WW Pmod SMLD Quality Control System

This project builds, maintains a database, and monitors SSVE Pmod performance by week.

It minimizes significant labor hours, eliminates human errors and speeds up the whole process.

This project makes monitoring WW Pmod SMLD Quality in real time possible.

Author

SSVE TVQA member @Zhang Liang, 20210804

Changelog

- v0.01, initial build
- v0.02, clean and format algorithms
- v0.03, change design pattern: split into two dataframes: defects, inputs
- v0.04, add more columns to prepare for visualization
- v0.05, fix TVPlant column, "SO'EM" -> "SOEM"

Design pattern

There was an impact to performance when using Python to clean "inputs" data in source files;

After built a VBA class API to preprocess the clean operation, the impact is alleviated;

Pattern

- (Class) using VBA preprocess class API to do basic data cleaning;
- (Class) using Python + Pandas to build a DataFrame Merger API to wrangle the preprocessed source files with split control: "defects", "inputs";
- (Class) using SQL to control all data entries and poka-yoke duplicate entries;
- (Class) exports SQL database to excel "pmod_smld.xlsx" with unique columns for further data visualization;
- (Module) using VBA to visualise pmod smld data;

@ZL, 20210806
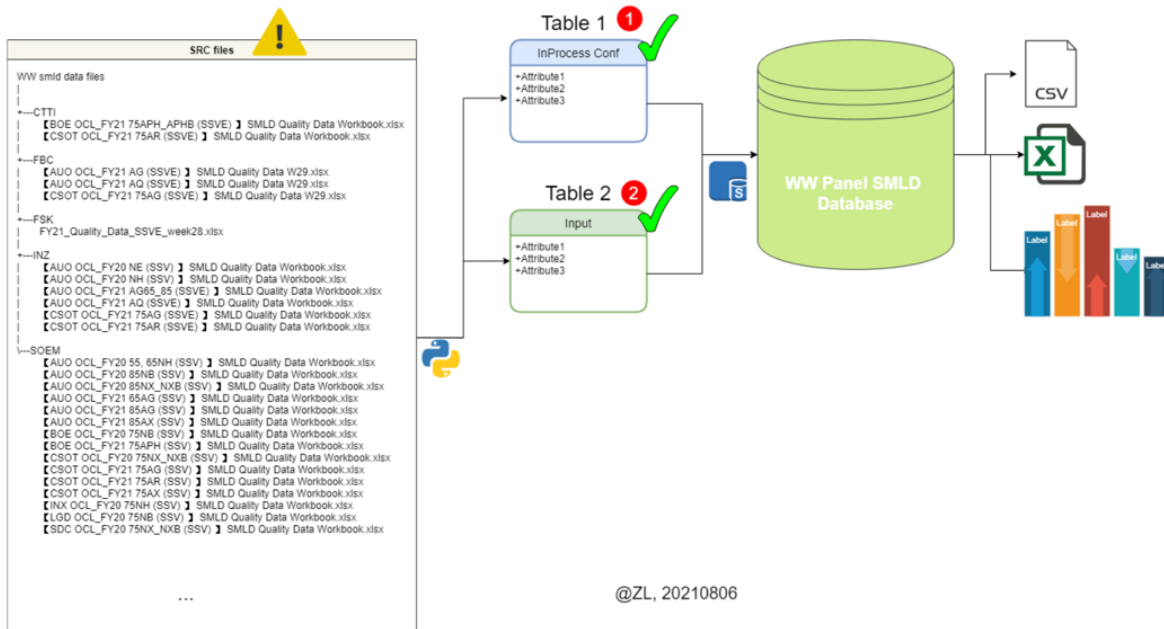
Performance

The average run time of whole process is around 3 minutes.

```
D:\pj_00_codelib\2019_pypj\20210805 QA_weekly_report>d:/DevEnv/WPy32-3741/python-3.7.4/python.exe
2021-08-24 10:11:49,564 Successed. time lapsed(s): 50.41
2021-08-24 10:11:49,580 【BOE OCL_FY21 75APH_APHB (SSVE) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:11:56,034 【CSOT OCL_FY21 75AR (SSVE) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:02,391 【AUO OCL_FY21 AG (SSVE) 】 SMLD Quality Data W32.xlsx
2021-08-24 10:12:08,645 【AUO OCL_FY21 AQ (SSVE) 】 SMLD Quality Data W32.xlsx
2021-08-24 10:12:15,143 【CSOT OCL_FY21 75AG (SSVE) 】 SMLD Quality Data W32.xlsx
2021-08-24 10:12:21,677 FY21_Quality_Data_SSVE_week33.xlsx
2021-08-24 10:12:22,498 【AUO OCL_FY20 NE (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:24,667 【AUO OCL_FY20 NH (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:26,627 【AUO OCL_FY21 AG65_85 (SSVE) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:29,846 【AUO OCL_FY21 AQ (SSVE) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:33,233 【CSOT OCL_FY21 75AG (SSVE) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:36,558 【CSOT OCL_FY21 75AR (SSVE) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:39,218 【AUO OCL_FY20 55, 65NH (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:45,161 【AUO OCL_FY20 85NB (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:50,713 【AUO OCL_FY20 85NX_NXB (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:12:56,698 【AUO OCL_FY21 65AG (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:02,418 【AUO OCL_FY21 85AG (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:08,188 【AUO OCL_FY21 85AX (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:14,084 【BOE OCL_FY20 75NB (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:19,746 【BOE OCL_FY21 75APH (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:25,680 【CSOT OCL_FY20 75NX_NXB (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:31,511 【CSOT OCL_FY21 75AG (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:37,216 【CSOT OCL_FY21 75AR (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:43,066 【CSOT OCL_FY21 75AX (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:48,743 【LGD OCL_FY20 75NB (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:13:54,324 【SDC OCL_FY20 75NX_NXB (SSV) 】 SMLD Quality Data Workbook.xlsx
2021-08-24 10:14:01,368 SQL update completed
2021-08-24 10:14:01,386 SQL update completed
2021-08-24 10:14:01,387 Successed. time lapsed(s): 131.81
```

