

Database Lab Assignment Task 2

Group 2

Members: Adeel Ahmed, Muhammad Usman, Sagnik Sarkar

1. Data Cleaning

1. Import Data

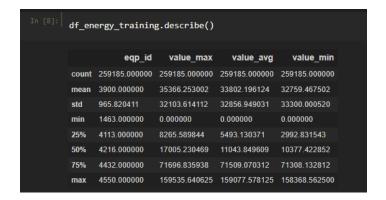
We have loaded the dataset of energy and wip using the pandas library.

```
import pandas as pd
import numpy as np

In [4]:  #Reading the data set and storing into pandas data frame

df_energy_training = pd.read_csv("D:\RCSE_WS21\RCSE_S22\Lab_Training_DB\energy_train_1d_dirty\energy_train_df_wip_training = pd.read_csv("D:\RCSE_WS21\RCSE_S22\Lab_Training_DB\wip_2d_dirty\wip_2d_dirty.csv", sep =
```

2. Applying info and Describe function on Energy data frame



Applying 'info' and 'describe' functions on the energy data frame helped us in getting the count of non-null values for data fields.

We have a total of 5,18,395 records with partially filled data and 2,59,210 records with full data.

So, there is discrepancy in data in the form of null, duplicate and negative values. We will clean the data by dropping those as data is not present in value fields like value_max, value_avg, value_min fields.

3. Cleaning energy data

```
df_energy_training = df_energy_training.convert_dtypes(infer_objects=True, convert_string=True,
                                                              convert_integer=True, convert_boolean=True,
                                                              convert_floating=True)
df_energy_training.dropna()
df_energy_training.drop_duplicates()
df_energy_training = df_energy_training[(df_energy_training['value_max'] >= 0) &
                                             (df_energy_training['value_avg'] >= 0) &
                                             (df_energy_training['value_min'] >= 0)]
df_energy_training.info()
 <class 'pandas.core.frame.DataFrame'>
 Data columns (total 6 columns):
 0 eqp_id 259185 non-null Int64
1 value_max 259185 non-null Float64
 2 value_avg 259185 non-null Float64
 3 value min 259185 non-null Float64
 4 from_ts 259185 non-null string 5 to_ts 259185 non-null string
 memory usage: 14.8 MB
```

Energy data has been cleaned by dropping duplicates, null values and negative values from loaded data. And as a result of the cleaned dataset we got 2,59,185 records.

4. Applying info and describe function on wip data frame:

```
#Info function applied on wip data frame

df_wip_training.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16076 entries, 0 to 16075

Data columns (total 13 columns):

# Column Non-Null Count Dtype
------
0 segment_name 16066 non-null object
1 duration 16076 non-null float64
2 valid_from 16076 non-null object
3 valid_to 16076 non-null object
4 conditioning_recipe 16066 non-null float64
5 eqp_id 16066 non-null float64
6 ei_ctrljob_id 16076 non-null int64
7 entity_id 16076 non-null int64
9 lot_id 16066 non-null float64
10 prodspec_id 12166 non-null float64
11 lc_recipe_id 12166 non-null float64
11 lc_recipe_id 12166 non-null float64
12 prodgrp_id 12166 non-null float64
dtypes: float64(8), int64(2), object(3)
memory usage: 1.6+ MB
```



Used Info and describe methods on wip data frame and got the count of non-null values for data fields as attached.

We have a total of 16,076 records.

We will not drop the data from wip as there is data in value fields.

Also, we will not fill the data in blank fields as empty fields are of id's, so it is not possible to fill the ID values by predicting or taking average of neighboring fields.

