2023-02-22_CreateCopyBooks-FLAT-v3

February 23, 2023

1 Load libraries. Requires install of 3rd party libraries Numpy, PANDAS and treelib

[1]:
```
import pandas as pd
from pathlib import Path
from treelib import Tree
import numpy as np
```

2 Read in CSV containing parsed IDMS schema source

[2]:
```
#copybook_data[ ['leading_whitespace','new_leading_whitespace'] ]
#metadata = pd.read_excel( '2022-12-15_IDMS_table_descriptions.xlsx',␣
 ↪index_col=0 )
#metadata.index.name = "table_index"
#wanted_subsystem = "DISCREPANCY PROCESSING AND ACCOUNT INFORMATION INQUIRY"
#metadata[ metadata[ wanted_subsystem ] == 1 ].copy()
```

[3]:
```
data = pd.read_csv('2023-02-14_FSA_FARMS_schema_from_source.csv', index_col=0)
#data = pd.read_excel( "2022-12-08_PLAS_IDMS_data_structure_WITH_VALID_VALUES.
 ↪xlsx", index_col=0 )
```

[4]:
```
# Start off with all lines commented out and bring them back in
# Only if they turn out to be un-redefined leaf nodes:
data['commented_out'] = True
```

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2218 entries, 0 to 12
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   table_name          2218 non-null   object
 1   indent_space_count  2218 non-null   int64
 2   data_level          2218 non-null   int64
```

```
 3   field_name          2218 non-null   object
 4   declaration_step    2218 non-null   int64
 5   end                 2218 non-null   object
 6   ELEMENT LENGTH      2154 non-null   float64
 7   POSITION            2218 non-null   int64
 8   data_type           1961 non-null   object
 9   REDEFINES           23 non-null     object
 10  VALUE               66 non-null     object
 11  OCCURS              31 non-null     object
 12  table_index         2218 non-null   int64
 13  table_vers          2218 non-null   int64
 14  BLANK ON            0 non-null      float64
 15  INDEXED BY          0 non-null      float64
 16  OLQ                 0 non-null      float64
 17  commented_out       2218 non-null   bool
dtypes: bool(1), float64(4), int64(6), object(7)
memory usage: 314.1+ KB
```

[6]:
```
data['data_level'].value_counts()
```

[6]:
```
5     1295
10     626
15     220
88      64
20      13
Name: data_level, dtype: int64
```

[7]:
```
pd.set_option( 'display.max_rows', None )
```

[8]:
```
#data.groupby( 'table_name')[ 'data_level' ].value_counts().to_frame()
```

[9]:
```
#test_df = data[ data['table_name'] == 'CLIENT' ]
```

2.1 List all table/record names in this schema

[10]:
```
data['table_name'].unique()
```

[10]:
```
array(['ACCT-DATA', 'ACQD-PROP', 'ADPS-CNTRL', 'ADVANCE', 'AID',
       'ALTMADJ', 'ALTMADJ-NEW', 'ALTMT', 'AMORTD-CST', 'APROPTN-LOOKUP',
       'ASSISTANCE', 'ASSOC-PRIN-BOND', 'CASE', 'CDEJUNC', 'CDESTR',
       'CHECKS', 'CK-CNTRL', 'CK-INFO', 'CK-INFO-FRADS', 'CLIENT',
       'CLIENT-SFSI', 'COHORT', 'CRBUR', 'CRCLAIMS', 'CRRATE',
       'CTY-LOOKUP', 'DALLOT', 'DALLOT-DTL', 'DALLOT-OBLGN', 'DALLOT-OTH',
       'DAPROC', 'DISCRP', 'DISCRP-MISC', 'DSTR-SETASD', 'DTEREC',
       'DTL-RATE', 'EASMNT', 'EQTYRVRS', 'EQUITY', 'FD-SIDE', 'FDCDE',
       'GLBORR', 'GLLNDR', 'IMCASE', 'INITMAN', 'INSRNC-AUTHY',
       'INSTALLMENT', 'INSURANCE', 'INT-ASSTNC', 'INT-BDWN', 'INVSTR-DTL',
       'INVSTR-INFO', 'INVSTR-INFO-MISC', 'JDGMT-3RD-PARTY',
```