VBA preprocess class API

```vba
Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   a WW PMod SMLD wrangler class handles to format source data
```

```vba
'  + The class has the following functionalities:
'       - enumerate source files
'       - complement miss data based on folder? file name?
'       - create new worksheet "input"
'       - transfer WW input data to "Input"

' @ZL, 20210818

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Private srcWB As Workbook
Private srcWS As Worksheet
Private TV_Plant As String
Private coll_invalid_wsname As ArrayList
Private fy_mode_mapping As Scripting.Dictionary

Const wsname_inProcess As String = "InProcess Conf" 'others
Const wsname_raw_fsk As String = "raw data"   'FSK
Const wsname_inputs As String = "inputs"
Const str_line_name As String = "Line"
Const strFSK As String = "FSK"

Const col_OccurredWC As Integer = 2
Const col_ConfirmWC As Integer = 3
Const col_tvplant As Integer = 7
Const col_line_or_sbi As Integer = 8
Const col_set_name As Integer = 10
Const col_model_name As Integer = 11
Const col_model_id As Integer = 12
Const col_week_code As Integer = 13
Const col_sony_pn As Integer = 14
Const col_classify As Integer = 20
Const start_row As Integer = 2
Const row_header As Integer = 1
Const FY_mod_col As Integer = 21
Const FY_mod_head As String = "FY_mod"

Const len_model_id As Integer = 22 'format: A5028174A$0000389_2111
Const str_sony_pn_dash As String = "-"
Const str_sony_pn_delimiter_dollar As String = "$"
Const str_sony_pn_delimiter_underscore As String = "_"
Const str_sony_pn_fmt As String = "0000000"

Const inputs_TVPlant As String = "TVPlant"
Const inputs_ModName As String = "ModelName"
Const inputs_weekcode As String = "WeekCode"
Const inputs_qty As String = "QTY"
Const inputs_fy As String = "FY_mod"

Const defects_moduleID As String = "Module ID"
Const defects_CellMod As String = "Cell Model"
Const defects_CellID As String = "Cell-ID"
Const defects_Weekcode As String = "Week-Code"
```

```vba
    Const col_oss_ffa_judge As Integer = 18
    Const col_notice_symptom As Integer = 19
    Const col_insert As Integer = 16
    Const col_rename_cell_model As Integer = 16
    Const col_rename_cell_id As Integer = 17
    Const col_rename_weekcode As Integer = 13


    Private common_col_headers As Variant

    Public Sub init(ByRef WB As Workbook, ByVal plant As String)
        Set srcWB = WB
        TV_Plant = plant

        If WSExists(srcWB, wsname_inProcess) Then
            Set srcWS = srcWB.Worksheets(wsname_inProcess)
        ElseIf WSExists(srcWB, wsname_raw_fsk) Then
            Set srcWS = srcWB.Worksheets(wsname_raw_fsk)
        Else
            MsgBox Formatter("WSNotFoundError:  {0} or {1}", wsname_inProcess,
    wsname_raw_fsk), vbInformation, "WSError"
            Exit Sub
        End If

        ' init arraylist
        Set coll_invalid_wsname = New ArrayList
        coll_invalid_wsname.Add wsname_inProcess
        coll_invalid_wsname.Add wsname_raw_fsk
        coll_invalid_wsname.Add wsname_inputs
        coll_invalid_wsname.Add "Fail list"
        coll_invalid_wsname.Add "General Monthly(by Classify)"
        coll_invalid_wsname.Add "General Weekly(by Classify)"
        coll_invalid_wsname.Add "Weekly"
        coll_invalid_wsname.Add "Monthly"

        'init dict
        Set fy_mode_mapping = New Scripting.Dictionary
        fy_mode_mapping.Add "A", "FY21"
        fy_mode_mapping.Add "N", "FY20"
        fy_mode_mapping.Add "S", "FY19"

        'common headers
        common_col_headers = Array("No", "Confirmation W/C", "Occurred W/C", "source",
    "Occurred date", "Confirm date", "TV Plant", "LineName Line / SBI", "Position",
    "Set name", "Model name", "Module ID", "Week-Code", "Sony PN", "SITE", "Cell
