider subsequently Consider PRODUCT lifecycle, Consider LOCATION lifecycle, Consider CUSTOMER lifecycle, DISTR\_CHANNEL lifecycle, i.e. = 1, 2, 3, 4):
     1. Select rows from 2 where PERIOD\_START\_DT = and SOURCE =
     2. add rows to have total share for each quadruple\_from (i.e. PRODUCT\_ID\_FROM, LOCATION\_ID\_FROM, CUSTOMER\_ID\_FROM and DISR\_CHANNEL\_ID\_FROM) equal to 1, it can be done as follows
        1. /\*add rows from from\_list to to\_list\*/

i UNION

SELECT PRODUCT\_ID\_FROM, LOCATION\_ID\_FROM, CUSTOMER\_ID\_FROM, DISTR\_CHANNEL\_ID\_FROM

PRODUCT\_ID\_TO = PRODUCT\_ID\_FROM,

LOCATION\_ID\_TO = LOCATION\_ID\_FROM,

CUSTOMER\_ID\_TO = CUSTOMER\_ID\_FROM,

DISTR\_CHANNEL\_ID\_TO = DISTR\_CHANNEL\_ID\_FROM,

MAX(0, 1 – SUM(SHARE)) as SHARE

PERIOD\_START\_DT,

missing as PERIOD\_END\_DT

as SOURCE

FROM i

GROUP BY PERIOD\_START\_DT, PRODUCT\_ID\_FROM, LOCATION\_ID\_FROM, CUSTOMER\_ID\_FROM, DISTR\_CHANNEL\_ID\_FROM

* + - 1. /\*add absent rows from to\_list to from\_list\*/

ii.1 UNION

(SELECT **\*\*\_ID\_TO,**

**to\_list.\*\*\_ID\_TO as \*\*\_ID\_FROM**

1 as SHARE,

, as PERIOD\_START\_DT

, missing as PERIOD\_END\_DT

, as Source

FROM ii.1

RIGHT JOIN (SELECT DISTINCT**\*\*\_ID\_TO** DISTINCT FROM ii.1) as to\_list

ON to\_list.\*\*\_ID\_TO = ii.1.\*\*\_ID\_FROM

WHERE ii.1.PRODUCT\_IF\_FROM is missing

) as ext\_from\_list

* + 1. Mark all rows in temp\_t1 that must be changed due to lifecycle changes at the moment :

Select temp\_t1.\*,

ii.\*\*\_ID\_TO, ii.Share

,rows\_to\_transform\_flg = CASE WHEN ii.PERIOD\_START\_DT is not missing THEN 1 ELSE 0 END

FROM temp\_t1

LEFT JOIN ii. on

PRODUCT\_ID = ii.PRODUCT\_ID\_FROM,

LOCATION\_ID = ii.LOCATION\_ID\_FROM,

CUSTOMER\_ID = ii.CUSTOMER\_ID\_FROM,

DISTR\_CHANNEL\_ID = ii.DISTR\_CHANNEL\_ID\_FROM,

ii.PERIOD\_START\_DT BETWEEN temp\_t1.PERIOD\_START\_DT and temp\_t1.PERIOD\_END\_DT

* + 1. Prepare correct base value before

/\*put as the last valid moment of the previous base value \*/

SELECT PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID, PROMO\_FLG

MAX(PERIOD\_START\_DT) as PERIOD\_START\_DT, /\*fake maximum\*/

Intnx(‘day’, , -1) as PERIOD\_END\_DT,

max(T1.BASE ) as BASE, /\*fake maximum\*/

0 as rows\_to\_transform\_flg

FROM iii.

WHERE rows\_to\_transform\_flg = 1

GROUP BY PRODUCT\_ID, LOCATION\_ID, CUSTOMER\_ID, DISTR\_CHANNEL\_ID, PROMO\_FLG

* + 1. Prepare correct base value since :

SELECT

**\*\*\_ID\_TO** as **\*\*\_ID**, PROMO\_FLG

as PERIOD\_START\_DT,

MAX(PERIOD\_END\_DT) as PERIOD\_END\_DT, /\*fake maximum\*/

sum(BASE\*SHARE ) as BASE,

0 as rows\_to\_transform\_flg

FROM iii.