```
       'JOB-RESTART', 'JURDCTN', 'LESSEE', 'LN-AID', 'LN-NO-INT',
       'LNRATE', 'LOAN', 'LOAN-DRE', 'LOAN-OTC', 'LOAN-SFSI',
       'LOCTN-LOOKUP', 'LSE-INFO', 'MALLOT', 'MALLOT-OBLGN', 'MALLOT-OTH',
       'MSTR-RATE', 'NOTIFY', 'NOTIFY-CNTRL', 'OBLGN', 'ORDERS',
       'ORGZTN-LOOKUP', 'OVFLO', 'PD-ACCT-RVRSL', 'PLEREP', 'PROG-SAVE',
       'PRTLSALE', 'RDA-ALTMT', 'RDA-AREA-DALLOT', 'RDA-AREA-OBLGN',
       'RDA-DALLOT-DTL', 'RDA-DALLOT-OBLGN', 'RDA-FD-SIDE',
       'RDA-INSRNC-AUTHY', 'RDA-MALLOT', 'RDA-MALLOT-OBLGN',
       'RDA-RGN-DTL', 'RDA-RGN-OBLGN', 'REJECT-TRNSCTN', 'RENTL-CNTRL',
       'RENTL-DTL', 'RENTL-FY-UNIT', 'RENTL-TTL', 'RESCHEDULE', 'RH-DFRL',
       'SITE-LOOKUP', 'SRCFDS', 'ST-LOOKUP', 'STAT', 'STOPPER', 'SUBSIDY',
       'SUBSRC', 'TRNSCTN-CNTRL', 'TRNSCTN-RVRSL', 'TRRATE', 'TRSTR',
       'USER-AUTHY', 'USER-DOMAIN', 'USER-STATCS', 'USERS'], dtype=object)
```

[11]:
```
data['data_level'].value_counts().sum()
```

[11]:
```
2218
```

[12]:
```
#data['indent_space_count'].value_counts()
```

[13]:
```
data.sample(10)
```

[13]:

	table_name	indent_space_count	data_level	field_name	\
65	GLBORR	6	15	DBT-ADJMT-CDE	
17	CK-INFO-FRADS	2	5	FY	
11	RENTL-CNTRL	2	5	FAM-UNIT-OBLGD-TTL	
25	ALTMADJ-NEW	6	15	PROG-PLANG-BDGTG-CDE	
3	ACCT-DATA	6	15	FD-CDE-2	
3	CLIENT	4	10	ID-NBR-OBLR	
5	CTY-LOOKUP	2	5	FIPS-CTY-CDE	
11	GLLNDR	4	10	LNDR-TYP	
87	GLLNDR	4	10	OL-LNS-LOSS-PD-CNT	
7	OVFLO	2	5	CARD-TYP	

	declaration_step	end	ELEMENT LENGTH	POSITION	data_type	REDEFINES	\
65	6600	DISPLAY	1.0	138	9(01)	NaN	
17	1800	DISPLAY	2.0	82	9(2)	NaN	
11	1200	COMP-3	5.0	54	S9(8)	NaN	
25	2600	DISPLAY	3.0	15	NaN	NaN	
3	400	DISPLAY	2.0	3	9(2)	NaN	
3	400	DISPLAY	10.0	6	9(10)	NaN	
5	600	DISPLAY	3.0	18	9(3)	NaN	
11	1200	DISPLAY	2.0	130	XX	NaN	
87	8800	COMP-3	3.0	535	S9(04)	NaN	
7	800	DISPLAY	1.0	76	X(01)	NaN	

	VALUE OCCURS	table_index	table_vers	BLANK ON	INDEXED BY	OLQ	\

```
65   NaN    NaN          41           1       NaN         NaN  NaN
17   NaN    NaN          18           1       NaN         NaN  NaN
11   NaN    NaN          92           1       NaN         NaN  NaN
25   NaN    NaN           6           1       NaN         NaN  NaN
3    NaN    NaN           0           1       NaN         NaN  NaN
3    NaN    NaN          19           1       NaN         NaN  NaN
5    NaN    NaN          25           1       NaN         NaN  NaN
11   NaN    NaN          42           1       NaN         NaN  NaN
87   NaN    NaN          42           1       NaN         NaN  NaN
7    NaN    NaN          75           1       NaN         NaN  NaN

    commented_out
65           True
17           True
11           True
25           True
3            True
3            True
5            True
11           True
87           True
7            True
```

3 Define functions related to flattening nested structure of data elements

[14]:
```
def TreeifyElements( grp_df : pd.DataFrame ) -> Tree:
    """Build up a tree representation of the nested data element
    structure within a single given record"""

    table_name = grp_df['table_name'].unique().squeeze()
    tree = Tree()
    tree.create_node( tag = table_name, identifier = 'root' )

    prev_data_level = 0
    prev_nodeid = 'root'
    data_level_to_parent_nodeid_dict = { 5 : 'root' }

    for row in grp_df.itertuples():

        if row.data_level > prev_data_level:
            # save this node as the parent of all nodes who have this data level
            data_level_to_parent_nodeid_dict[ row.data_level ] = prev_nodeid
        elif row.data_level < prev_data_level:
            # erase any subparents that we know will not have further children
            data_level_to_parent_nodeid_dict = \
```

```
                { level : node_id for level, node_id in␣
↪data_level_to_parent_nodeid_dict.items() if level <= row.data_level }

        parent_id = data_level_to_parent_nodeid_dict[ row.data_level ]
        tree.create_node(
            tag = row.field_name,
            identifier = row.Index,
            parent = parent_id,
            data = row.data_level
        )
        prev_nodeid = row.Index
        prev_data_level = row.data_level

    return tree
```

[15]:
```
def UncommentLeafElements( grp_df : pd.DataFrame ) -> pd.DataFrame:
    """Uses the tree structure of the elements listed within a record
    to flatten out the element structure within a record for the purposes
    of defining a logical data model.

    Elements that are "leaf nodes" within the nested element structure
    are unnested and promoted to the top level, i.e., given a data level
    of "05". Elements that are REDEFINED are removed from the model."""

    grp_df = grp_df.copy()
    tree = TreeifyElements( grp_df )

    overwrite_these = grp_df['REDEFINES'].dropna().values

    delete_these = grp_df.index[ grp_df['field_name'].isin( overwrite_these ) ]

    for delete_this_node in delete_these:
        tree.remove_node( delete_this_node )

    leaf_indices = [ _.identifier for _ in tree.leaves() ]

    # Add any non-leaf elements that have 88 valid values underneath them
    all_nodes = [ tree[ nodeid ] for nodeid in tree.expand_tree() ]
    named_value_nodes = [ _ for _ in all_nodes if _.data == 88 ]
    parent_nodes_of_named_values = set( [ tree.parent( _.identifier ).
↪identifier for _ in named_value_nodes ] )

    wanted_indices = leaf_indices + list( parent_nodes_of_named_values )

    grp_df.loc[ wanted_indices, 'commented_out' ] = False

    # not necessary since all lines start off as False
```

```
    #grp_df[ 'commented_out' ] = grp_df[ 'commented_out' ].fillna( True )

    named_value_node_indices = [ _.identifier for _ in named_value_nodes ]
    non_88_leaf_node_indices = list( set( wanted_indices ) - set(␣
↪named_value_node_indices ) )

