Analytical Oil Splitting Analysis

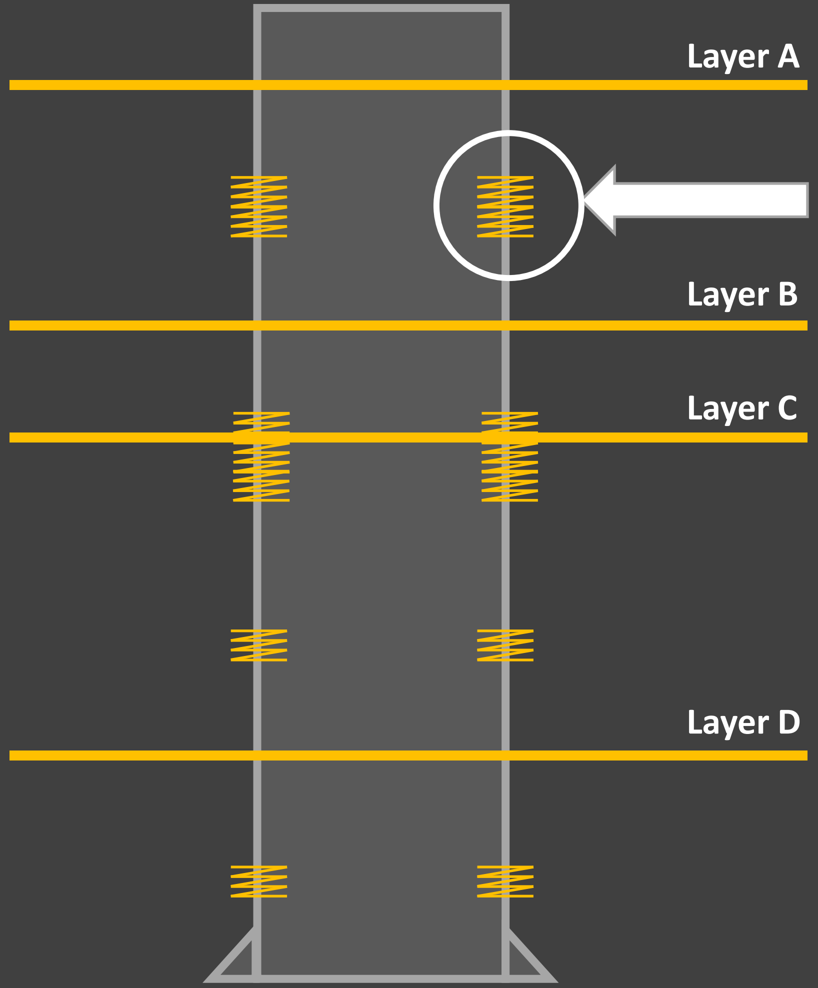
Grha Gandana P

Petroleum Engineer/ Data Analyst

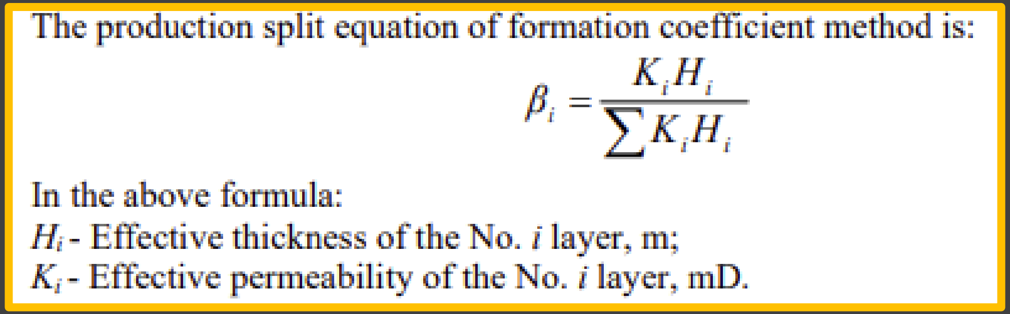
Preface

“During the oil field exploitation, the current status of reservoir exploitation guides adjustment exploitation plan. In the multilayer reservoir, we can take effective layering adjustment measures and exploiting residual reserves measures in multilayer reservoir. Therefore, production split is the significant method for multilayer reservoir exact exploitation. The common production split method is using KH method as the weighting factor.“

Problem Statement



The problem is:
If the oil production in a day was 1000 BOPD (barrel oil per day) and the perforation was opened in layer A,B,C,D. Then how much the oil produced from each layer?



The production split equation of formation coefficient method is:

$$\beta_i = \frac{K_i H_i}{\sum K_i H_i}$$

In the above formula:
H_i - Effective thickness of the No. *i* layer, m;
K_i - Effective permeability of the No. *i* layer, mD.