WHERE rows\_to\_transform\_flg = 1

GROUP BY PRODUCT\_ID\_TO, LOCATION\_ID\_TO, CUSTOMER\_ID\_TO, DISTR\_CHANNEL\_ID\_TO, PROMO\_FLG

* + 1. Update temp\_T1:

SELECT \*\*\_ID, period\_start\_dt, period\_end\_dt, base, PROMO\_FLG

FROM iii WHERE rows\_to\_transform\_flg = 0

UNION SELECT \* FROM iv

UNION SELECT \* FROM v

ASSUMPTION: if there is several transformations regarding particular quadruple product/location/customer/distr\_channel then it is provided in the input data at the very detailed level.

Output: As a result of this step, a table of the following structure is constructed, *T2.*

|  |  |
| --- | --- |
| Column Name | Description |
| **PRODUCT\_ID** | Product ID (the lowest level of the product hierarchy) |
| **LOCATION\_ID** | Location ID |
| **CUSOMER\_ID** | Customer id |
| **DISTR\_CHANNEL\_ID** | Distribution Channel ID |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **PROMO\_FLG** | 1|promo event was active  0|no promo event |
| **BASE** | Average tgt variable value |

## Calculate Forecast Share

**Inputs:** T2, dictionaries

**Transformation algorithm:**

1. Add out\_lvls to T2:
   1. T2 left join PRODUCT (LOCATION, CUSTOMER, DISTR\_CHANNEL)
   2. Add out\_product\_lvl (out\_location\_lvl, out\_customer\_lvl, out\_distr\_channel\_lvl)
   3. Adddagl\_product\_lvl (dag\_location\_lvl, dag\_customer\_lvl, dag\_distr\_channel\_lvl)
2. Calculate periods of constant base values within each element of dag\_\*\*\_lvl level

i.e. create list of not-overlapping intervlas period\_start\_dt , period\_end\_dt within each quadruple dag\_product\_lvl, dag\_location\_lvl, dag\_customer\_lvl, dag\_distr\_channel\_lvl from 1

i.e. within each quadruple of dag level product, location, customer, distr\_channel

* 1. Select all quadruples of out-level that are related to a quadruple of ml level (see figure Figure 1: base value behavior within all quadruples of id-level)

Figure 1: base value behavior within all quadruples of id-level

* 1. Add rows to cover all intervals from min(period\_start\_dt) to max(period\_end\_dt) for each quadruple of out-level (see figure below )



* 1. Order all period\_start\_dt, select distinct dates and add column row\_number
  2. Order all period\_end\_dt, select distinct dates and add column row\_number
  3. Join c. and d on row\_number, final result will contain intervals without changes in all quadruples of id-level
  4. Put detailed periods to T2 table:

SELECT 1.\* (apart from period\_start\_dt, period\_end\_dt),

, max(1.period\_start\_dt, b.period\_start\_dt) as period\_start\_dt

, min(1.period\_end\_dt, b.period\_end\_dt) as period\_end\_dt

FROM 1

left join e. on \*\*\_dag\_lvl and (1.period\_start\_dt <= b.period\_end\_dt and 1.period\_end\_dt >= b.period\_start\_dt)

Note: last join is aimed to merge all intersecting intervals from both tables (see figure below):

1. Calculate base value of each product\_id, location\_id, customer\_id, distr\_channel\_id within

product\_lvl\_id<out\_product\_lvl>, location\_lvl\_id<out\_location\_lvl>, customer\_lvl\_id<out\_customer\_lvl>, distr\_channel\_lvl\_id<out\_distr\_channel\_lvl>

SELECT 2.**\*\*\_out\_lvl**, **\*\*\_dag\_lvl**, promo\_flg,

COALESCE( SUM(BASE), 0) as BASE\_out\_lvl,

COALECSE(COUNT(), 0) as cnt\_out\_lvl,

period\_start\_dt, period\_end\_dt

FROM 2.

