List of functions

**Completion Date Extraction**

Controller

**extractDataDate ()**  
Main method for completion date extraction processes

**extractCompletedDate ()**  
Extract completion date from input text (resolution) and return result.

**dateSplitter ()**  
Split input date string using input splitter character and return the date in following format, ‘dd-mm-yy’

**checkDateValidity ()**  
Throw error if input date string does not exist, return true if it exists

**adjustDate ()**  
Adjust input date string to one day after input creation date if the difference is bigger than 5 days

**addOneDay ()**  
Add one day to input creation date, return the new date as string

DAO (Data Access Object)

**getResDateByPartsStationProbCat ()**  
Get resolution and creation date for completion date extraction

**getCompletedDateFromStationDowntime ()**  
Get completed date from Mesralink data

**insertCompletedDate ()**  
Insert extracted completion date into service flowco duplicate table

**Pump & Gas Type Extraction**

Controller

**extractDataPumpGasType ()**  
Main method for pump and gas type extraction processes

**extractGasType ()**  
Extract gas type from input text (problem description) and add the result to gas position list. Return result text after extraction.

**checkNumberOfOccurance ()**  
Return number of input filter occurrences in input text.

**extractPump ()**  
extract pump from input text (problem description) and add the result to pump position list. Return result text after extraction.

**checkNumber ()**  
Check if the input string number is a valid integer. Return true if valid, false otherwise.

**pumpSplitter ()**  
Split the numbers inside the number string and return the result

**pumpGasMatching ()**  
Match the gas position list and pump position list using position data. Return the result as a list of strings.

**checkOneSide ()**  
Return true if the gas type or pump always on one side. Example: {pump,pump,gas,gas} or {gas,gas,pump,pump}

DAO (Data Access Object)

**getProbDescByPartsStationProbCat ()**  
Get problem description for pump and gas type extraction

**getPumpGasTypeFromStationDowntime ()**  
Get pump and gas type from Mesralink data

**getAllPumpFromProfile ()**  
Get all pumps and gas type list from station profile (11 pilot) filtered by input station

**insertPumpGas ()**  
Insert matched pump and gas type into service flowco duplicate table

**insertDuplicatePumpGas ()**  
If more than one matched pump and gas type found, duplicate the original record and adjust the pump and gas type accordingly in service flowco duplicate table

**Extracted Data Table**

DAO (Data Access Object)

**scriptCreateExtractedDataTable ()**  
Create new extracted data table in the database

**scriptInsertAllPumpGasExtractedDataTable ()**  
Insert the starting point for each entity at date: 2017-01-01 into extracted data table

**scriptInsertExtractedDataTable ()**  
Extract data from service flowco duplicate into extracted data table

**scriptAlterExtractedDataTable ()**  
Alter the extracted data table to add the following columns: days to action, lifetime, transaction days, transaction count, adjusted transaction count, total volume, adjusted total volume

**Lifetime, Days to Action, Transaction Count, Adjusted Transaction Count, Total Volume, Adjusted Total Volume Extraction**

Controller

**extractDataLifetime ()**Main method for lifetime extraction, transaction count and total volume extraction from transaction data and adjustment of transaction count and total volume

**calculateAdjustedTransactionCountVolume ()**  
Return adjusted value for transaction count and total volume it the inputs does not contain 0. Formula: (lifetime - daystoAction) \* countVolume / transactionDays

DAO (Data Access Object)

**getDatesByStationPumpGas ()**  
Get creation date and completion date for lifetime extraction

**insertLifetime ()**  
Insert extracted lifetime into extracted data table

**insertDaysToAction ()**  
insert extracted days to action into extracted data table

**getDaysToAction ()**  
Get days to action form extracted data table

**getTransactionCountVolume ()**  
Get transaction count, total volume and transaction days from transaction data

**insertTransactionCountVolume ()**  
Insert extracted transaction count, total volume and transaction days into extracted data table

**updateAdjustedTransactionVolume ()**  
Insert adjusted transaction count and adjusted total volume

**Problem Type & Replacement Parts Number Encoding, Shifting Encoded Data**

Controller

**extractDataProblemEncode ()**  
Main method for extraction of problem type, encode it and insert the result into problem encode table

**extractDataPartsEncode ()**  
Main method for extraction of replacement parts number, encode it and insert the result into parts encode table

**shiftEncodedData ()**  
Shift the encoded data in the problem and parts encode tables upwards by one record or one row for each entity

DAO (Data Access Object)

**scriptCreateEncodeTable ()**  
Create problem encode table or parts encode table

**getDetailsExtractedData ()**  
Get details from extracted data table

**getProblemTypeByCategory ()**  
Get all problem type filtered by problem category

**getProblemTypeByDetails ()**  
Get problem type from service flowco duplicate filtered by the details input

**insertProblemTypeEncode ()**  
Insert encoded problem type into problem encode table

**getReplacementPartsByCategory ()**  
Get all replacement parts number filtered by problem category

**getReplacementPartsByDetails ()**  
Get replacement parts number from service flowco duplicate filtered by the details input