Objective

“The objective is to create an analytical oil splitting for multilayer reservoir. In this case, the splitting is conducted in the “X” Field which has large datasets, consists of 1900 wells and has 28 layers of reservoir. Each reservoir has low quality reservoir and high quality reservoir. In this splitting our target is only the low quality reservoir.

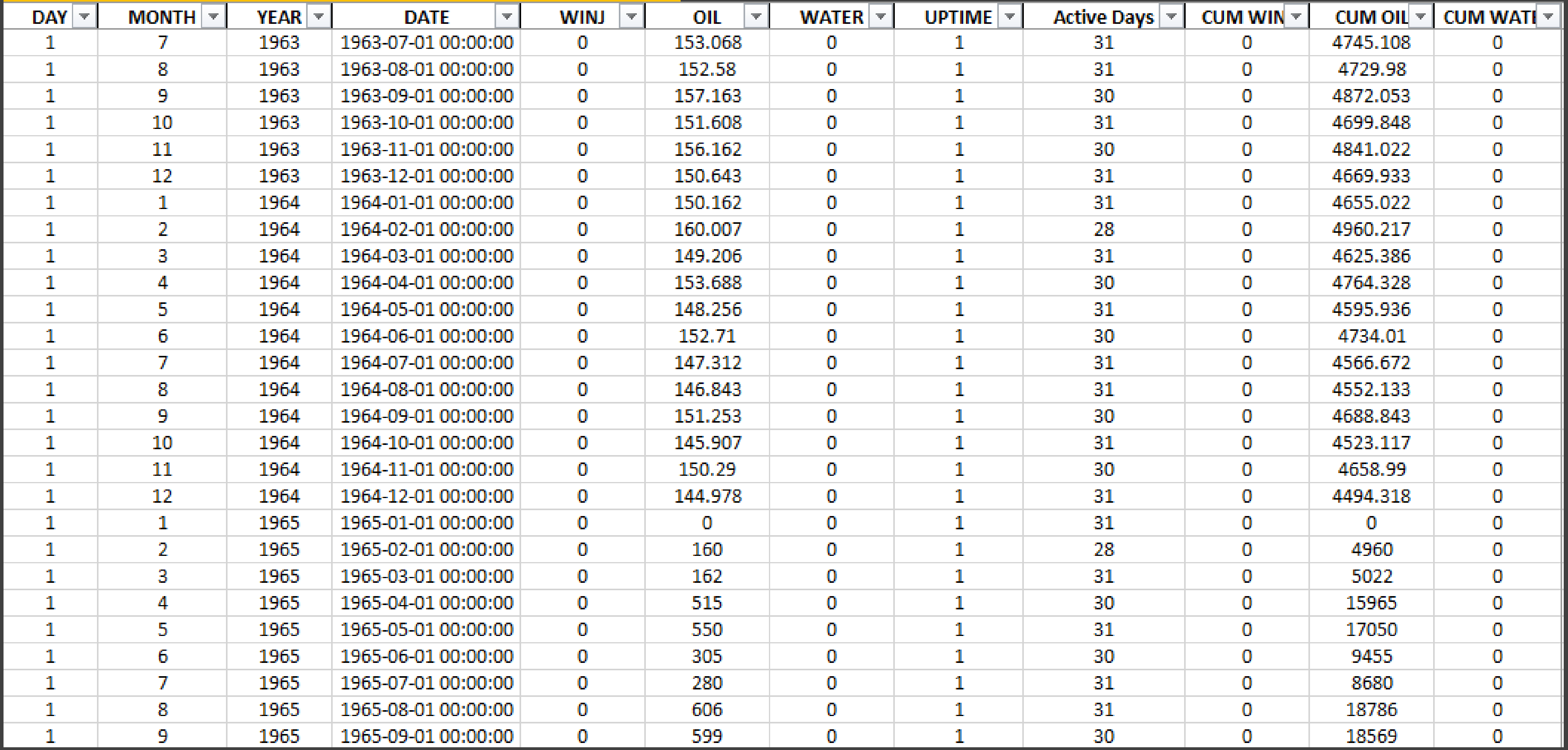
Commonly, this splitting conducted by more than 4 persons and take around two weeks even more which is make it ineffective and inefficient. To make the splitting process more effective and efficient, the splitting machine is conducted using the integration of Python and Microsoft Excel.“

Data Preparation

- Data collection (production data, marker data, completion data, and lumping data)
- Data cleaning (handling missing data and removing fault data)
- Data structuring (rearrange columns and adding columns)