    Model", "Cell-ID", "Noticed symptom", "OSS or FFA judgement", "Classify",
    "FY_mod")
    End Sub

    Public Sub clean()
        Call clean_inProcess
        Call clean_Inputs
        Call clean_fileType
    End Sub
```

```vba
Private Function create_fy_mod(ByVal mode_name As String) As String
    Dim key As Variant
    Const na As String = "na"
    Dim rv
    For Each key In fy_mode_mapping.Keys
        rv = InStr(1, UCase(mode_name), key, vbTextCompare)
        If IsNull(rv) Then
            create_fy_mod = vbNullString
        Else
            If rv > 0 Then
                create_fy_mod = fy_mode_mapping(key)
                Exit For
            Else
                create_fy_mod = na
            End If
        End If
    Next
End Function

Private Sub clean_FSK_inProcess(ByRef srcWS_FSK)
    ''' especially clean up FSK in process
    Const header_start_col As Integer = 1
    With srcWS
        If Not .Cells(row_header, FY_mod_col).Value = FY_mod_head Then
            .Columns(col_insert).Insert Shift:=xlToRight,
CopyOrigin:=xlFormatFromLeftOrAbove
            .Columns(col_insert).Insert Shift:=xlToRight,
CopyOrigin:=xlFormatFromLeftOrAbove
            .Range(.Cells(row_header, header_start_col), .Cells(row_header,
FY_mod_col)) = common_col_headers
'           .Cells(row_header, col_model_id).Value = defects_moduleID
'           .Cells(row_header, col_rename_cell_model).Value = defects_CellMod
'           .Cells(row_header, col_rename_cell_id).Value = defects_CellID
'           .Cells(row_header, FY_mod_col).Value = FY_mod_head
'           .Cells(row_header, col_rename_weekcode).Value = defects_Weekcode
        End If
    End With
End Sub

Private Sub clean_inProcess()
    ''' Clean SMLD defect entry
    Dim last_row As Integer: last_row = GetLastRow(srcWS, col_model_id)
    Dim i As Integer
    Dim tmp_val

    If TV_Plant = strFSK Then
        Call clean_FSK_inProcess(srcWS)
    Else
        srcWS.Cells(row_header, FY_mod_col).Value = FY_mod_head
    End If

    For i = start_row To last_row
        With srcWS
```

```vba
            ' Occurred W/C
            If IsEmpty(.Cells(i, col_OccurredWC).Value) Then
                .Cells(i, col_OccurredWC).Value = .Cells(i, col_ConfirmWC).Value
            End If
            ' Confirm W/C
            If IsEmpty(.Cells(i, col_ConfirmWC).Value) Then
                .Cells(i, col_ConfirmWC).Value = .Cells(i, col_OccurredWC).Value
            End If
            ' TV Plant
            If IsEmpty(.Cells(i, col_tvplant).Value) Then
                .Cells(i, col_tvplant).Value = TV_Plant
            End If
            ' Set Name
            If IsEmpty(.Cells(i, col_set_name).Value) Then
                .Cells(i, col_set_name).Value = .Cells(i, col_model_name).Value
            End If
            ' Line Name
            If IsEmpty(.Cells(i, col_line_or_sbi).Value) Then
                .Cells(i, col_line_or_sbi).Value = str_line_name
            End If
            ' Week Code
            If IsEmpty(.Cells(i, col_week_code).Value) Then
                .Cells(i, col_week_code).Value = ParseWeekCode(Trim$(.Cells(i,
col_model_id)))
            End If
            ' Model ID
            tmp_val = .Cells(i, col_model_id).Value
            If Len(tmp_val) < len_model_id And Not IsEmpty(tmp_val) Then
                .Cells(i, col_model_id).Value = Replace(.Cells(i,
col_sony_pn).Value, str_sony_pn_dash, vbNullString) _
                                                        &
str_sony_pn_delimiter_dollar _
                                                        & Format(.Cells(i,
col_model_id).Value, str_sony_pn_fmt) _
                                                        &
str_sony_pn_delimiter_underscore _
                                                        & .Cells(i,
col_week_code)
            End If