**insertReplacementPartsEncode ()**  
Insert encoded replacement parts number into parts encode table

**getEncodedData ()**  
Get problem encode table or parts encode tables

**updateEncodeData ()**  
Update problem encode table or parts encode tables after shifting

**General Function**

Controller

**getDifferenceDays ()**  
Return the difference of two input dates in days

**executeScript ()**  
Help with the execution of functions inside Dao java file

**exportToCSV ()**  
Main method to export the final extraction result from database into csv file

**followCVSformat ()**  
Cover for cases which data contains ‘ ” ’ character

**writeLine ()**  
Write list of input strings into a line in CSV file

DAO (Data Access Object)

**getConnection ()**  
Get current database connection

**setConnection ()**  
Set new database connection

**closeConnection ()**  
Close current database connection

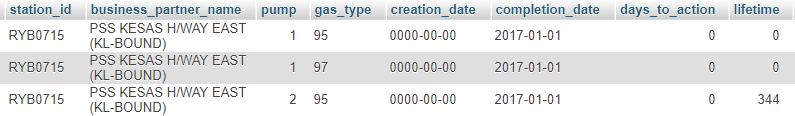
**getTotalPartsByStation ()**  
Get the number of total parts for the input station

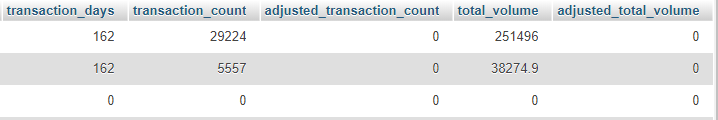
**getIdentityExtractedData ()**  
Get data for each entity (entity - result after group by station, pump and gas type)

**getResultTable ()**  
Get the combination of extracted data table, problem encode table and parts encode table

Output

**extracted\_data\_nozzle**





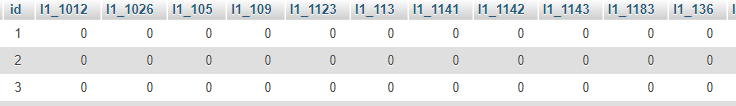
1. **id**: Id of each row. Match with ids of problem\_encode\_nozzle and parts\_encode\_nozzle.
2. **case\_id**: Extracted from service\_flowco or raw\_data\_pdb\_station\_downtime (Mesralink data) if it is available. If it has multiple case\_id, only the first one would be selected.
3. **station\_id**: Id of the station.
4. **business\_partner\_name**: Name of the station.
5. **pump**: Pump or dispenser number
6. **gas\_type**: Fuel type.
7. **creation\_date**: The date when the report is created. Assumed that it is the date when the failure occured. creation\_date would be null or empty ‘0000-00-00’ for the starting point ‘2017-01-01’ (which is also the first completion\_date).
8. **completion\_date**: The date when the report was marked completed. Assumed that it is the date when the repair was completed.
9. **days\_to\_action**: The number of days between creation\_date and completion\_date. If the days\_to \_action is bigger than 5, then it is assumed that the extracted completion\_date may have been entered or keyed in wrongly. In such cases, the completion\_date would be corrected to one day after creation\_date.
10. **lifetime**: Lifetime is the number of days between the completion\_date of current record and completion\_date of next record for each particular station, pump and gas type. If there is only single record then the lifetime would be 0. The lifetime for the final record would also be 0.
11. **transaction\_ days**: The number of days that transaction occurred. It is measured by counting distinct dates in transaction data from completion\_date of current record to creation\_date of next record for each particular station, pump and gas type. For the last record, it is counted until current system date.
12. **transaction\_count**: The number of transactions that occurred between completion\_date of current record to creation\_date of next record for each particular station, pump and gas type. For the last record, it is counted until current system date.
13. **adjusted\_transaction\_count**: Since the transaction data have time windows and incomplete, the transaction\_count needs to be adjusted to represent the actual transaction count during the lifetime. It is adjusted using simple ratio (adjusted\_transaction\_count = (lifetime – days\_to\_action) \* transaction\_count / transaction\_days). This is done with the assumption that there is at least a transaction for each particular station, pump and gas type every day.
14. **total\_volume**: The amount in liters that was dispensed between completion\_date of current record to creation\_date of next record for each particular station, pump and gas type. For the last record, it is measured until current system date.
15. **adjusted\_total\_volume**: Since the transaction data have time windows and incomplete, the total\_volume needs to be adjusted to represent the actual total volume dispensed during the lifetime. It is adjusted using simple ratio (15. adjusted\_total\_volume = (lifetime – days\_to\_action) \* total\_volume / transaction\_days). This is done with the assumption that there is at least a transaction for each particular station, pump and gas type every day.

**problem\_encode\_nozzle**



This table was constructed by using all problem type that occured in service\_flowco for problem category nozzle. There are 8 columns including id that matches with extracted\_data\_nozzle and parts\_encode\_nozzle. The data encoded as 0 if it does not occur and 1 if it does occur.

**parts\_encode\_nozzle**



This table was constructed by using all replacement parts number that occured in service\_flowco for problem category nozzle. There are 125 columns including id that matches with extracted\_data\_nozzle and problem\_encode\_nozzle. The data encoded as 0 if it does not occur and 1 if it does occur.