Data Collection

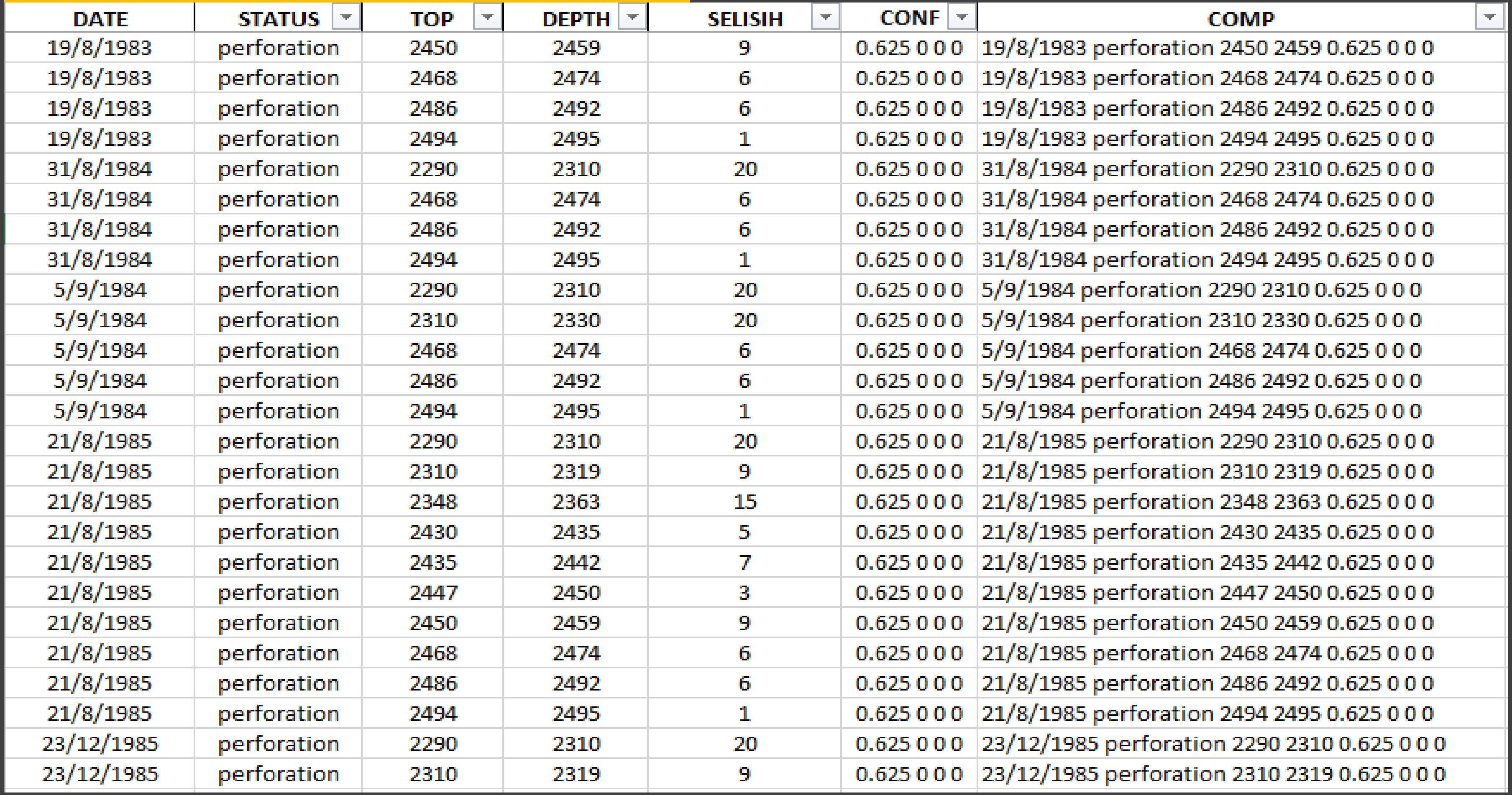
Production Data (around 600k rows)