    # print( "*" * 100 )
    # print( "wanted_indices len=", len(wanted_indices), wanted_indices )
    # print( "named_value_nodes len=", len(named_value_nodes),␣
↪named_value_nodes )
    # print( "non_88_leaf_nodes len=", len(non_88_leaf_node_indices),␣
↪non_88_leaf_node_indices )
    # print()
    # print( "before:\n",  grp_df[ 'data_level' ].value_counts() )

    grp_df.loc[ non_88_leaf_node_indices, 'data_level' ] = 5

    # print( "after:\n",  grp_df[ 'data_level' ].value_counts() )
    # print()

    return grp_df
```

[16]:
```
#test_tree = TreeifyElements( test_df )
```

[17]:
```
#mod_test_df = UncommentLeafElements( test_df )
```

[18]:
```
#mod_test_df
```

4 Define function to add a line for the record name, etc.

[19]:
```
def Create_Copybook_Parts( grp : pd.DataFrame ) -> pd.DataFrame:
    """Massages the input spreadsheet. Takes one record's worth of
    elements at a time and formats the component strings and gets them
    ready for concatenation."""

    grp = UncommentLeafElements( grp )
    table_index = grp['table_index'].iloc[0]
    table_ver = grp['table_vers'].iloc[0]
    table_n_fields = len(grp)
    table_name = grp['table_name'].iloc[0]

    print( 'table', table_index,
           'ver', table_ver,
           'n fields =', table_n_fields,
           'name =', table_name
    )
```

```
    temp_index = grp.index

    sorted_grp = grp.sort_values( 'declaration_step' )
    step_numbers = grp['declaration_step'].astype(str).str.zfill( 6 )
    comment_column = np.where( grp['commented_out'], '*', ' ' )
    #indent = 1 + ( grp['indent_space_count'].astype( int ) - 2 ) * 4
    #indent_spaces = indent.apply( lambda i: " " * i )
    indent_spaces = grp['indent_space_count'].apply( lambda i: " " * i )
    # indent_number
    sep = pd.Series( [ "  " for _ in range( len( grp ) ) ], index=temp_index )

    clauses = [ 'PIC', 'BLANK ON', 'INDEXED BY', 'OCCURS', 'OLQ', 'REDEFINES',␣
↪'VALUE' ]
    formatted_cols = {}
    for clause in clauses:
        if clause == 'PIC':
            col_name = 'data_type'
        else:
            col_name = clause
        col = grp[col_name]
        col[ col.notna() ] = col[ col.notna() ].apply( lambda t:  f'{clause}␣
↪{t}' )
        formatted_cols[ col_name ] = col

    formatted_data = dict(
        step_numbers=step_numbers,
        comment_column=comment_column,
        indent_spaces=indent_spaces,
        data_level = grp['data_level'].astype(str).str.zfill( 2 ),
        sep = sep,
        field_name = grp['field_name'],
        pic_clauses= formatted_cols[ 'data_type' ],
        comp_clause = grp[ 'end' ],
        value_clauses = formatted_cols[ 'VALUE' ],
        occurs_clauses = formatted_cols[ 'OCCURS' ],
        #redefines_clauses = formatted_cols[ 'REDEFINES' ],
        redefines_clauses = [ "" for _ in range( len( grp ) ) ],
        blank_on_clauses = formatted_cols[ 'BLANK ON' ],
        indexed_by_clauses = formatted_cols[ 'INDEXED BY' ],
        olq_clauses = formatted_cols[ 'OLQ' ],
    )

    lengths = { len(_) for _ in formatted_data.values() }
    assert len( lengths ) == 1
    assert lengths.pop() == table_n_fields
```

```
    df = pd.DataFrame( formatted_data )

    assert len( df ) == table_n_fields, f'len( df ) = {len( df )},␣
↪table_n_fields = {table_n_fields}\n\n{formatted_data}'

    table_name_line = pd.DataFrame( columns=df.columns )
    table_name_line.loc[ 0, 'step_numbers' ] = str( 50 ).zfill( 6 )
    table_name_line.loc[ 0, 'comment_column' ] = " "
    table_name_line.loc[ 0, 'indent_spaces' ] = "" # " "
    table_name_line.loc[ 0, 'data_level' ] = '01'
    table_name_line.loc[ 0, 'sep' ] = "  "
    table_name_line.loc[ 0, 'field_name' ] = table_name
    table_name_line = table_name_line.fillna( '' )

    df = pd.concat( ( table_name_line, df ), axis=0 )
    return df
```

[20]:
```
data.shape
```

[20]: (2218, 18)

[21]:
```
formatted_df = data.groupby( 'table_index' ).apply( Create_Copybook_Parts )
```