            ' OSS or FFA judgement
            If IsEmpty(.Cells(i, col_oss_ffa_judge).Value) Then
                .Cells(i, col_oss_ffa_judge).Value = .Cells(i,
col_notice_symptom).Value
            End If
            ' Classify
            If IsEmpty(.Cells(i, col_classify).Value) Then
                .Cells(i, col_classify).Value = str_sony_pn_dash
            End If
            ' FY_mod
            If Not IsError(.Cells(i, col_set_name)) Then
                .Cells(i, FY_mod_col).Value = create_fy_mod(.Cells(i,
col_set_name).Value)
            End If
        End With
```

```vba
        Next i
End Sub

Private Sub clean_Inputs()
    ''' clean inputs
    Dim WS As Worksheet
    Call Create_WS_on_GivenWB(srcWB, wsname_inputs)
    Dim dstWS As Worksheet
    Set dstWS = srcWB.Worksheets(wsname_inputs)
    ' clear
    dstWS.Cells.ClearContents
    ' headers
    Dim dstStartRow As Integer: dstStartRow = 1
    Dim dstStartCol As Integer: dstStartCol = 1

    With dstWS
        .Cells(dstStartRow, dstStartCol).Value = inputs_TVPlant
        .Cells(dstStartRow, dstStartCol + 1).Value = inputs_ModName
        .Cells(dstStartRow, dstStartCol + 2).Value = inputs_weekcode
        .Cells(dstStartRow, dstStartCol + 3).Value = inputs_qty
        '.Cells(dstStartRow, dstStartCol + 4).Value = inputs_fy
    End With

    ' transfer
    Dim srcStartCol As Integer, srcLastCol As Integer
    Dim i As Integer
    Dim srcHeaderRow  As Integer
    Const srcDataRow As Integer = 29
    Const srcPmodRow As Integer = 28
    srcHeaderRow = IIf(TV_Plant = strFSK, 28, 33)
    srcStartCol = IIf(TV_Plant = strFSK, 3, 4)
    Dim qty

    For Each WS In srcWB.Worksheets
        If WS.Visible = xlSheetVisible And (Not
coll_invalid_wsname.contains(WS.name)) Then
            srcLastCol = GetLastCol(WS, srcHeaderRow) - 2 ' exclude last col --
"ave"
            For i = srcStartCol To srcLastCol
                With dstWS
                    qty = WS.Cells(srcDataRow, i).Value
                    If (Not IsEmpty(qty)) And (qty > 0) Then
                        .Cells(dstStartRow + 1, dstStartCol).Value = TV_Plant
'Plant
                        If TV_Plant = strFSK Then 'PMod
                            .Cells(dstStartRow + 1, dstStartCol + 1).Value =
ParserFSKPmodName(Trim$(WS.Cells(srcPmodRow, srcStartCol - 1).Value))
                        Else
                            .Cells(dstStartRow + 1, dstStartCol + 1).Value =
WS.Cells(srcPmodRow, srcStartCol - 1).Value
                        End If
                        .Cells(dstStartRow + 1, dstStartCol + 2).Value =
WS.Cells(srcHeaderRow, i).Value 'Weekcode
                        .Cells(dstStartRow + 1, dstStartCol + 3).Value = qty  '
```

```vba
qty
                              '.Cells(dstStartRow + 1, dstStartCol + 4).Value =
vbNullString   ' FY_mod placeholder, Python handles it later
                              dstStartRow = dstStartRow + 1
                      End If
                  End With
            Next i
        End If
    Next WS
End Sub

Private Sub clean_fileType()
    ''' Change file type from old excel version to new version
    Dim fso As New Scripting.FileSystemObject
    Dim ext As String: ext = GetFileExtension(srcWB.name)
    Dim newFn As String

    Const old_ext As String = "xls"
    Const new_ext As String = ".xlsx"
    Dim path As String: path = srcWB.FullName
    If ext = old_ext Then
        newFn = srcWB.path & Application.PathSeparator &
fso.GetBaseName(srcWB.name) & new_ext
        srcWB.SaveAs Filename:=newFn, FileFormat:=51
        Application.Windows(srcWB.name).Visible = True
        srcWB.Close False
        Kill path
    Else
        ' Application.Windows(srcWB.name).Visible = True
        srcWB.Close True
    End If
    Set fso = Nothing
End Sub
```