GROUP BY product\_lvl\_id<out\_product\_lvl>, location\_lvl\_id<out\_location\_lvl>, customer\_lvl\_id<out\_customer\_lvl>, distr\_channel\_lvl\_id<out\_distr\_channel\_lvl>, period\_start\_dt, period\_end\_dt, promo\_flg,

/\*fake group by, just to have \*\*\_dag\_lvl ids in output \*/

product\_lvl\_id<dag\_product\_lvl>, location\_lvl\_id<dag\_location\_lvl>, customer\_lvl\_id<dag\_customer\_lvl>, distr\_channel\_lvl\_id<dag\_distr\_channel\_lvl>

1. Calculate base value of each product\_id, location\_id, customer\_id, distr\_channel\_id within product\_lvl\_id<dag\_product\_lvl>, location\_lvl\_id<dag\_location\_lvl>, customer\_lvl\_id<dag\_customer\_lvl>, distr\_channel\_lvl\_id<dag\_distr\_channel\_lvl>

SELECT 2.**\*\*\_dag\_lvl**, promo\_flg,

COALESCE( SUM(BASE), 0) as BASE\_dag\_lvl

COALECSE(COUNT(), 0) as cntdagl\_lvl

period\_start\_dt, period\_end\_dt

FROM 2.

GROUP BY product\_lvl\_iddagl\_product\_lvl>, location\_lvl\_id<dag\_location\_lvl>, customer\_lvl\_id<dag\_customer\_lvl>, distr\_channel\_lvl\_id< dag \_distr\_channel\_lvl>, period\_start\_dt, period\_end\_dt, promo\_flg

1. 3 left join 4 on correspondent \*\*\_lvl-s and period\_start\_dt, period\_end\_dt and keep base values for out-lvl and dag-lvl for all quadruples of out-lvl

SELECT 3.\*,

3.BASE\_OUT\_LVL as BASE\_OUT\_LVL

4.BASEdagl\_lvl as BASE\_dag\_LVL

3.cnt\_out\_lvl

4.cnt\_dag\_lvl

FROM 3

LEFT JOIN 4 on product\_lvl\_id<dag\_product\_lvl>, location\_lvl\_id<dag\_location\_lvl>, customer\_lvl\_id<dag\_customer\_lvl>, distr\_channel\_lvl\_id<dag\_distr\_channel\_lvl>, period\_start\_dt, period\_end\_dt, promo\_flg

Assumption: there is no need to move demand forecast from all phased-out quadruples to active ones.

Output: As a result of this step, a table of the following structure is constructed, *T3.*

|  |  |
| --- | --- |
| Column Name | Description |
| **product\_lvl\_id<m>** | I m =dag**\_product\_lvl** |
| **location\_lvl\_id<n>** | I, n =dag**\_location\_lvl** |
| **customer\_lvl\_id<l>** | d, l =dag**\_customer\_lvl** |
| **distr\_channel\_lvl\_id<k>** | , k = dag**\_distr\_channel\_lvl** |
| **product\_lvl\_id<M>** | ID of product group for which specific ML model will be trained, m = **out\_product\_lvl** |
| **location\_lvl\_id<N>** | ID of location group for which specific ML model will be trained, n = **out\_location\_lvl** |
| **customer\_lvl\_id<K>** | ID of customer group for which specific ML model will be trained, l = **out\_customer\_lvl** |
| **distr\_channel\_lvl\_id<L>** | ID of distr\_channel group for which specific ML model will be trained, k = **out\_distr\_channel\_lvl** |
| **PERIOD\_START\_DT** | Period start date |
| **PERIOD\_END\_DT** | Period end date |
| **PROMO\_FLG** | 1|promo event was active  0|no promo event |
| **BASE\_OUT\_LVL** | Base value at out-lvl |
| **BASE\_DAG\_LVL** | Base value at dag-lvl |

## Calculate disaggregated forecast

**Inputs:** T3, HYBRID\_FORECAST

**Transformation algorithm:**

0.i select max(PERIOD\_DT)

into :fc\_end

from HYBRID\_FORECAST

0.ii

select sum(intnx(step\_interval, 0, horizon\_length, 's'))

into:delays\_config\_length from DELAYS\_CONFIG;

0.iii

if max(PERIOD\_DT) > intnx(day, IB\_HIST\_END\_DT, delays\_config\_length.)