```
table 0 ver 1 n fields = 29 name = ACCT-DATA
table 1 ver 1 n fields = 114 name = ACQD-PROP
table 2 ver 1 n fields = 5 name = ADPS-CNTRL
table 3 ver 1 n fields = 6 name = ADVANCE
table 4 ver 1 n fields = 16 name = AID
table 5 ver 1 n fields = 36 name = ALTMADJ
table 6 ver 1 n fields = 37 name = ALTMADJ-NEW
table 7 ver 1 n fields = 5 name = ALTMT
table 8 ver 1 n fields = 7 name = AMORTD-CST
table 9 ver 1 n fields = 10 name = APROPTN-LOOKUP
table 10 ver 1 n fields = 12 name = ASSISTANCE
table 11 ver 1 n fields = 7 name = ASSOC-PRIN-BOND
table 12 ver 1 n fields = 6 name = CASE
table 13 ver 1 n fields = 4 name = CDEJUNC
table 14 ver 1 n fields = 24 name = CDESTR
table 15 ver 1 n fields = 9 name = CHECKS
table 16 ver 1 n fields = 8 name = CK-CNTRL
table 17 ver 1 n fields = 28 name = CK-INFO
table 18 ver 1 n fields = 28 name = CK-INFO-FRADS
table 19 ver 1 n fields = 76 name = CLIENT
table 20 ver 1 n fields = 16 name = CLIENT-SFSI
table 21 ver 1 n fields = 2 name = COHORT
table 22 ver 1 n fields = 32 name = CRBUR
table 23 ver 1 n fields = 59 name = CRCLAIMS
table 24 ver 1 n fields = 8 name = CRRATE
```

```
table 25 ver 1 n fields = 16 name = CTY-LOOKUP
table 26 ver 1 n fields = 8 name = DALLOT
table 27 ver 1 n fields = 28 name = DALLOT-DTL
table 28 ver 1 n fields = 9 name = DALLOT-OBLGN
table 29 ver 1 n fields = 1 name = DALLOT-OTH
table 30 ver 1 n fields = 3 name = DAPROC
table 31 ver 1 n fields = 48 name = DISCRP
table 32 ver 1 n fields = 6 name = DISCRP-MISC
table 33 ver 1 n fields = 39 name = DSTR-SETASD
table 34 ver 1 n fields = 7 name = DTEREC
table 35 ver 1 n fields = 12 name = DTL-RATE
table 36 ver 1 n fields = 20 name = EASMNT
table 37 ver 1 n fields = 13 name = EQTYRVRS
table 38 ver 1 n fields = 34 name = EQUITY
table 39 ver 1 n fields = 6 name = FD-SIDE
table 40 ver 1 n fields = 4 name = FDCDE
table 41 ver 1 n fields = 233 name = GLBORR
table 42 ver 1 n fields = 121 name = GLLNDR
table 43 ver 1 n fields = 6 name = IMCASE
table 44 ver 1 n fields = 39 name = INITMAN
table 45 ver 1 n fields = 15 name = INSRNC-AUTHY
table 46 ver 1 n fields = 6 name = INSTALLMENT
table 47 ver 1 n fields = 5 name = INSURANCE
table 48 ver 1 n fields = 7 name = INT-ASSTNC
table 49 ver 1 n fields = 16 name = INT-BDWN
table 50 ver 1 n fields = 29 name = INVSTR-DTL
table 51 ver 1 n fields = 24 name = INVSTR-INFO
table 52 ver 1 n fields = 3 name = INVSTR-INFO-MISC
table 53 ver 1 n fields = 9 name = JDGMT-3RD-PARTY
table 54 ver 1 n fields = 4 name = JOB-RESTART
table 55 ver 1 n fields = 4 name = JURDCTN
table 56 ver 1 n fields = 15 name = LESSEE
table 57 ver 1 n fields = 13 name = LN-AID
table 58 ver 1 n fields = 4 name = LN-NO-INT
table 59 ver 1 n fields = 5 name = LNRATE
table 60 ver 1 n fields = 45 name = LOAN
table 61 ver 1 n fields = 5 name = LOAN-DRE
table 62 ver 1 n fields = 12 name = LOAN-OTC
table 63 ver 1 n fields = 8 name = LOAN-SFSI
table 64 ver 1 n fields = 67 name = LOCTN-LOOKUP
table 65 ver 1 n fields = 32 name = LSE-INFO
table 66 ver 1 n fields = 16 name = MALLOT
table 67 ver 1 n fields = 9 name = MALLOT-OBLGN
table 68 ver 1 n fields = 1 name = MALLOT-OTH
table 69 ver 1 n fields = 11 name = MSTR-RATE
table 70 ver 1 n fields = 33 name = NOTIFY
table 71 ver 1 n fields = 3 name = NOTIFY-CNTRL
table 72 ver 1 n fields = 48 name = OBLGN
```