Python DataFrame Merger API

Using Python to read all preprocessed files, and clean DataFrame further;

```python
"""
this module reads all WW SMLD src file and aggregates into one file.

it has the following functionalities.
- enumerate all source files from a given folder
- clean defects dataframe
- format inputs dataframe
- dump cleaned dataframes into sqlite3 database

Author
- @ZL, 20210804
```

```python
    """

import datetime, time, os, logging, sqlite3, sys, contextlib, functools
sys.path.append('.')
import pandas as pd
from lib.date import utils
from typing import NewType, Dict, Any, Union, List, Callable
from lib import setname_mapping

logging.basicConfig(level=logging.INFO, format='%(asctime)s %(message)s')
Path      = NewType('Path', str)
DataFrame = NewType('DataFrame', pd.DataFrame)
Cursor    = NewType('Cursor', sqlite3.Cursor)

def timer(func:Callable)->Callable:
    @functools.wraps(func)
    def inner(*args:Any, **kwargs:Any)->Any:
        beg = time.perf_counter()
        rv  = func(*args, **kwargs)
        end = time.perf_counter()
        logging.info('Successed. time lapsed(s): {0:.2f}'.format(end - beg))
        return rv
    return inner

class Merger:
    def __init__(self, folder:Path, db:Path,
                 fix_defects_setname:str=None,
fix_defects_setname_mapping:Dict[Any, Any]=None,
                 fix_inputs_modelname:str=None,
fix_inputs_modelname_mapping:Dict[Any,Any]=None,
                 fix_inputs_fy:str=None, fix_inputs_fy_mapping:Dict[Any,Any]=None,
                 fix_inputs_wkcode:str=None
    ):
        """"read preprocessed source excel files(*.xlsx) from a given directory,
and dump into database

        :param folder: a folder that holds source excel files(*.xlsx)
        :type folder: Path
        :param db: database
        :type db: Path
        :param fix_defects_setname: [description], defaults to None
        :type fix_defects_setname: str, optional
        :param fix_defects_setname_mapping: [description], defaults to None
        :type fix_defects_setname_mapping: Dict[Any, Any], optional
        :param fix_inputs_modelname: [description], defaults to None
        :type fix_inputs_modelname: str, optional
        :param fix_inputs_modelname_mapping: [description], defaults to None
        :type fix_inputs_modelname_mapping: Dict[Any,Any], optional
        :param fix_inputs_fy: [description], defaults to None
        :type fix_inputs_fy: str, optional
        :param fix_inputs_fy_mapping: [description], defaults to None
        :type fix_inputs_fy_mapping: Dict[Any,Any], optional
        :param fix_inputs_wkcode: [description], defaults to None
        :type fix_inputs_wkcode: str, optional
```

```python
            """
        self._folder = folder
        self._db      = db

        self.fix_defects_setname           = fix_defects_setname
        self.fix_defects_setname_mapping   = fix_defects_setname_mapping
        self.fix_inputs_modelname          = fix_inputs_modelname
        self.fix_inputs_modelname_mapping  = fix_inputs_modelname_mapping
        self.fix_inputs_fy                 = fix_inputs_fy
        self.fix_inputs_fy_mapping         = fix_inputs_fy_mapping
        self.fix_inputs_wkcode             = fix_inputs_wkcode

        self._dfDefect:DataFrame = None
        self._dfInputs:DataFrame = None

    def _filter(self):
        """filter source files, especially excel temporary files which start with
'~$'

        :yield: excel file
        :rtype: path-like string
        """
        xl_ext:str = '.xlsx'
        tmp_xl:str = '~$'
        idns:list = ['smld', 'quality']

        for root, _, files in os.walk(self._folder):
            for file in files:
                ext = os.path.splitext(file)[-1]
                if any(idn in file.lower() for idn in idns) and not
file.startswith(tmp_xl) and xl_ext == ext:
                    logging.info(file)
                    yield os.path.abspath(os.path.join(root, file))

    @contextlib.contextmanager
    def __enter_defects_table(self, name:str, cur:Cursor, act:str)->None:
        cur.execute(
            """
            CREATE TABLE IF NOT EXISTS {0}(
                No INTEGER,
                ConfirmationWC INTEGER,
                OccurredWC INTEGER,
                source TEXT,
                OccurredDate TEXT,
                ConfirmDate TEXT,
                TVPlant TEXT,
                LineName TEXT,
                Position TEXT,
                SetName TEXT,
                ModelName TEXT,
                ModuleID TEXT,
                WeekCode INTEGER,
                SonyPN TEXT,
                SITE TEXT,
```

```python
                CellModel TEXT,
                CellID TEXT,
                NoticedSymptom TEXT,
                OSSorFFAjudgement TEXT,
                Classify TEXT,
                FY_mod TEXT,

                UNIQUE(ConfirmationWC, TVPlant, SetName, ModelName, ModuleID,
Classify) ON CONFLICT IGNORE
            );
            """.format(name))
        try:
            yield
        finally:
            logging.info(f'SQL {act} completed')


    @contextlib.contextmanager
    def __enter_inputs_table(self, name:str, cur:Cursor, act:str)->None:
        cur.execute(
            """
            CREATE TABLE IF NOT EXISTS {0}(
                TVPlant TEXT,
                ModelName TEXT,
                WeekCode INTEGER,
                QTY INTEGER,
                FY_mod TEXT,
                FY_MM TEXT,
                UNIQUE(TVPlant, ModelName, WeekCode, QTY) ON CONFLICT IGNORE
            );
            """.format(name))
        try:
            yield
        finally:
