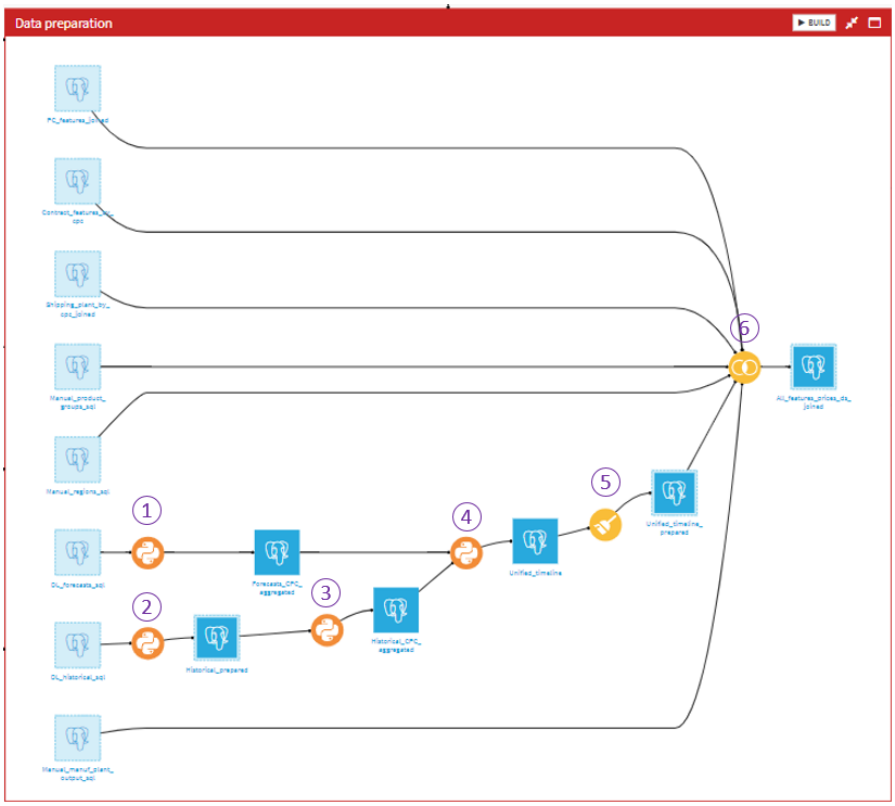
Processing steps (CS)

Forecasts and historical data

You will find below the details of the processing steps for these data.



Forecast data cleaning and aggregation (1)

In this first step, we perform a bit of data filtering and cleaning and then aggregate the forecast data at a CPC level.

Data filtering and cleaning:

1. Columns renaming to match the names used in our backend.
2. Filter on the right GBU (through *gbu_id* field).
3. Filter on the last running month (*running_dt* field).

4. Filter forecasts of current and next 11 months (*forecast_dt* field) => this is mandatory as we will then sum the sales and volumes at a CPC level and want to stay on a 12 months basis.
5. Delete records with null customer and product keys (*shipto_customer_id* / *material_id* for Novecare).
6. Generate our CPC identifier (*shipto_customer_id* / *material_id* for Novecare).
7. Keep only the columns we need.

CPC aggregation:

The data we get from the data lake is at a **CPC + distribution channel + month** level. This means that each distribution channel for a given CPC can have its own value for some dimensions (especially incoterms, group of activity and enterprise segment).
Given the fact that we only want one record by CPC, we will keep the values associated to the **highest amount of sales**, following these steps :

1. Compute total sales by distribution channel for every CPC (new field *total_sales_by_channel*).
2. Sort the data by cpc, total_sales_by_channel (highest first) and month (closest first).
3. Aggregate the sorted data by CPC :
    a. keeping the first non-null value for all the dimensions and last_invoice_price
    b. summing the measures (volume and sales)

The output of this first step therefore contains the **forecasted data aggregated at a CPC level**, which means we always get 1 record for each CPC :

- The dimensions displayed are the ones of the closest forecasted month for the distribution channel with the most sales.
- The volume and sales measures are summed over the 12 first months of forecasts.
- The unit price is not yet computed at this stage.

Historical data cleaning (2)

This is the equivalent of the forecast data cleaning step.

Note that :

- This time, we **separate** the aggregation step as we need the cleaned data at a CPC + month level in order to **compute sales evolution features** (TODO : add doc link).
- We do not need to filter months as it is already done by the Data Lake.