then hyb = vf /\*mid-term forecast\*/

else hyb = ml**;**

0.iv /\*aggregate hybrid\_forecast on dag\_lvl\*/

SELECT

(t1.PRODUCT\_LVL\_ID&&hyb. \_product\_lvl as product\_&hyb. l\_lvl,

(t1.LOCATION\_LVL\_ID&&hyb. \_location\_lvl as location\_&hyb. \_lvl,

(t1.CUSTOMER\_LVL\_ID&&hyb. \_customer\_lvl as customer\_&hyb. \_lvl,

(t1.DISTR\_CHANNEL\_LVL\_ID&&hyb. \_distr\_channel\_lvl as distr\_channel\_&hyb. \_lvl,

(t1.PERIOD\_DT),

(t1.PERIOD\_END\_DT),

(t1.SEGMENT\_NAME),

(t1.demand\_type),

(t1.assortment\_type),

(t1.VF\_FORECAST\_VALUE),

(t1.ML\_FORECAST\_VALUE),

(t1.HYBRID\_FORECAST\_VALUE),

(t1.ENSEMBLE\_FORECAST\_VALUE),

(t1.FORECAST\_SOURCE),

(t2.product\_lvl\_id& product\_dag\_lvl) as product\_dag\_lvl,

(t3.LOCATION\_lvl\_id&location\_dag\_lvl) as location\_dag\_lvl,

(t4.customer\_lvl\_id&customer\_dag\_lvl) as customer\_dag\_lvl,

(t5.distr\_channel\_lvl\_id& distr\_channel\_dag\_lvl) as distr\_channel\_dag\_lvl

FROM

HYBRID\_FORECAST t1

INNER JOIN (SELECT DISTINCT

t1.product\_lvl\_id&i

%do i=%eval(dag\_product\_lvl+ 1) %to &hyb. \_product\_Lvl:

,t1.product\_lvl\_id&i.

%end;

FROM IN\_PRODUCT t1)

INNER JOIN (SELECT DISTINCT

t1.LOCATION\_lvl\_id&i

%do i=%eval(dag\_LOCATION\_lvl+ 1) %to &hyb. \_LOCATION\_Lvl:

,t1.LOCATION\_lvl\_id&i.

%end;

FROM IN\_LOCATION t1)

INNER JOIN (SELECT DISTINCT

t1.CUSTOMER\_lvl\_id&i

%do i=%eval(dag\_CUSTOMER\_lvl+ 1) %to &hyb. \_CUSTOMER\_Lvl:

,t1.CUSTOMER\_lvl\_id&i.

%end;

FROM IN\_CUSTOMER t1)

INNER JOIN

(SELECT DISTINCT

t1.DISTR\_CHANNEL\_lvl\_id&i

%do i=%eval(&hyb. \_DISTR\_CHANNEL\_lvl+ 1) %to &hyb. \_DISTR\_CHANNEL\_Lvl:

,t1.DISTR\_CHANNEL\_lvl\_id&i.

%end;

FROM IN\_DISTR\_CHANNEL t1)

**0.ii**

SELECT

(t1.PRODUCT\_DAG\_LVL),

(t1.LOCATION\_DAG\_LVL),

(t1.CUSTOMER\_DAG\_LVL),

(t1.DISTR\_CHANNEL\_DAG\_LVL),

(t1.PERIOD\_DT),

(t1.PERIOD\_END\_DT),

(SUM(t1.VF\_FORECAST\_VALUE)) AS VF\_FORECAST\_VALUE,

(SUM(t1.ML\_FORECAST\_VALUE)) AS ML\_FORECAST\_VALUE,

(SUM(t1.HYBRID\_FORECAST\_VALUE)) AS HYBRID\_FORECAST\_VALUE,

(SUM(t1.ENSEMBLE\_FORECAST\_VALUE)) AS ENSEMBLE\_FORECAST\_VALUE

FROM

0.i t1

GROUP BY (t1.PRODUCT\_DAG\_LVL),

(t1.LOCATION\_DAG\_LVL),

(t1.CUSTOMER\_DAG\_LVL),

(t1.DISTR\_CHANNEL\_DAG\_LVL),

(t1.PERIOD\_DT),

(t1.PERIOD\_END\_DT)

1. Add base values to HYBRID\_FORECAST

SELECT 0.ii\*

,SUM(BASE\_OUT\_LVL\*(min(T3.period\_end\_dt, I1.period\_end\_dt) – max(T3.period\_start\_dt, I1.period\_dt) + 1)) as BASE\_OUT\_LVL