            logging.info(f'SQL {act} completed')


    def _match_fymod(self, x:str, d:Dict[Any,Any])->Union[str, None]:
        """return FY21 based on the unique character inside a give string x;

        :param x: pmod name. i.e: AG65
        :type x: str
        :param d: a dictionary contains unique character and FYxx pairs
        :type d: Dict[Any,Any]
        :return: string if found else None
        :rtype: Union[str, None]
        """
        for k in d.keys():
            if k in x:
                return d[k]
        else:
            return None


    def _clean_dfDefects(self, DataFrameList:List[DataFrame])->None:
        """clean defects DataFrame before dumping into database
```

```python
        :param DataFrameList: a list contains raw dataframe
        :type DataFrameList: List[DataFrame]
        """
        # ~ Defects
        na_col:str          = 'Module ID'
        defects_cols:int    = 21
        defects_col_date:str = 'Confirm date'
        defects_na_date  = pd.Timestamp('20210101')
        defects_str_col1 = 'Occurred date'
        defects_str_col2 = 'Confirm date'
        defects_col_FYMM = 'FY_mod'

        _na = 'na'

        new_column_names   = {
            "No"                    :    "No",
            "Confirmation W/C"      :    "ConfirmationWC",
            "Occurred W/C"          :    "OccurredWC",
            "source"                :    "source",
            "Occurred date"         :    "OccurredDate",
            "Confirm date"          :    "ConfirmDate",
            "TV Plant"              :    "TVPlant",
            "LineName Line / SBI"   :    "LineName",
            "Position"              :    "Position",
            "Set name"              :    "SetName",
            "Model name"            :    "ModelName",
            "Module ID"             :    "ModuleID",
            "Week-Code"             :    "WeekCode",
            "Sony PN"               :    "SonyPN",
            "SITE"                  :    "SITE",
            "Cell Model"            :    "CellModel",
            "Cell-ID"               :    "CellID",
            "Noticed symptom"       :    "NoticedSymptom",
            "OSS or FFA judgement"  :    "OSSorFFAjudgement",
            "Classify"              :    "Classify",
            "FY_mod"                :    "FY_mod",
        }

        defects_tv_plants = 'TVPlant'
        defects_tv_plants_mapping = {"SO\'EM" : 'SOEM'}

        df:DataFrame = pd.concat(DataFrameList, ignore_index=True, sort=False)
        df = df[pd.notnull(df[na_col])]
        df = df.iloc[:, : defects_cols]
        # ~ convert nasty DATETIME columns into TEXT, because sqlite3 does not
like them
        df[defects_str_col1] = df[defects_str_col1].astype(str)
        df[defects_str_col2] = df[defects_str_col2].astype(str)
        # ~ fill na for rest of empty values to cater sqlite3
        df.fillna(_na, inplace=True)
        if self.fix_defects_setname:
            df[self.fix_defects_setname].replace(self.fix_defects_setname_mapping,
inplace=True)
            df[defects_col_FYMM] = df[self.fix_defects_setname].apply(lambda x:
```

```python
self._match_fymod(x, self.fix_inputs_fy_mapping))
        # ~ rename all stupid headers to fit sqlite3 schema
        df.rename(columns=new_column_names, inplace=True)
        df[defects_tv_plants].replace(defects_tv_plants_mapping, inplace=True)
        self._dfDefect = df

    def _clean_dfInputs(self, DataFrameList:List[DataFrame])->None:
        """clean inputs DataFrame before dumping into database

        :param DataFrameList: a list contains raw dataframe
        :type DataFrameList: List[DataFrame]
        """
        # ~ Inputs
        inputs_Modname:str  = 'ModelName'
        inputs_weekcode:str = 'FY_MM'

        df:DataFrame = pd.concat(DataFrameList, ignore_index=True, sort=False)
        # ~ replace stupid entries: model name, input fy, input weekcode; 20210818
        if self.fix_inputs_modelname and self.fix_inputs_modelname_mapping:

df[self.fix_inputs_modelname].replace(self.fix_inputs_modelname_mapping,
inplace=True)
        if self.fix_inputs_fy and self.fix_inputs_fy_mapping:
            df[self.fix_inputs_fy] = df[inputs_Modname].apply(lambda x:
self._match_fymod(x, self.fix_inputs_fy_mapping))
        if self.fix_inputs_wkcode:
            df[inputs_weekcode] =
df[self.fix_inputs_wkcode].astype(int).apply(lambda x:
utils.wkno2ym(int(float(x)), datetime.datetime.today().year))
        self._dfInputs = df

    @timer
    def work(self)->None:
        sheet_name:str = 'inputs'
        df_defects = []
        df_inputs  = []

        for file in self._filter():
            tmp_dfDefects = pd.read_excel(file)
            tmp_dfInput   = pd.read_excel(file, sheet_name=sheet_name)
            df_defects.append(tmp_dfDefects)
            df_inputs.append(tmp_dfInput)
        self._clean_dfDefects(df_defects)
        self._clean_dfInputs(df_inputs)

        # ~ migrate to db
        sql_table_name_defects:str = 'defects'
        sql_table_name_inputs:str  = 'inputs'
        with sqlite3.connect(self._db) as conn:
            cur = conn.cursor()
            info = 'update {}'.format(sql_table_name_defects)
            with self.__enter_defects_table(sql_table_name_defects, cur, info):
                self._dfDefect.to_sql(sql_table_name_defects, conn,
if_exists='append', index=False)
```

```
            info = 'update {}'.format(sql_table_name_inputs)
            with self.__enter_inputs_table(sql_table_name_inputs, cur, info):
                self._dfInputs.to_sql(sql_table_name_inputs, conn,
if_exists='append', index=False)
```