```
table 73 ver 1 n fields = 6 name = ORDERS
table 74 ver 1 n fields = 21 name = ORGZTN-LOOKUP
table 75 ver 1 n fields = 9 name = OVFLO
table 76 ver 1 n fields = 41 name = PD-ACCT-RVRSL
table 77 ver 1 n fields = 63 name = PLEREP
table 78 ver 1 n fields = 1 name = PROG-SAVE
table 79 ver 1 n fields = 14 name = PRTLSALE
table 80 ver 1 n fields = 5 name = RDA-ALTMT
table 81 ver 1 n fields = 24 name = RDA-AREA-DALLOT
table 82 ver 1 n fields = 10 name = RDA-AREA-OBLGN
table 83 ver 1 n fields = 24 name = RDA-DALLOT-DTL
table 84 ver 1 n fields = 10 name = RDA-DALLOT-OBLGN
table 85 ver 1 n fields = 6 name = RDA-FD-SIDE
table 86 ver 1 n fields = 16 name = RDA-INSRNC-AUTHY
table 87 ver 1 n fields = 19 name = RDA-MALLOT
table 88 ver 1 n fields = 10 name = RDA-MALLOT-OBLGN
table 89 ver 1 n fields = 14 name = RDA-RGN-DTL
table 90 ver 1 n fields = 16 name = RDA-RGN-OBLGN
table 91 ver 1 n fields = 4 name = REJECT-TRNSCTN
table 92 ver 1 n fields = 14 name = RENTL-CNTRL
table 93 ver 1 n fields = 23 name = RENTL-DTL
table 94 ver 1 n fields = 3 name = RENTL-FY-UNIT
table 95 ver 1 n fields = 10 name = RENTL-TTL
table 96 ver 1 n fields = 4 name = RESCHEDULE
table 97 ver 1 n fields = 22 name = RH-DFRL
table 98 ver 1 n fields = 21 name = SITE-LOOKUP
table 99 ver 1 n fields = 2 name = SRCFDS
table 100 ver 1 n fields = 8 name = ST-LOOKUP
table 101 ver 1 n fields = 9 name = STAT
table 102 ver 1 n fields = 1 name = STOPPER
table 103 ver 1 n fields = 21 name = SUBSIDY
table 104 ver 1 n fields = 3 name = SUBSRC
table 105 ver 1 n fields = 2 name = TRNSCTN-CNTRL
table 106 ver 1 n fields = 28 name = TRNSCTN-RVRSL
table 107 ver 1 n fields = 5 name = TRRATE
table 108 ver 1 n fields = 6 name = TRSTR
table 109 ver 1 n fields = 4 name = USER-AUTHY
table 110 ver 1 n fields = 6 name = USER-DOMAIN
table 111 ver 1 n fields = 14 name = USER-STATCS
table 112 ver 1 n fields = 13 name = USERS
```

[22]:
```
formatted_df.shape
```

[22]:
```
(2331, 14)
```

[23]:
```
formatted_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
MultiIndex: 2331 entries, (0, 0) to (112, 12)
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   step_numbers        2331 non-null   object
 1   comment_column      2331 non-null   object
 2   indent_spaces       2331 non-null   object
 3   data_level          2331 non-null   object
 4   sep                 2331 non-null   object
 5   field_name          2331 non-null   object
 6   pic_clauses         2074 non-null   object
 7   comp_clause         2331 non-null   object
 8   value_clauses       179 non-null    object
 9   occurs_clauses      144 non-null    object
 10  redefines_clauses   2331 non-null   object
 11  blank_on_clauses    113 non-null    object
 12  indexed_by_clauses  113 non-null    object
 13  olq_clauses         113 non-null    object
dtypes: object(14)
memory usage: 276.7+ KB
```

5 Begin concatenating copybook syntax components

[24]:
```
#formatted_df
```

[25]:
```
formatted_df['first_part'] = \
    formatted_df['step_numbers'] + \
    formatted_df['comment_column'] + \
    formatted_df['indent_spaces'] + \
    formatted_df['data_level'] + \
    formatted_df['sep'] + \
    formatted_df['field_name']
```

[26]:
```
#formatted_df
```

6 Add enough space between element name and PICTURE clause to right justify text

[27]:
```
formatted_df['first_part_len'] = formatted_df['first_part'].apply( len )
```

[28]:
```
#formatted_df['first_part_len'].hist()
```

[29]:
```
(formatted_df['first_part_len'] >= 49).sum()
```

[29]: 0

[30]:
```
#formatted_df['first_part_len'].describe()
```

[31]:
```
#long_rows = formatted_df[ formatted_df['first_part_len'] >= 49].index
```

[32]:
```
#formatted_df.loc[long_rows, 'first_part_len'].describe()
```

[33]:
```
#formatted_df['first_part'] = formatted_df['first_part'].str.ljust(50)
```

[34]:
```
formatted_df['white_space_middle_part'] = formatted_df['first_part_len'].apply(␣
 ↪lambda l: (" " * (50 - l)) if l < 49 else " " )
```

[35]:
```
formatted_df['white_space_middle_part'].values
```

[35]:
```
array(['                                ',
       '                                ',
       '                                ', …, '                                ',
       '                    ', '                                '],
      dtype=object)
```

7 Add copybook syntax after element name, including the following clauses:

- "OCCURS" and "REDEFINES" clauses go before PIC clause
- "VALUE" clause is either in lieu of or after PIC clause
- COMP-3 goes after PIC clause

[36]:
```
formatted_df = formatted_df.fillna("")
```

[37]:
```
formatted_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 2331 entries, (0, 0) to (112, 12)
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   step_numbers            2331 non-null   object
 1   comment_column          2331 non-null   object
 2   indent_spaces           2331 non-null   object
 3   data_level              2331 non-null   object
 4   sep                     2331 non-null   object
 5   field_name              2331 non-null   object
 6   pic_clauses             2331 non-null   object
 7   comp_clause             2331 non-null   object
 8   value_clauses           2331 non-null   object
 9   occurs_clauses          2331 non-null   object
 10  redefines_clauses       2331 non-null   object
 11  blank_on_clauses        2331 non-null   object
 12  indexed_by_clauses      2331 non-null   object
```

