TECHNISCHE UNIVERSITÄT
ILMENAU

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.

```
In [1]: #importing pandas and numpy
        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

```
In [5]: #Using info function on the energy data frame
        df_energy_training.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 518395 entries, 0 to 518394
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   eqp_id     518395 non-null  int64
 1   value_max  259195 non-null  float64
 2   value_avg  259195 non-null  float64
 3   value_min  259195 non-null  float64
 4   from_ts    518395 non-null  object
 5   to_ts      518395 non-null  object
dtypes: float64(3), int64(1), object(2)
memory usage: 23.7+ MB
```

```
In [8]: df_energy_training.describe()
```

	eqp_id	value_max	value_avg	value_min
count	259185.000000	259185.000000	259185.000000	259185.000000
mean	3900.000000	35366.253002	33802.196124	32759.467502
std	965.820411	32103.614112	32856.949031	33300.000520
min	1463.000000	0.000000	0.000000	0.000000
25%	4113.000000	8265.589844	5493.130371	2992.831543
50%	4216.000000	17005.230469	11043.849609	10377.422852
75%	4432.000000	71696.835938	71509.070312	71308.132812
max	4550.000000	159535.640625	159077.578125	158368.562500

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

```
In [7]: #Cleaning Energy Dataframe

        #Casting correct data types by converting object data type to String
        df_energy_training = df_energy_training.convert_dtypes(infer_objects=True, convert_string=True,
                                                               convert_integer=True, convert_boolean=True,
                                                               convert_floating=True)

        #Removing the null values
        df_energy_training.dropna()

        #Eliminating duplciate values
        df_energy_training.drop_duplicates()

        #Eliminating negative values
        df_energy_training = df_energy_training[(df_energy_training['value_max'] >= 0) &
                                                (df_energy_training['value_avg'] >= 0) &
                                                (df_energy_training['value_min'] >= 0)]

        #Checking the info of the data frame after cleaning
        df_energy_training.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 259185 entries, 0 to 501194
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 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
dtypes: Float64(3), Int64(1), string(2)
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:**

```
In [9]: #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
 8   wafer_id            16066 non-null  float64
 9   lot_id              16066 non-null  float64
 10  prodspec_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
```

```
In [10]: #Describe function applied on wip data frame
         df_wip_training.describe()
```

	duration	conditioning_recipe	eqp_id	ei_ctrljob_id	entity_id	wafer_id	lot_id	prodspec_id	lc_
count	1.607600e+04	1.606600e+04	16066.000000	1.607600e+04	1.607600e+04	1.606600e+04	1.606600e+04	1.216600e+04	1.21
mean	1.369468e-01	1.056314e+09	4072.573074	1.011479e+09	1.033165e+09	1.037711e+09	1.179609e+09	1.169181e+09	9.16
std	7.837263e-01	2.309386e+08	638.833486	5.083305e+08	6.186373e+08	5.960129e+08	5.769997e+08	5.035335e+08	5.38
min	5.104167e-09	1.613750e+08	1463.000000	8.925000e+03	1.757400e+04	4.150000e+02	4.208000e+03	1.489750e+05	4.18
25%	9.145689e-05	1.001096e+09	4113.000000	6.929987e+08	6.126679e+08	5.757592e+08	7.799759e+08	9.218701e+08	4.43
50%	1.033939e-03	1.001096e+09	4214.000000	1.001096e+09	9.427769e+08	1.001096e+09	1.160031e+09	1.015621e+09	7.34
75%	1.730069e-02	1.001096e+09	4218.000000	1.357678e+09	1.713947e+09	1.527186e+09	1.736831e+09	1.534898e+09	1.52
max	8.910985e+00	2.123242e+09	5551.000000	2.144058e+09	2.129333e+09	2.147060e+09	2.144207e+09	2.095984e+09	2.12

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.

```
In [14]: #Changing the object data type to date time
         df_wip_training['valid_from'] = pd.to_datetime(df_wip_training['valid_from'])

         df_wip_training['valid_to'] = pd.to_datetime(df_wip_training['valid_to'])
```

2. Data Analysis

1. **Which is the most frequent recipe?**

```
In [16]: """
         Data Analysis
         Question 1. Which is the most frequent recipe?
         """
         df_wip_training['lc_recipe_id'].value_counts().keys()[0]

Out[16]: 627241750.0
```

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?**

```
In [18]: """
         Question 2. How many subcomponents (entities)
         does each tool (equipment) have and how many wafers were processed by each tool?
         """
         df_wip_training[['eqp_id','entity_id','wafer_id']].groupby('eqp_id').count()
```

eqp_id	entity_id	wafer_id
1463.0	864	864
4103.0	3019	3019
4113.0	748	748
4119.0	3339	3339
4214.0	1422	1422
4215.0	1601	1601
4216.0	63	63
4217.0	339	339
4218.0	1381	1381
4219.0	584	584
4432.0	471	471
4531.0	2225	2225
5551.0	10	0

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?**

```
In [19]: #Question 3. Which tool achieved the peak energy consumption and how much was this?
         df_energy_training.describe()
```

	eqp_id	value_max	value_avg	value_min
count	259185.000000	259185.000000	259185.000000	259185.000000
mean	3900.000000	35366.253002	33802.196124	32759.467502
std	965.820411	32103.614112	32856.949031	33300.000520
min	1463.000000	0.000000	0.000000	0.000000
25%	4113.000000	8265.589844	5493.130371	2992.831543
50%	4216.000000	17005.230469	11043.849609	10377.422852
75%	4432.000000	71696.835938	71509.070312	71308.132812
max	4550.000000	159535.640625	159077.578125	158368.562500

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?**

```
In [23]: """
         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?
         """


         #Changing the string data type of 'from_ts' and 'to_ts' to date time from the energy data frame
         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
          4218    2          54498860.71875
          4219    11          54440363.4375
                  3          54283414.507812
          4207    5          54211812.890625
                  10         54167473.828125
                             ...
          4433    8                 0.0
                  7                 0.0
                  6                 0.0
                  5                 0.0
                  12                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?**

```
In [26]: """
         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')
```