ONEDOT DATA ENGINEER TASK

SUPPLIER DATA INTEGRATION

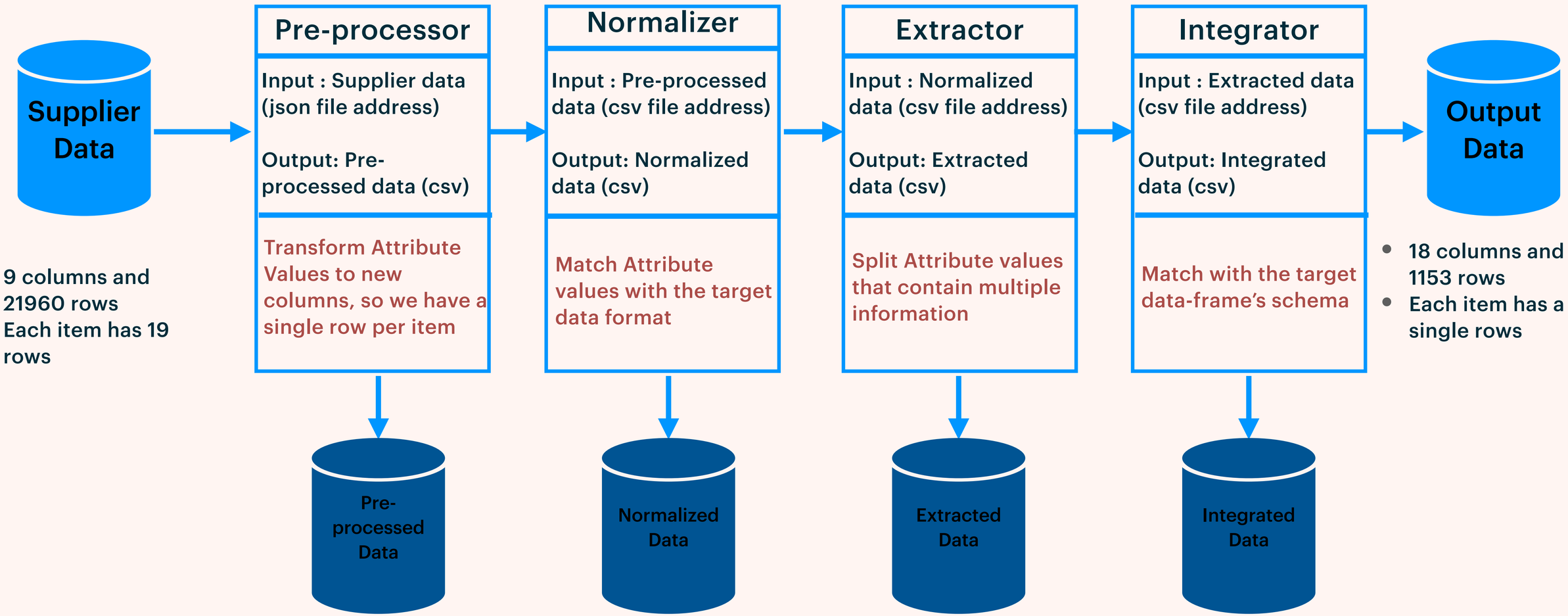
Erdem Böcügöz

OUTLINE

1. General Overview
2. Pre-processing
3. Normalisation
4. Extraction
5. Integration
6. Product matching, enriching existing products, adding new products

GENERAL OVERVIEW

GENERAL OVERVIEW



- 9 columns and 21960 rows
- Each item has 19 rows

- 18 columns and 1153 rows
- Each item has a single rows

TASK 1:
PRE-PROCESSING

PRE-PROCESSING (LOGIC)

- Each item has 19 rows
- Attribute Names and Attribute Values columns can be mapped to new columns where value of the Attribute Names are the name of new columns and the value of Attribute Values are the values for the respected columns
- In order to achieve this goal first we can group by columns : [‘ID’, ’MakeText’, 'ModelText', 'ModelTypeText', 'TypeName', ‘TypeNameFull’], and collect attribute names and values pairs in a list.
- We can remove the column entity ID
- Input data frame has 9 columns and 21960 rows
- Output Data frame has 25 columns and 1153 rows

TASK 2: NORMALISATION

NORMALISATION

- Colors that are in German are translated into the target format.
- Car brands which are all in capital letters are translated into the target format. The only exceptions are BMW and VW which are all in uppercase letters.
- Body types are translated into the target format.

TASK 3: EXTRACTION

EXTRACTION

- **CO2 Emission and Consumption columns are split into new two columns, where one of them contains the value and the other one contains the unit.**

TASK 4: INTEGRATION

INTEGRATION

- Column names are mapped to column names of the target data where possible.
- New columns are added to match with the schema of target data. Since the supplier is a Swiss company some attributes are filled in accordance, e.g. currency: “CHF”, country: “CH”. Since mileage_unit is known it is filled with “kilometer”
- Other columns are removed.
- The order of the columns is re-arranged to match with the target schema.

TASK 5:
PRODUCT MATCHING, ENRICHING EXISTING PRODUCTS, ADDING NEW PRODUCTS

PRODUCT MATCHING, ENRICHING EXISTING PRODUCTS, ADDING NEW PRODUCTS

- In order to match products we can use outer join. For pandas there is a indicator for merge function where it tells if the row exists on both data frames or only in the righ/ left one.

- We can use a similar logic for the pyspark. First we join(outer) the supplier df and the target df on the desired columns . Then we can create a new column named “_merge” by using withColumn function, then we can decide if we have the item in the both data frames by checking if the join columns are null for the data frames separately.

CHALLENGE

- One challenge with this approach is the ambiguity of the columns that are selected for the join operation.
- In the perfect world, we would have the same columns and the normalized values with the target data. However, it may not be the case in the real world. For example, there could be a very similar car from a different supplier (or from the same supplier) in the target data.
- For example, there could be a very similar car from a different supplier (or from the same supplier) in the target data. It is possible to mismatch with this item in the event of missing attribute values.