```
 13  olq_clauses              2331 non-null   object
 14  first_part               2331 non-null   object
 15  first_part_len           2331 non-null   int64
 16  white_space_middle_part  2331 non-null   object
dtypes: int64(1), object(16)
memory usage: 331.4+ KB
```

[38]:
```
( (formatted_df['occurs_clauses'] != "") & (formatted_df['redefines_clauses'] !
 ↪= "") ).sum()
```

[38]:
```
0
```

[39]:
```
#formatted_df['redefines_clauses_len'] = formatted_df['redefines_clauses'].
 ↪apply( len )
```

[40]:
```
#formatted_df['redefines_clauses_len'].describe()
```

[41]:
```
#formatted_df['occurs_clauses_len'] = formatted_df['occurs_clauses'].apply( len␣
 ↪)
```

[42]:
```
#formatted_df['occurs_clauses_len'].describe()
```

[43]:
```
# Add a separating space to the pre-pic clauses when you have both
#formatted_df.loc[ (formatted_df['occurs_clauses'] != "") &␣
 ↪(formatted_df['redefines_clauses'] != ""), 'occurs_clauses'] = \
#    formatted_df.loc[ (formatted_df['occurs_clauses'] != "") &␣
 ↪(formatted_df['redefines_clauses'] != ""), 'occurs_clauses'].apply( lambda s:
 ↪ " " + s )
```

[44]:
```
formatted_df['pre_pic_clause'] = formatted_df['redefines_clauses'] +␣
 ↪formatted_df['occurs_clauses']
```

7.1 Remove any big clumps of whitespace that make lines unnecessarily wide

[45]:
```
formatted_df[ 'pre_pic_clause' ] = formatted_df[ 'pre_pic_clause' ].str.
 ↪replace( pat=r'\s+', repl=' ', regex=True )
```

[46]:
```
formatted_df['second_part'] = formatted_df['pre_pic_clause']
```

8 Diagnostic: show me some examples where a data element has an OCCURS clause or a REDEFINES CLAUSE

[47]:
```
formatted_df.loc[ (formatted_df['occurs_clauses'] != "") |␣
 ↪(formatted_df['redefines_clauses'] != ""), 'second_part'].values[:50]
```

[47]:
```
array(['OCCURS 0 TO 10 TIMES DEPENDING ON NBR-OF-OCCURS',
       'OCCURS 0 TO 10 TIMES DEPENDING ON NBR-OF-OCCURS',
```