DAY	MONTH	YEAR	DATE	WINJ	OIL	WATER	UPTIME	Active Days	CUM WIN	CUM OIL	CUM WATI
1	7	1963	1963-07-01 00:00:00	0	153.068	0	1	31	0	4745.108	0
1	8	1963	1963-08-01 00:00:00	0	152.58	0	1	31	0	4729.98	0
1	9	1963	1963-09-01 00:00:00	0	157.163	0	1	30	0	4872.053	0
1	10	1963	1963-10-01 00:00:00	0	151.608	0	1	31	0	4699.848	0
1	11	1963	1963-11-01 00:00:00	0	156.162	0	1	30	0	4841.022	0
1	12	1963	1963-12-01 00:00:00	0	150.643	0	1	31	0	4669.933	0
1	1	1964	1964-01-01 00:00:00	0	150.162	0	1	31	0	4655.022	0
1	2	1964	1964-02-01 00:00:00	0	160.007	0	1	28	0	4960.217	0
1	3	1964	1964-03-01 00:00:00	0	149.206	0	1	31	0	4625.386	0
1	4	1964	1964-04-01 00:00:00	0	153.688	0	1	30	0	4764.328	0
1	5	1964	1964-05-01 00:00:00	0	148.256	0	1	31	0	4595.936	0
1	6	1964	1964-06-01 00:00:00	0	152.71	0	1	30	0	4734.01	0
1	7	1964	1964-07-01 00:00:00	0	147.312	0	1	31	0	4566.672	0
1	8	1964	1964-08-01 00:00:00	0	146.843	0	1	31	0	4552.133	0
1	9	1964	1964-09-01 00:00:00	0	151.253	0	1	30	0	4688.843	0
1	10	1964	1964-10-01 00:00:00	0	145.907	0	1	31	0	4523.117	0
1	11	1964	1964-11-01 00:00:00	0	150.29	0	1	30	0	4658.99	0
1	12	1964	1964-12-01 00:00:00	0	144.978	0	1	31	0	4494.318	0
1	1	1965	1965-01-01 00:00:00	0	0	0	1	31	0	0	0
1	2	1965	1965-02-01 00:00:00	0	160	0	1	28	0	4960	0
1	3	1965	1965-03-01 00:00:00	0	162	0	1	31	0	5022	0
1	4	1965	1965-04-01 00:00:00	0	515	0	1	30	0	15965	0
1	5	1965	1965-05-01 00:00:00	0	550	0	1	31	0	17050	0
1	6	1965	1965-06-01 00:00:00	0	305	0	1	30	0	9455	0
1	7	1965	1965-07-01 00:00:00	0	280	0	1	31	0	8680	0
1	8	1965	1965-08-01 00:00:00	0	606	0	1	31	0	18786	0
1	9	1965	1965-09-01 00:00:00	0	599	0	1	30	0	18569	0

Data Collection

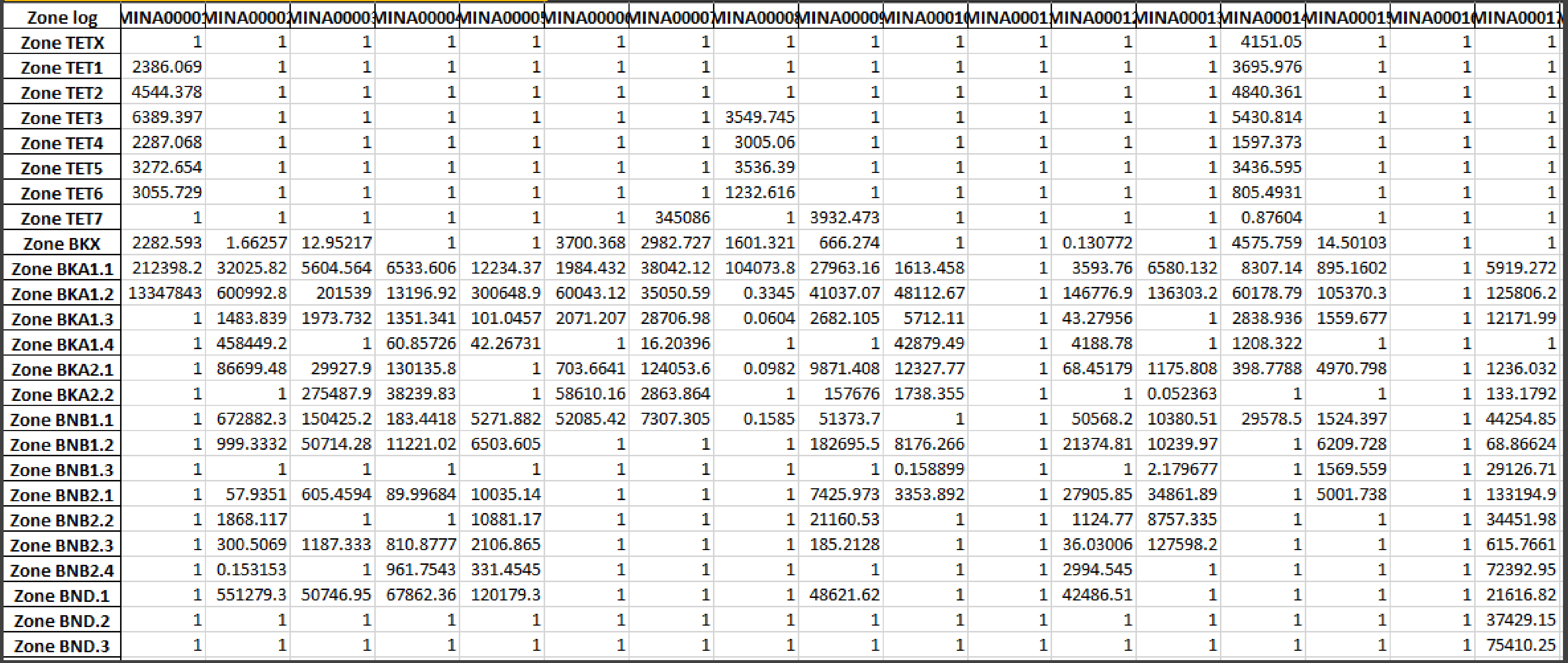
Completion Data (around 200k rows)