,SUM(BASE\_DAG\_LVL\*(min(T3.period\_end\_dt, I1.period\_end\_dt) – max(T3.period\_start\_dt, I1.period\_dt)+1)) as BASE\_DAG\_LVL

, SUM(CNT\_OUT\_LVL\*(min(T3.period\_end\_dt, I1.period\_end\_dt) – max(T3.period\_start\_dt, I1.period\_dt) + 1)) as CNT\_OUT\_LVL

, SUM(CNT\_ML\_LVL\*(min(T3.period\_end\_dt, I1.period\_end\_dt) – max(T3.period\_start\_dt, I1.period\_dt) + 1)) as CNT\_ML\_LVL

, CASE WHEN calculated BASE\_DAG\_LVL = 0 THEN

CASE WHEN calculated CNT\_OUT\_LVL = 0 THEN 0 ELSE

CNT\_OUT\_LVL/CNT\_DAG\_LVL END

ELSE

BASE\_OUT\_LVL/ BASE\_DAG\_LVL

END as SHARE

From HYBRID\_FORECAST as I1

left join T3 on product\_lvl\_id<ml\_product\_lvl>, location\_lvl\_id<ml\_location\_lvl>, customer\_lvl\_id<ml\_customer\_lvl>, distr\_channel\_lvl\_id<ml\_distr\_channel\_lvl> and

I1.period\_dt <=T3.period\_end\_dt and I1.period\_end\_dt >=T3.period\_start\_dt and

T3.promo\_flg = CASE WHEN I1.DEMAND\_TYPE = ‘promo’ THEN 1 ELSE 0 END

GROUP BY T3.product\_lvl\_id<out\_product\_lvl>, T3.location\_lvl\_id<out\_location\_lvl>, T3.customer\_lvl\_id<out\_customer\_lvl>, T3.distr\_channel\_lvl\_id<out\_distr\_channel\_lvl>, I1.period\_dt, I1.period\_end\_dt

1. Calculate Disaggregated Forecast Value
   1. **VF\_FORECAST\_VALUE = VF\_FORECAST\_VALUE\*Share**
   2. **ML\_FORECAST\_VALUE = ML\_FORECAST\_VALUE\*Share**
   3. **HYBRID\_FORECAST\_VALUE = HYBRID\_FORECAST\_VALUE\*Share**
   4. **ENSEMBLE\_FORECAST\_VALUE = ENSEMBLE\_FORECAST\_VALUE\*Share**

Output: As a result of this step, a table of the following structure is constructed, AGG\_HYBRID\_FORECAST (detailed description see in section 2.6)*.*

## Output from the Algorithm

1. Table with all types of forecast

|  |  |
| --- | --- |
| **AGG\_HYBRID\_FORECAST\_<target variable type>** | |
| Column Name | Description |
| **product\_lvl\_id<m>** | ID of product group for which specific ML model will be trained, m = **out\_product\_lvl** |
| **location\_lvl\_id<n>** | ID of location group for which specific ML model will be trained, n = **out\_location\_lvl** |
| **customer\_lvl\_id<l>** | ID of customer group for which specific ML model will be trained, l = **out\_customer\_lvl** |
| **distr\_channel\_lvl\_id<k>** | ID of distr\_channel group for which specific ML model will be trained, k = **out\_distr\_channel\_lvl** |
| **PERIOD\_DT** | Date of sales in format of **out\_time\_lvl** |
| **PERIOD\_END\_DT** | End date of the period |
| **VF\_FORECAST\_VALUE** | VF forecast value |
| **SEGMENT\_NAME** | Name of segment that was linked to a pair product/location within VF Project (can be missing) |
| **DEMAND\_TYPE** | ‘promo’ or ‘regular’ – type ML model is used to forecast |
| **ASSORTMENT\_TYPE** | new, or old |
| **ML\_FORECAST\_VALUE** | ML forecasted value |
| **HYBRID\_FORECAST\_VALUE** | Hybrid forecast value |
| **ENSEMBLE\_FORECAST\_VALUE** | Ensemble forecast value |
| **FORECAST\_SOURCE** | Forecast source |