```
'OCCURS 2 TIMES', 'OCCURS 10 TIMES', 'OCCURS 4 TIMES',
'OCCURS 4 TIMES', 'OCCURS 2 TIMES', 'OCCURS 16 TIMES',
'OCCURS 4 TIMES', 'OCCURS 10 TIMES', 'OCCURS 4 TIMES',
'OCCURS 6 TIMES', 'OCCURS 6 TIMES', 'OCCURS 3 TIMES',
'OCCURS 3 TIMES', 'OCCURS 3 TIMES',
'OCCURS 0 TO 1800 TIMES DEPENDING ON NBR-OF-OCCURS',
'OCCURS 5 TIMES', 'OCCURS 6 TIMES', 'OCCURS 6 TIMES',
'OCCURS 2 TIMES', 'OCCURS 4 TIMES', 'OCCURS 8 TIMES',
'OCCURS 4 TIMES', 'OCCURS 10 TIMES', 'OCCURS 10 TIMES',
'OCCURS 0 TO 1850 TIMES DEPENDING ON MULTI-CARD-DATA-LGTH',
'OCCURS 3 TIMES', 'OCCURS 4 TIMES', 'OCCURS 200 TIMES',
'OCCURS 2 TIMES'], dtype=object)
```

[48]:
```
#formatted_df.loc[ (formatted_df['occurs_clauses'] != "") &␣
↪(formatted_df['redefines_clauses'] != ""), 'first_and_second_part'].values[:
↪50]
```

[49]:
```
formatted_df['second_part_len'] = formatted_df['second_part'].apply( len )
```

[50]:
```
formatted_df['second_part_len'].describe()
```

[50]:
```
count    2331.000000
mean        0.250536
std         2.562999
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max        56.000000
Name: second_part_len, dtype: float64
```

[51]:
```
pd.set_option( 'display.max_colwidth', None )
```

[52]:
```
( formatted_df['pre_pic_clause'] != "" ).sum()
```

[52]:
```
31
```

[53]:
```
# Add a separating space to the pic clauses when you have a pre-pic clause
formatted_df.loc[ (formatted_df['pre_pic_clause'] != "") &␣
↪(formatted_df['pic_clauses'] != ""), 'pic_clauses'] = \
   formatted_df.loc[ (formatted_df['pre_pic_clause'] != "") &␣
↪(formatted_df['pic_clauses'] != ""), 'pic_clauses'].apply( lambda s: " " + s␣
↪)
```

[54]:
```
formatted_df['second_and_third_part'] = formatted_df['second_part'] +␣
↪formatted_df['pic_clauses']
```

9 Trailing clauses are COMP and VALUE

[55]:
```
#formatted_df['comp_clause']
```

[56]:
```
formatted_df['comp_clause'].value_counts()
```

[56]:
```
DISPLAY           1502
COMP-3             647
                   113
CONDITION-NAME      64
COMP                 5
Name: comp_clause, dtype: int64
```

[57]:
```
formatted_df.loc[ (formatted_df['comp_clause'] == 'DISPLAY'), 'comp_clause' ] =␣
↪''
```

[58]:
```
formatted_df.loc[ (formatted_df['comp_clause'] == 'COND'), 'comp_clause' ] = ''
```

[59]:
```
formatted_df.loc[ (formatted_df['comp_clause'] == 'CONDITION-NAME'),␣
↪'comp_clause' ] = ''
```

[60]:
```
formatted_df['value_clauses'].value_counts()
```

[60]:
```
               2265
VALUE 'S'        10
VALUE 'D'        10
VALUE 'C'        10
VALUE 'T'         8
VALUE 'U'         8
VALUE 'V'         8
VALUE '0'         4
VALUE '1'         4
VALUE SPACE       2
VALUE 'R'         1
VALUE 'A'         1
Name: value_clauses, dtype: int64
```

[61]:
```
( (formatted_df['comp_clause'] != "") & (formatted_df['value_clauses'] != "") ).
↪sum()
```

[61]:
```
0
```

[62]:
```
# Add a separating space to the post-pic clauses when you have both
formatted_df.loc[ (formatted_df['comp_clause'] != "") &␣
↪(formatted_df['value_clauses'] != ""), 'value_clauses'] = \
   formatted_df.loc[ (formatted_df['comp_clause'] != "") &␣
↪(formatted_df['value_clauses'] != ""), 'value_clauses'].apply( lambda s: " "␣
↪+ s )
```

[63]:
```
formatted_df['post_pic_clauses'] = formatted_df['comp_clause'] +␣
↪formatted_df['value_clauses']
```

[64]:
```
# Add a separating space to the pic clauses when you have a post-pic clause
formatted_df.loc[ (formatted_df['post_pic_clauses'] != "") &␣
↪((formatted_df['pre_pic_clause'] != "") | (formatted_df['pic_clauses'] !=␣
↪"")), 'post_pic_clauses'] = \
   formatted_df.loc[ (formatted_df['post_pic_clauses'] != "") &␣
↪((formatted_df['pre_pic_clause'] != "") | (formatted_df['pic_clauses'] !=␣
↪"")), 'post_pic_clauses'].apply( lambda s: " " + s )
```

[65]:
```
#formatted_df.loc[formatted_df['post_pic_clauses'] != "", 'post_pic_clauses'].
↪values
```

[66]:
```
formatted_df[ 'second_and_third_part' ].values[:50]
```

[66]:
```
array(['', 'PIC 9(2)', '', '', 'PIC 9(2)', 'PIC 9(1)', 'PIC 9(1)',
       'PIC 9(2)', 'PIC 9(2)V9(4)', 'PIC V9(6)', 'PIC 9(1)', 'PIC 9(1)',
       'PIC 9(06)', '', 'PIC 9(1)', 'PIC 9(1)', 'PIC 9(3)', 'PIC 9(2)',
       'PIC X(1)', 'PIC 9(06)', 'PIC 9(3)', 'PIC S9(8)V99', 'PIC X(01)',
       'PIC 9(1)', 'PIC 9(6)', 'PIC 9(01)', 'PIC 9(1)', 'PIC 9(2)V9(4)',
       'PIC V9(6)', 'PIC X(0019)', '', '', '', 'PIC 9(2)', 'PIC 9(1)',
       'PIC 9(1)', 'PIC X(07)', 'PIC X(07)', '', 'PIC 9(2)', 'PIC 9(3)',
       'PIC 9(5)', '', 'PIC X(1)', '', '', '', '', '', ''], dtype=object)
```

[67]:
```
formatted_df[ 'second_third_and_forth_part'] = formatted_df[␣
↪'second_and_third_part' ] + formatted_df['post_pic_clauses']
```

[68]:
```
formatted_df[ 'second_third_and_forth_part'].sample(50).values
```

[68]:
```
array(['PIC 9(5)', 'PIC 9(1)', 'PIC S9(9)V99 COMP-3', 'PIC 9(2)',
       'PIC 9(2)', 'PIC 9', 'PIC 9(3)', '', 'PIC S9(08)V99 COMP-3',
       'PIC 9(2)', '', 'PIC X(01)', 'PIC S9(04) COMP-3', 'PIC 9(1)', '',
       'PIC 9(2)', 'PIC 9(06)', 'PIC 9(06)', 'PIC 9(2)', 'PIC XX',
       'PIC X(19)', 'PIC X(19)', 'PIC 9(3)', 'PIC 9(02)', '', 'PIC 9(6)',
       '', 'PIC 9(2)', 'PIC S9(08) COMP', 'PIC S9(10)V99 COMP-3',
       'PIC 9(3)', 'PIC 9(2)', 'PIC 9(06)', 'PIC 9', 'PIC 9(2)',
       'PIC S9(8) COMP-3', 'PIC 9(5)', 'PIC S9(7)V99 COMP-3', 'PIC 9(02)',
       'PIC 9(1)', 'PIC S9(11)V99 COMP-3', '', '', 'PIC S9(05) COMP-3',
       'PIC 9(2)', 'PIC 9(3)', '', 'PIC S9(06) COMP-3', '', 'PIC 9(02)'],
      dtype=object)
```

[69]:
```
formatted_df[ 'second_third_and_forth_part_len'] = formatted_df[␣
↪'second_third_and_forth_part' ].apply( len )
```

[70]:
```
formatted_df[ 'second_third_and_forth_part_len'].describe()
```

[70]:
```
count    2331.000000
mean       10.506650
std         6.630099
min         0.000000
25%         8.000000
50%         9.000000
75%        17.000000
max        65.000000
Name: second_third_and_forth_part_len, dtype: float64
```

[71]:
```
formatted_df['content_len'] = formatted_df['first_part_len'] + formatted_df[␣
↪'second_third_and_forth_part_len'] + 2 # separating space + period
```

10 Diagnostic: show me the widest lines, because they're not supposed to be wider than 72 characters

- IMPORTANT NOTE: YOU MUST GO IN AND HAND EDIT THESE LINES IN THE OUTPUT COPYBOOK TO BREAK THEM APART INTO MULTIPLE LINES

[72]:
```
formatted_df['content_len'].sort_values().tail(30)
```

[72]:
```
table_index
26           3      60
51           10     60
77           3      60
26           2      60
41           81     60
77           6      60
             24     60
             11     60
26           1      60
41           82     61
10           5      61
48           3      61
16           3      61
11           3      61
48           2      61
83           22     62
41           165    62
77           9      62
             8      62
96           1      62
41           187    62
27           26     62
81           22     62
1            35     62
11           4      62
```