Data filtering and cleaning:

1. Columns renaming to match the names used in our backend.
2. Filter on the right GBU (through *gbu_id* field).
3. Delete records with null customer and product keys (*shipto_customer_id* / *material_id* for Novecare).
4. Generate our CPC identifier (*shipto_customer_id* / *material_id* for Novecare).
5. Keep only the columns we need.

The output of this first step contains the **historical data aggregated at a CPC + incoterms + month level**, which means we always get 1 record for each CPC, incoterms and month.

Historical CPC aggregation (3)

The data we get from the data lake is at a **CPC + incoterms + month level**. Each incoterms for a given CPC can have its own value for some dimensions
Given the fact that we only want one record by CPC, we will keep the values associated to the **highest amount of sales**, following these steps :

1. Compute total sales by incoterms for every CPC (new field *total_sales_by_incoterms*).
2. Sort the data by cpc, total_sales_by_incoterms (highest first) and month (closest first).
3. Aggregate the sorted data by CPC :
    a. keeping the first non-null value for all the dimensions and last_invoice_price
    b. summing the measures (volume and sales)

The output of this first step contains the **historical data aggregated at a CPC level**, which means we always get 1 record for each CPC.

Historical and forecast data merge (4)

In this step, we are **merging together** the forecast and historical data. For each of the dimension, we take the value from the **forecast** data when available, else we **revert to the value of the historical** data.

It allows us to fill some missing data in the forecasts and therefore still keep the CPC in the analysis.

For the following measures, we keep both the forecasted and historical values in their dedicated fields : sales, volume, last invoice price.

We therefore end up with a dataset having one record by CPC and one value for every dimensions, as long as both the historical and forecasted measures.

Unified timeline final preparation (5)

What we call the **unified timeline** is the dataset we get by **combining** the historical and forecast data as described above. Once we do this, we can apply our last steps of data preparation.

Filters:

Removal of the records having no *product_line_04* or *forecast_qty_vkg*

Unit price computation:

Since now, we did not manipulate **unit prices** at all, even if they are the central element of our price optimization. Indeed, due to the previously mentioned aggregation steps, we can not use the unit prices directly available in the Data Lake but **we can compute it again** based on the sales and volumes we have at this stage.

- Compute forecasted_unit_price by multiplying *forecast_qty_vkg* and *forecast_sales_eur.*
- Compute historical_unit_price by multiplying *qty_sold_external_vkg* and *net_sales_external_eur.*
- Compute the final unit price by taking the first non-0 value in the following order :
    - Forecasted unit price
    - Last invoice price from forecasts
    - Last invoice price from historical data
    - Historical unit price
- Filter out the records not having a positive final unit price.

Final sales computation:

For some of our features, we have to use the sales amount of the CPC.

Due to the fact that we only filter out CPC having no forecasted volume, we still face (rare) situations in which a CPC **has forecasted volume but no sales related**. When it happens, the unit price we get is from the historical data and the CPC kept in our models. Therefore, we also **have to re-compute the sales** based on the final unit price (which should be equivalent as the historical sales).

Data formatting:

- Remove leading 0s on *product_id* column.
- Fill empty values of all sales, volumes and prices measures by 0.
- Round volume and money measures.
- Format the customer and product codes properly.
- Capitalize the *product_line_04* and *gbu_product_family* fields
- Map some of the *product_line_04* values to the ones we used to have to limit the impacts on the rest of the tool.

Is distributor creation:

This boolean feature is computed based on the *gbu_customer_seg* data : the feature value is *true* if it contains "Distributor", else *false*

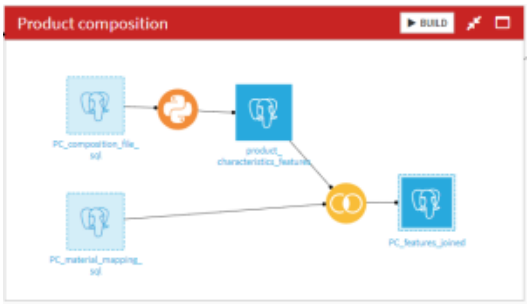
List of fields used from V_FACT_sales_forecast_enriched_current

```
[
    'shipto_customer_medium_name',
    'group_of_activity_id',
    'group_of_activity_name',
    'country_ship_to_id',
    'gbr_enduse_name',
    'gbr_enterprise_segment_w_name',
    'gbu_customer_segment_name',
    'product_id',
    'gbu_product_family_name',
    'soldto_customer_id',
    'soldto_customer_medium_name',
    'material_id',
    'shipto_customer_id',
    'soldto_customer_corporate_group_id',
    'soldto_customer_corporate_group_name',
    'incoterms_id',
    'manufacturing_plant_id',
    'gbr_market_name',
    'forecast_qty_vkg',
    'forecast_net_price_eur_vkg',
    'forecast_sales_eur',
    'last_invoice_price_eur_vkg',
    'shipto_customer_corporate_group_id',
    'shipto_customer_corporate_group_name',
    'manufacturing_plant_name',
    'forecast_dt',
    'shipping_plant_id',
    'country_ship_to_name',
    'h4_l4_name',
    'final_consignee_id',
    'final_consignee_customer_medium_name',
    'final_consignee_customer_corporate_group_name'
]
```