DATE	STATUS	TOP	DEPTH	SELISIH	CONF	COMP
19/8/1983	perforation	2450	2459	9	0.625 0 0 0	19/8/1983 perforation 2450 2459 0.625 0 0 0
19/8/1983	perforation	2468	2474	6	0.625 0 0 0	19/8/1983 perforation 2468 2474 0.625 0 0 0
19/8/1983	perforation	2486	2492	6	0.625 0 0 0	19/8/1983 perforation 2486 2492 0.625 0 0 0
19/8/1983	perforation	2494	2495	1	0.625 0 0 0	19/8/1983 perforation 2494 2495 0.625 0 0 0
31/8/1984	perforation	2290	2310	20	0.625 0 0 0	31/8/1984 perforation 2290 2310 0.625 0 0 0
31/8/1984	perforation	2468	2474	6	0.625 0 0 0	31/8/1984 perforation 2468 2474 0.625 0 0 0
31/8/1984	perforation	2486	2492	6	0.625 0 0 0	31/8/1984 perforation 2486 2492 0.625 0 0 0
31/8/1984	perforation	2494	2495	1	0.625 0 0 0	31/8/1984 perforation 2494 2495 0.625 0 0 0
5/9/1984	perforation	2290	2310	20	0.625 0 0 0	5/9/1984 perforation 2290 2310 0.625 0 0 0
5/9/1984	perforation	2310	2330	20	0.625 0 0 0	5/9/1984 perforation 2310 2330 0.625 0 0 0
5/9/1984	perforation	2468	2474	6	0.625 0 0 0	5/9/1984 perforation 2468 2474 0.625 0 0 0
5/9/1984	perforation	2486	2492	6	0.625 0 0 0	5/9/1984 perforation 2486 2492 0.625 0 0 0
5/9/1984	perforation	2494	2495	1	0.625 0 0 0	5/9/1984 perforation 2494 2495 0.625 0 0 0
21/8/1985	perforation	2290	2310	20	0.625 0 0 0	21/8/1985 perforation 2290 2310 0.625 0 0 0
21/8/1985	perforation	2310	2319	9	0.625 0 0 0	21/8/1985 perforation 2310 2319 0.625 0 0 0
21/8/1985	perforation	2348	2363	15	0.625 0 0 0	21/8/1985 perforation 2348 2363 0.625 0 0 0
21/8/1985	perforation	2430	2435	5	0.625 0 0 0	21/8/1985 perforation 2430 2435 0.625 0 0 0
21/8/1985	perforation	2435	2442	7	0.625 0 0 0	21/8/1985 perforation 2435 2442 0.625 0 0 0
21/8/1985	perforation	2447	2450	3	0.625 0 0 0	21/8/1985 perforation 2447 2450 0.625 0 0 0
21/8/1985	perforation	2450	2459	9	0.625 0 0 0	21/8/1985 perforation 2450 2459 0.625 0 0 0
21/8/1985	perforation	2468	2474	6	0.625 0 0 0	21/8/1985 perforation 2468 2474 0.625 0 0 0
21/8/1985	perforation	2486	2492	6	0.625 0 0 0	21/8/1985 perforation 2486 2492 0.625 0 0 0
21/8/1985	perforation	2494	2495	1	0.625 0 0 0	21/8/1985 perforation 2494 2495 0.625 0 0 0
23/12/1985	perforation	2290	2310	20	0.625 0 0 0	23/12/1985 perforation 2290 2310 0.625 0 0 0
23/12/1985	perforation	2310	2319	9	0.625 0 0 0	23/12/1985 perforation 2310 2319 0.625 0 0 0

