jmec2

May 27, 2024

[ ]:
```python
import numpy as np
import pandas as pd
```

[ ]:
```python
ex = pd.read_excel('./  .xlsx', header=None)

display(ex.head(20))
print(len(ex))
```

```
           0        1
0   [  (1)]       1
1               1
2                 1
3                  1
4        NaN      NaN
5   [  (2)]       2
6               2
7                 2
8                  2
9        NaN       2
10       NaN       2
11       NaN      NaN
12  [  (3)]       3
13            3.1
14              3.1
15               3.1
16       NaN      NaN
17            3.2
18              3.2
19               3.2
```

```
31
```

[ ]:
```python
def parse_jmec(
    keys: list[str], values: list[str], primary_key: str, section_num: int
):

    # <-- INPUT VALIDATION -->
    if len(keys) != len(values):
        raise ValueError("Keys and values must have the same length")
    # <-- MAIN PROCESS -->
    #
    clean_data: list[tuple[str, str]] = []
    str_list: list[str] = []
    for i in range(len(keys)):

        reversed_idx = len(keys) - i - 1

        # keys values
        if keys[reversed_idx] is np.nan and values[reversed_idx] is np.nan:
            continue
        # keys  values     values
        elif keys[reversed_idx] is np.nan and values[reversed_idx] is not np.
↪nan:
            str_list.append(values[reversed_idx])
        # keys   values
        elif keys[reversed_idx] is not np.nan and values[reversed_idx] is np.
↪nan:
            raise Exception('index     value   ')
        # keys values       str_list
        elif keys[reversed_idx] is not np.nan and values[reversed_idx] is not␣
↪np.nan:
            str_list.append(values[reversed_idx])
            str_list.reverse()
            string = ''.join(str_list)
            str_list = []
            clean_data.append([keys[reversed_idx], string])

    clean_data.reverse()
    print("DEBUG: clean_data")
    print(clean_data, end='\n\n')

    # primary_key idx
    pkey_indices = []
    for i in range(len(clean_data)):
        key = clean_data[i][0]
        if key.startswith(primary_key):
            pkey_indices.append(i)

    pkey_indices.append(len(clean_data))

    print("DEBUG: pkey_indices")
    print(pkey_indices, end='\n\n')

    # primary_key
    sectoin_cnts = []
    for i in range(1, len(pkey_indices)):
        prev = pkey_indices[i-1]
        curr = pkey_indices[i]

        cnt = (curr - prev - 1) // section_num
        sectoin_cnts.append(cnt)

    print("DEBUG: sectoin_cnts")
    print(sectoin_cnts, end='\n\n')

    #
    ans = []
    for i in range(len(pkey_indices) - 1): # end   -1
        for j in range(sectoin_cnts[i]):

            # 1  pkey
            ans_row = [clean_data[pkey_indices[i]][1]]

            for k in range(section_num):

                #
                idx = pkey_indices[i] + section_num * j + k + 1
                ans_row.append(clean_data[idx][1])

            ans.append(ans_row)

    print("DEBUG: ans")
    print(ans, end='\n\n')

    return ans


ans = parse_jmec(ex[0].tolist(), ex[1].tolist(), '[  ', 3)
```

```
DEBUG: clean_data
[['[  (1)]', ' 1'], ['   ', '  1'], ['  ', ' 1'], [' ', ' 1'],
['[  (2)]', ' 2'], ['   ', '  2'], ['  ', ' 2'], [' ', ' 2 2 2'],
['[  (3)]', ' 3'], ['   ', '  3.1'], ['  ', ' 3.1'], [' ', ' 3.1'],
['   ', '  3.2'], ['  ', ' 3.2'], [' ', ' 3.2'], ['[  (4)]', ' 4'],
['   ', '  4.1'], ['  ', ' 4.1'], [' ', ' 4.1 4.1'], ['   ',
'  4.2'], ['  ', ' 4.2'], [' ', ' 4.2 4.2']]

DEBUG: pkey_indices
[0, 4, 8, 15, 22]

DEBUG: sectoin_cnts
[1, 1, 2, 2]
```

```
DEBUG: ans
[[' 1', '  1', ' 1', ' 1'], [' 2', '  2', ' 2', ' 2 2 2'], [' 3',
'  3.1', ' 3.1', ' 3.1'], [' 3', '  3.2', ' 3.2', ' 3.2'], [' 4',
'  4.1', ' 4.1', ' 4.1 4.1'], [' 4', '  4.2', ' 4.2', ' 4.2 4.2']]
```

[ ]:
```
df = pd.DataFrame(ans)
display(df)
```

```
     0        1      2           3
0    1        1      1           1
1    2        2      2     2 2 2
2    3     3.1    3.1         3.1
3    3     3.2    3.2         3.2
4    4     4.1    4.1     4.1 4.1
5    4     4.2    4.2     4.2 4.2
```

[ ]:
```

```