List of fields used from V_FACT_sales_history_cpc_last12months

```
[
    'shipto_customer_medium_name',
    'group_of_activity_id',
    'group_of_activity_name',
    'country_ship_to_id',
    'gbr_enduse_name',
    'gbr_enterprise_segment_w_name',
    'gbu_customer_segment_name',
    'product_id',
    'gbu_product_family_name',
    'soldto_customer_id',
    'soldto_customer_medium_name',
    'material_id',
    'shipto_customer_id',
    'soldto_customer_corporate_group_id',
    'soldto_customer_corporate_group_name',
    'incoterms_id',
    'manufacturing_plant_id',
    'gbr_market_name',
    'qty_sold_external_vkg',
    'unit_price_eur_vkg',
    'net_sales_external_eur',
    'last_invoice_price_eur_vkg',
    'shipto_customer_corporate_group_id',
    'shipto_customer_corporate_group_name',
    'manufacturing_plant_name',
    'pnl_posting_month_dt',
    'last_invoice_price_pnl_posting_month_dt',
    'shipping_plant_id',
    'country_ship_to_name',
    'h4_l4_name',
    'final_consignee_id',
    'final_consignee_customer_medium_name',
    'final_consignee_customer_corporate_group_name'
]
```

Product characteristics



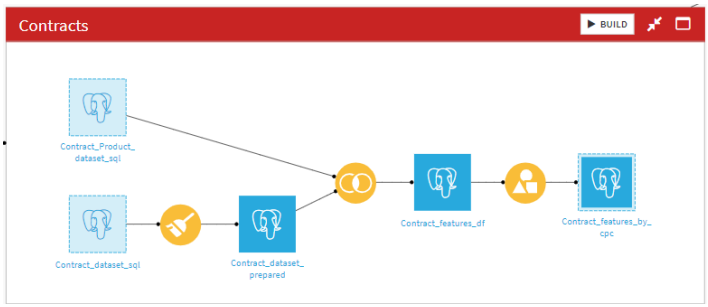
- Both **product composition** files are provided in this GSheet file for now. This source will have to be industrialized through the data lake.

  - The *PC_composition_file_sql* dataset contains the product composition at a "EHS_Product" level of granularity.
  - The *PC_matserial_mapping_sql* dataset contains the mapping between the "EHS_Product" and the material codes we are using in the rest of the project.

- In the Python recipe, we compute the **product composition features** at a **product level** by aggregating together the different substances having the same component type.

- The Join recipe adds the material code to the dataset using the "*EHS_Product*" key mapping.

Contracts



Note : Contracts data are not currently used as features because too few CPCs are under contract in the data we use.

- Both **contracts** files are provided in this GSheet file for now. This source will have to be industrialized through the data lake if used in the future.

  - The *Contract_Product_dataset_sql* dataset contains the link between a contract and the products it is applied on. When a contract does not have any product linked, we consider that it applies to **all of the products** sold to the customer.
  - The *Contract_dataset_sql* dataset contains the contract information at a customer level.

- In the first prepare recipe, we only keep the contracts with a "Signed" status as the other ones are not considered active.

- We then group the dataset by CPC to only keep one contract if several are active. We currently do not have a rule to define priorities so we keep one randomly.

GBU-specific processing

All data preparation relating to a specific GBU is carried out in a dedicated module in the project's library called GBU-specific processing.

For Novecare, the specific data processing for this GBU are as follows:

- For **Sulfosuccinates_Healthcare** family:

  - Products are filtered on only two products ***(product codes: ["90071529", "90071532"]***);
  - Country Ship-To for Docusate product (***product_code = "90071532"***) is grouped into two groups: **Group 1: "US"**, **Group 2: Rest**.

- For **Phosphate_Esters** family:

  - The components_nb feature is divided into two groups: **Group 1: components_nb equals 0**, **Group 2: components_nb different from 0**.