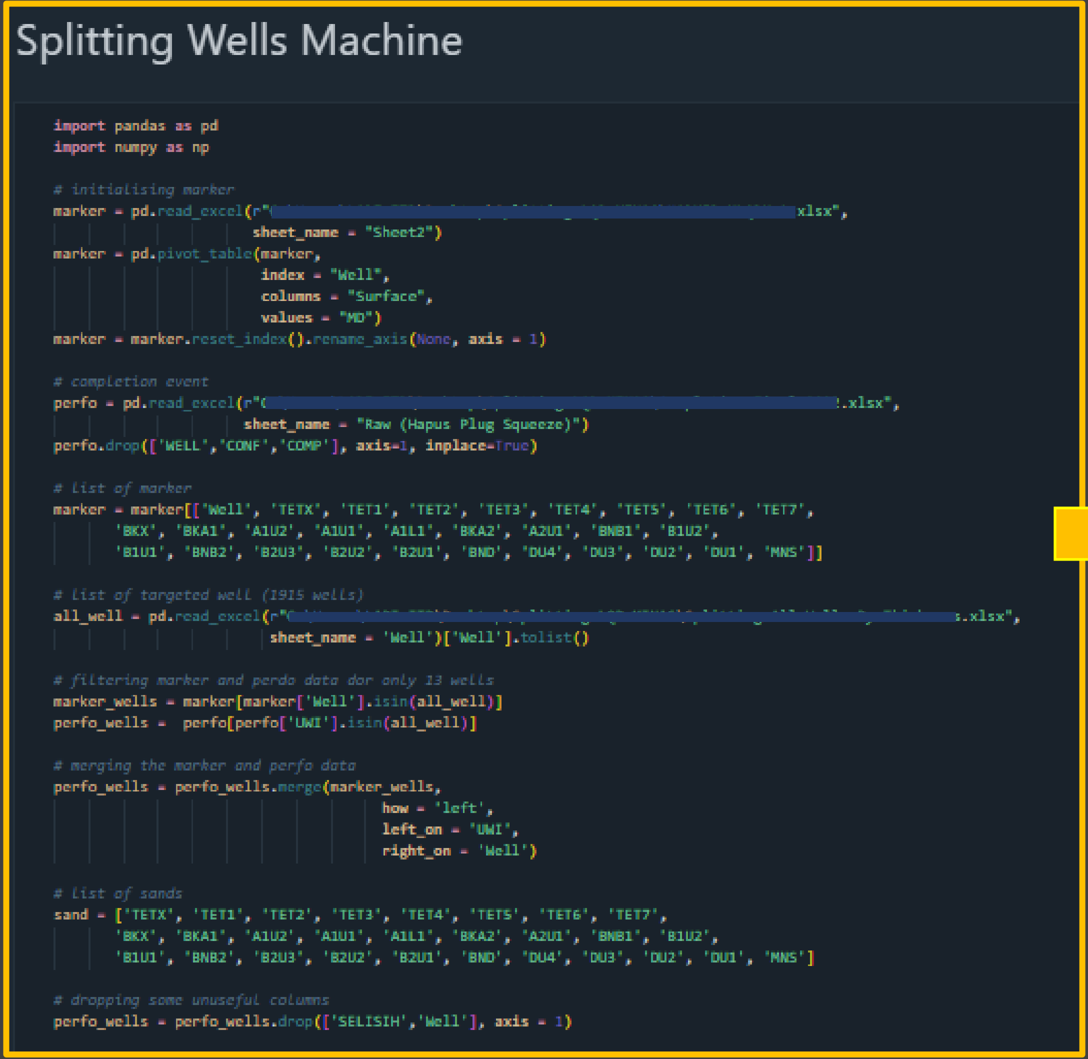
Data Collection

Lumping Data



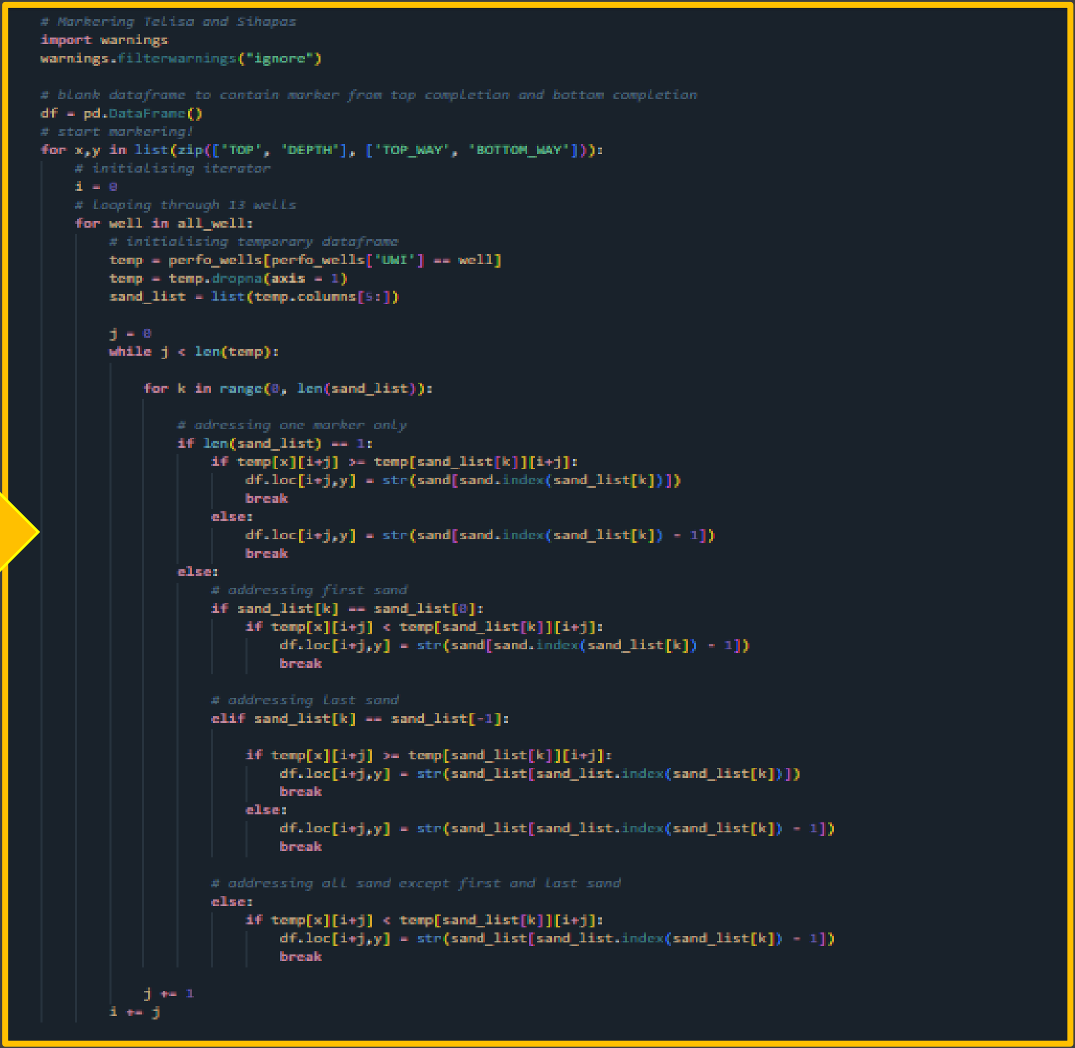
Zone log	MINA00001	MINA00002	MINA00003	MINA00004	MINA00005	MINA00006	MINA00007	MINA00008	MINA00009	MINA00010	MINA00011	MINA00012	MINA00013	MINA00014	MINA00015	MINA00016	MINA00017
Zone TETX	1	1	1	1	1	1	1	1	1	1	1	1	1	4151.05	1	1	1
Zone TET1	2386.069	1	1	1	1	1	1	1	1	1	1	1	1	3695.976	1	1	1
Zone TET2	4544.378	1	1	1	1	1	1	1	1	1	1	1	1	4840.361	1	1	1
Zone TET3	6389.397	1	1	1	1	1	1	3549.745	1	1	1	1	1	5430.814	1	1	1
Zone TET4	2287.068	1	1	1	1	1	1	3005.06	1	1	1	1	1	1597.373	1	1	1
Zone TET5	3272.654	1	1	1	1	1	1	3536.39	1	1	1	1	1	3436.595	1	1	1
Zone TET6	3055.729	1	1	1	1	1	1	1232.616	1	1	1	1	1	805.4931	1	1	1
Zone TET7	1	1	1	1	1	1	345086	1	3932.473	1	1	1	1	0.87604	1	1	1
Zone BKX	2282.593	1.66257	12.95217	1	1	3700.368	2982.727	1601.321	666.274	1	1	0.130772	1	4575.759	14.50103	1	1
Zone BKA1.1	212398.2	32025.82	5604.564	6533.606	12234.37	1984.432	38042.12	104073.8	27963.16	1613.458	1	3593.76	6580.132	8307.14	895.1602	1	5919.272
Zone BKA1.2	13347843	600992.8	201539	13196.92	300648.9	60043.12	35050.59	0.3345	41037.07	48112.67	1	146776.9	136303.2	60178.79	105370.3	1	125806.2
Zone BKA1.3	1	1483.839	1973.732	1351.341	101.0457	2071.207	28706.98	0.0604	2682.105	5712.11	1	43.27956	1	2838.936	1559.677	1	12171.99