Project structure

```
C:.
│   20210806 WW_Panel_SMLD_Database v0.02.pptx
│   main.py
│   pmod_smld.db
│   pmod_smld.xlsx
│   requirements.txt
│
├─data
│  ├─CTTI
│  │      ~$【BOE OCL_FY21 75APH_APHB (SSVE) 】 SMLD Quality Data Workbook.xlsx
│  │      【BOE OCL_FY21 75APH_APHB (SSVE) 】 SMLD Quality Data Workbook.xlsx
│  │      【CSOT OCL_FY21 75AR (SSVE) 】 SMLD Quality Data Workbook.xlsx
│  │
│  ├─FBC
│  │      【AUO OCL_FY21 AG (SSVE) 】 SMLD Quality Data W33.xlsx
│  │      【AUO OCL_FY21 AQ (SSVE) 】 SMLD Quality Data W33.xlsx
│  │      【CSOT OCL_FY21 75AG (SSVE) 】 SMLD Quality Data W33.xlsx
│  │
│  ├─FSK
│  │      FY21_Quality_Data_SSVE_week33.xlsx
│  │
│  ├─INZ
│  │      【AUO OCL_FY20 NE (SSV) 】 SMLD Quality Data Workbook.xlsx
│  │      【AUO OCL_FY20 NH (SSV) 】 SMLD Quality Data Workbook.xlsx
│  │      【AUO OCL_FY21 AG65_85 (SSVE) 】 SMLD Quality Data Workbook.xlsx
│  │      【AUO OCL_FY21 AQ (SSVE) 】 SMLD Quality Data Workbook.xlsx
│  │      【CSOT OCL_FY21 75AG (SSVE) 】 SMLD Quality Data Workbook.xlsx
│  │      【CSOT OCL_FY21 75AR (SSVE) 】 SMLD Quality Data Workbook.xlsx
│  │
│  └─SOEM
│          【AUO OCL_FY20 55, 65NH (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【AUO OCL_FY20 85NB (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【AUO OCL_FY20 85NX_NXB (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【AUO OCL_FY21 65AG (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【AUO OCL_FY21 85AG (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【AUO OCL_FY21 85AX (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【BOE OCL_FY20 75NB (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【BOE OCL_FY21 75APH (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【CSOT OCL_FY20 75NX_NXB (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【CSOT OCL_FY21 75AG (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【CSOT OCL_FY21 75AR (SSV) 】 SMLD Quality Data Workbook.xlsx
│          【CSOT OCL_FY21 75AX (SSV) 】 SMLD Quality Data Workbook.xlsx
```

```
|                【LGD OCL_FY20 75NB (SSV) 】 SMLD Quality Data Workbook.xlsx
|                【SDC OCL_FY20 75NX_NXB (SSV) 】 SMLD Quality Data Workbook.xlsx
|
├─docs
|       develop_manual.md
|       info.text
|       smld_db.drawio
|       smld_db.png
|
├─lib
|   |   core.py
|   |   query.py
|   |   setname_mapping.py
|   |   __init__.py
|   |
|   ├─date
|   |       utils.py
|   |       __init__.py
|   |
|   └─vba
|           caller.py
|           __init__.py
|
├─reports
|       pmod_smld_viz_v0.05.xlsm
|       pmod_smld_viz_v0.06.xlsm
|       pmod_smld_viz_v0.07.xlsm
|
└─test
        cleaned_headers.py
        df_replace.py
        field_names.py
```

Usage

The project itself is a well-tested console application.