```
37          11     71
5           16     76
6           16     82
44          38     91
91          3      97
Name: content_len, dtype: int64
```

[73]:
```
formatted_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 2331 entries, (0, 0) to (112, 12)
Data columns (total 25 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   step_numbers                     2331 non-null   object
 1   comment_column                   2331 non-null   object
 2   indent_spaces                    2331 non-null   object
 3   data_level                       2331 non-null   object
 4   sep                              2331 non-null   object
 5   field_name                       2331 non-null   object
 6   pic_clauses                      2331 non-null   object
 7   comp_clause                      2331 non-null   object
 8   value_clauses                    2331 non-null   object
 9   occurs_clauses                   2331 non-null   object
 10  redefines_clauses                2331 non-null   object
 11  blank_on_clauses                 2331 non-null   object
 12  indexed_by_clauses               2331 non-null   object
 13  olq_clauses                      2331 non-null   object
 14  first_part                       2331 non-null   object
 15  first_part_len                   2331 non-null   int64
 16  white_space_middle_part          2331 non-null   object
 17  pre_pic_clause                   2331 non-null   object
 18  second_part                      2331 non-null   object
 19  second_part_len                  2331 non-null   int64
 20  second_and_third_part            2331 non-null   object
 21  post_pic_clauses                 2331 non-null   object
 22  second_third_and_forth_part      2331 non-null   object
 23  second_third_and_forth_part_len  2331 non-null   int64
 24  content_len                      2331 non-null   int64
dtypes: int64(4), object(21)
memory usage: 477.1+ KB
```

10.1 If the line is too wide, cut down on the whitespace in the middle for right justifying the PIC clauses, etc.

[74]:
```
formatted_df['alternative_white_space_middle_part'] = [ " " * ( 72 - _ ) for _␣
↪in formatted_df['content_len'].values ]
```

[75]:
```
formatted_df['alternative_white_space_middle_part_len'] =␣
↪formatted_df['alternative_white_space_middle_part'].apply( len )
```

[76]:
```
formatted_df['alternative_white_space_middle_part_len'].describe()
```

[76]:
```
count    2331.000000
mean       31.470184
std         9.770251
min         0.000000
25%        24.000000
50%        32.000000
75%        38.000000
max        56.000000
Name: alternative_white_space_middle_part_len, dtype: float64
```

[77]:
```
formatted_df.loc[ formatted_df['alternative_white_space_middle_part'] == "",␣
↪'alternative_white_space_middle_part' ] = " "
```

[78]:
```
formatted_df['alternative_white_space_middle_part_len'] =␣
↪formatted_df['alternative_white_space_middle_part'].apply( len )
```

[79]:
```
formatted_df['alternative_white_space_middle_part_len'].describe()
```

[79]:
```
count    2331.000000
mean       31.471900
std         9.764807
min         1.000000
25%        24.000000
50%        32.000000
75%        38.000000
max        56.000000
Name: alternative_white_space_middle_part_len, dtype: float64
```

[80]:
```
#formatted_df['alternative_white_space_middle_part_len']
```

[81]:
```
formatted_df['white_space_middle_part_len'] =␣
↪formatted_df['white_space_middle_part'].apply( len )
```

[82]:
```
(~(formatted_df['white_space_middle_part_len'] <␣
↪formatted_df['alternative_white_space_middle_part_len'])).sum()
```

[82]:
```
363
```

[83]:
```
formatted_df['white_space_middle_part'] =␣
 ↪formatted_df['white_space_middle_part'].where(
    cond = formatted_df['white_space_middle_part_len'] <␣
 ↪formatted_df['alternative_white_space_middle_part_len'],
    other = formatted_df['alternative_white_space_middle_part']
)
```

[84]:
```
(formatted_df[ 'second_third_and_forth_part_len'] == 0).sum()
```

[84]:
```
293
```

11 Add the period to the end of each line and we're done!

[85]:
```
formatted_df[ 'line_completion' ] = '.'
```

[86]:
```
formatted_df[ 'line_completion' ] = formatted_df[ 'line_completion' ].where(
    cond = formatted_df[ 'second_third_and_forth_part' ] == "",
    other = formatted_df['white_space_middle_part'] + formatted_df[␣
 ↪'second_third_and_forth_part' ] + '.'
)
```

[87]:
```
formatted_df[ 'full_line'] = formatted_df['first_part'] + formatted_df[␣
 ↪'line_completion' ]
```

[88]:
```
formatted_df[ 'full_line_len' ] = formatted_df[ 'full_line'].apply(len)
```

[89]:
```
formatted_df[ 'full_line_len' ].describe()
```

[89]:
```
count    2331.000000
mean       58.208494
std        13.494025
min        15.000000
25%        59.000000
50%        60.000000
75%        68.000000
max        97.000000
Name: full_line_len, dtype: float64
```

[90]:
```
#formatted_df[ 'full_line_len' ].sort_values()
```

[91]:
```
#formatted_df.sort_values('full_line_len' )['full_line'].tail(50)
```

12 Dump the lines to a Copybook text file and give to Samee

[92]:
```
formatted_df['full_line'].
  ↪to_csv("2023-02-21_IDMS_Copybooks_FARMS_FLAT_STRUCTURE.txt", header=False,␣
  ↪index=False )
```

[93]:
```
!head 2023-02-21_IDMS_Copybooks_FARMS_FLAT_STRUCTURE.txt
```

```
000050 01  ACCT-DATA.
000100    05  LN-NBR                                PIC 9(2).
000200*   05  FD-CDE.
000300*     10  FD-CDE-3.
000400         05  FD-CDE-2                         PIC 9(2).
000500         05  FD-CDE-3RD                       PIC 9(1).
000600       05  FD-CDE-4TH                         PIC 9(1).
000700    05  KIND-CDE-LN                           PIC 9(2).
000800*   05  INT-RATE-NOTE                         PIC 9(2)V9(4).
000900    05  INT-RATE-NOTE-1ST                     PIC V9(6).
```