Zone BKA1.4	1	458449.2	1	60.85726	42.26731	1	16.20396	1	1	42879.49	1	4188.78	1	1208.322	1	1	1
Zone BKA2.1	1	86699.48	29927.9	130135.8	1	703.6641	124053.6	0.0982	9871.408	12327.77	1	68.45179	1175.808	398.7788	4970.798	1	1236.032
Zone BKA2.2	1	1	275487.9	38239.83	1	58610.16	2863.864	1	157676	1738.355	1	1	0.052363	1	1	1	133.1792
Zone BNB1.1	1	672882.3	150425.2	183.4418	5271.882	52085.42	7307.305	0.1585	51373.7	1	1	50568.2	10380.51	29578.5	1524.397	1	44254.85
Zone BNB1.2	1	999.3332	50714.28	11221.02	6503.605	1	1	1	182695.5	8176.266	1	21374.81	10239.97	1	6209.728	1	68.86624
Zone BNB1.3	1	1	1	1	1	1	1	1	1	0.158899	1	1	2.179677	1	1569.559	1	29126.71
Zone BNB2.1	1	57.9351	605.4594	89.99684	10035.14	1	1	1	7425.973	3353.892	1	27905.85	34861.89	1	5001.738	1	133194.9
Zone BNB2.2	1	1868.117	1	1	10881.17	1	1	1	21160.53	1	1	1124.77	8757.335	1	1	1	34451.98
Zone BNB2.3	1	300.5069	1187.333	810.8777	2106.865	1	1	1	185.2128	1	1	36.03006	127598.2	1	1	1	615.7661
Zone BNB2.4	1	0.153153	1	961.7543	331.4545	1	1	1	1	1	1	2994.545	1	1	1	1	72392.95
Zone BND.1	1	551279.3	50746.95	67862.36	120179.3	1	1	1	48621.62	1	1	42486.51	1	1	1	1	21616.82
Zone BND.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	37429.15
Zone BND.3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	75410.25