Usage 1

User may write the following command to run this application.
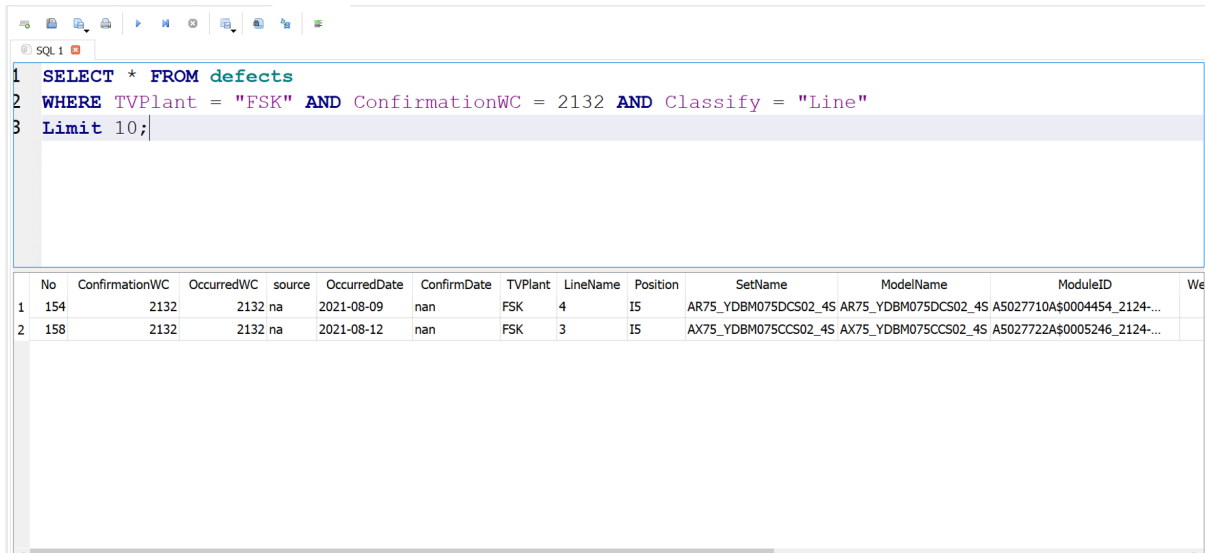
```
@echo off
cd "root of this procject directory"
python main.py
```

Usage 2

If user was familiar with `SQL`, user should use SQLite3 Studio to link "pmod_smld.db".

```sql
SELECT * FROM defects
WHERE TVPlant = "FSK" AND ConfirmationWC = 2132 AND Classify = "Line"
LIMIT 10;
```


Usage 3

If user was familar with `Excel`, user could use Excel application to manipulate data as well;

```vb
Public Sub load_src()
    ''' load source data from a given workbook @ZL, 20210825
    Dim beg As Single: beg = Timer

    Dim dstWB As Workbook: Set dstWB = ThisWorkbook
    Dim strSRC As String: strSRC = GetFilePath(ThisWorkbook.path)
    If Not FileExists(strSRC) Or strSRC = vbNullString Then
        Exit Sub
    End If
    Dim srcWB As Workbook: Set srcWB = GetObject(strSRC)

    Const wsn_defects As String = "defects"
    Const wsn_inputs As String = "inputs"

    Dim srcDefects As Worksheet, srcInputs As Worksheet
    Dim dstDefects As Worksheet, dstInputs As Worksheet
    If (Not WSExists(srcWB, wsn_defects)) Or (Not WSExists(srcWB, wsn_inputs))
  Then
        MsgBox "WSNotFoundError", vbInformation, "WorkSheetError"
        Exit Sub
    End If
```

```vba
    Set srcDefects = srcWB.Worksheets(wsn_defects)
    Set srcInputs = srcWB.Worksheets(wsn_inputs)
    Set dstDefects = dstWB.Worksheets(wsn_defects)
    Set dstInputs = dstWB.Worksheets(wsn_inputs)

    dstDefects.Cells(1, 1).Resize(srcDefects.UsedRange.Rows.Count,
srcDefects.UsedRange.Columns.Count) = srcDefects.UsedRange.Value
    dstInputs.Cells(1, 1).Resize(srcInputs.UsedRange.Rows.Count,
srcInputs.UsedRange.Columns.Count) = srcInputs.UsedRange.Value

    Const hiddenColumnDefects As String = "V:AC"
    Const hiddenColumnsInputs As String = "G:O"
    dstDefects.Columns(hiddenColumnDefects).EntireColumn.Hidden = True
    dstInputs.Columns(hiddenColumnsInputs).EntireColumn.Hidden = True

    hidde_WS_OldColumns dstWB

    MsgBox "Successed. time lapsed(s): " & (Timer - beg), vbInformation, "Reload"
End Sub
```

About

MIT License