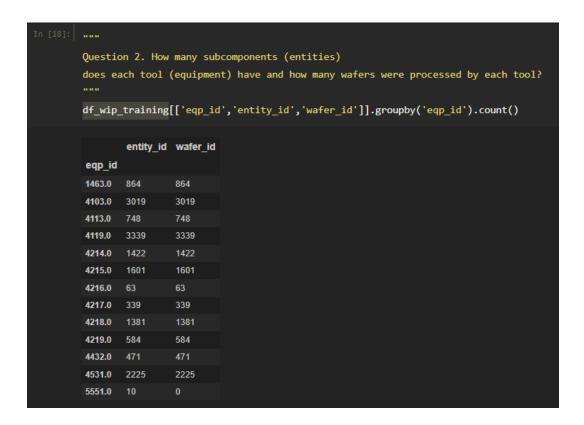
Data type of date fields(valid_from and valid_to) in the wip data frame is stored as an object. We have changed the data type to date time in order to manipulate it easily during data analysis.

2. Data Analysis

1. Which is the most frequent recipe?

627241750.0 is the **most frequent recipe** and query is as below

2. How many subcomponents (entities) does each tool (equipment) have and how many wafers were processed by each tool?



4119 has highest number of subcomponents and wafers with count as 3339

And **5551** has **lowest number of subcomponents and wafers** with count as 10 and 0 respectively

3. Which tool achieved the highest energy consumption and how much was this?



Applying the describe function on the dataset we can get the maximum value.

4550 has the highest energy consumption and it counts to 159535.640625.

4. Which tool needed the most total energy on the given day based on the value avg field and how high was the average per hour?

```
Question 4. Which tool needed the most total energy on the given day based
on the value avg field and how high was the average per hour?
df_energy_training['from_ts'] = pd.to_datetime(df_energy_training['from_ts'])
df energy training['to ts'] = pd.to_datetime(df energy training['to ts'])
seg = df_energy_training.groupby([df_energy_training['eqp_id'],
df_energy_training['from_ts'].dt.hour]).value_avg.sum()
pd.Series.sort_values(seg, ascending = False)
eqp_id from_ts
              54498860.71875
 4218 2
4219 11
                 54440363.4375
              54283414.507812
                54211812.890625
                54167473.828125
       10
                          0.0
                          0.0
 Name: value_avg, Length: 360, dtype: Float64
```

4218 needed the highest total energy per hour and the value is 54498860.71875

5. What activity (segment) was the tool with ID 4216 in from 1999-08-01 07:00 to 1999-08-01 08:00 and how much total energy did this cost?

```
Question 5. What activity (segment) was the tool with ID 4216 in from 1999-08-01 07:00 to 1999-08-01 08:0 and how much total energy did this cost?

"""

df_wip_training[ (df_wip_training['eqp_id']== 4216) & (df_wip_training['valid_from'] > '1999-08-01 07:00') & (df_wip_training['valid_to'] < '1999-08-01 08:00' ) ]['segment_name']

9543 TheHappeningHappened Name: segment_name, dtype: object
```

Code: $df_{energy_training[(df_{energy_training['eqp_id']== 4216) \& (df_{energy_training['from_ts'] > ' 1999-08-01 07:00') \& (df_{energy_training['to_ts'] < '1999-08-01 08:00') } ['value_max'].sum()$

ID 4216 has 'TheHappeningHappened' activity/segment from 1999-08-01 07:00 to 1999-08-01 08:00, and total energy consumption for that duration was 51899993.648438

3. Dataset Join

In order to join two datasets that does not fit into memory, as per to our knowledge we have two possibilities:

1. First is using Index Optimization strategy for two datasets, which includes a set index for both datasets i.e. indexing columns on which both datasets are joined. This way both tables are joined using a lookup table based on indexes.

This idea is similar to RDBMS where we create indexes on tables.

2. Second is using DASK extending Pandas API.A Dask DataFrame is a large parallel DataFrame composed of many smaller pandas DataFrames, split along the index. These pandas DataFrames may live on disk for larger-than-memory computing on a single machine, or on many different machines in a cluster. One Dask DataFrame operation triggers many operations on the constituent pandas DataFrames. A Dask DataFrame is partitioned row-wise, grouping rows by index value for efficiency.

Using Pandas:

import pandas as pd
merged = wip_data.set_index('eqp_id').join(energy_training.set_index('eqp_id'), how='left')

Using dask:

import dask.dataframe as dd
merged = dd.merge(wip_data, energy_training, on='eqp_id', how='inner')