Code Time! (1)



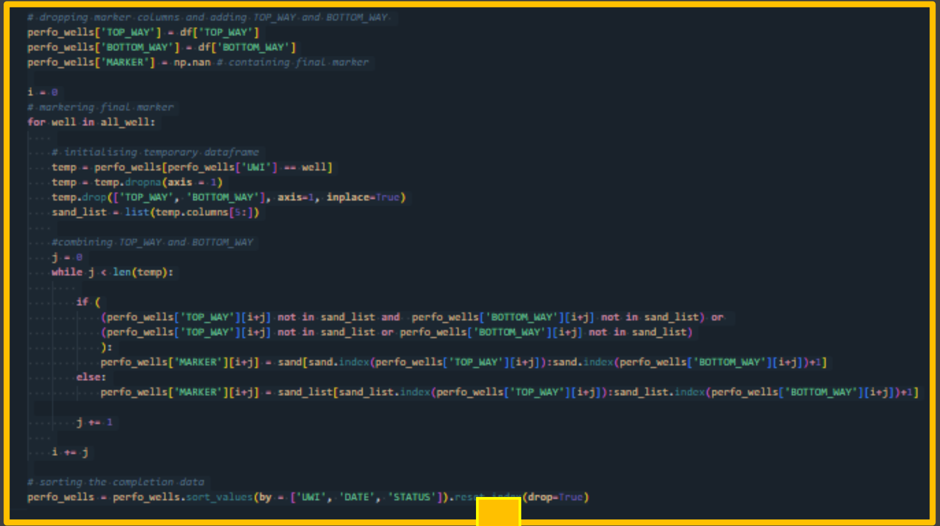
Splitting Wells Machine

```python
import pandas as pd
import numpy as np

# initialising marker
marker = pd.read_excel(r"[redacted]xlsx",
                       sheet_name = "Sheet2")
marker = pd.pivot_table(marker,
                        index = "Well",
                        columns = "Surface",
                        values = "MD")
marker = marker.reset_index().rename_axis(None, axis = 1)

# completion event
perfo = pd.read_excel(r"[redacted].xlsx",
                      sheet_name = "Raw (Hapus Plug Squeeze)")
perfo.drop(['WELL','CONF','COMP'], axis=1, inplace=True)

# list of marker
marker = marker[['Well', 'TETX', 'TET1', 'TET2', 'TET3', 'TET4', 'TET5', 'TET6', 'TET7',
        'BKX', 'BKA1', 'A1U2', 'A1U1', 'A1L1', 'BKA2', 'A2U1', 'BNB1', 'B1U2',
        'B1U1', 'BNB2', 'B2U3', 'B2U2', 'B2U1', 'BND', 'DU4', 'DU3', 'DU2', 'DU1', 'MNS']]

# list of targeted well (1915 wells)
all_well = pd.read_excel(r"[redacted].xlsx",
                         sheet_name = 'Well')['Well'].tolist()

# filtering marker and perdo data dor only 13 wells
marker_wells = marker[marker['Well'].isin(all_well)]
perfo_wells =  perfo[perfo['UWI'].isin(all_well)]

# merging the marker and perfo data
perfo_wells = perfo_wells.merge(marker_wells,
                                how = 'left',
                                left_on = 'UWI',
                                right_on = 'Well')

# list of sands
sand = ['TETX', 'TET1', 'TET2', 'TET3', 'TET4', 'TET5', 'TET6', 'TET7',
        'BKX', 'BKA1', 'A1U2', 'A1U1', 'A1L1', 'BKA2', 'A2U1', 'BNB1', 'B1U2',
        'B1U1', 'BNB2', 'B2U3', 'B2U2', 'B2U1', 'BND', 'DU4', 'DU3', 'DU2', 'DU1', 'MNS']

# dropping some unuseful columns
perfo_wells = perfo_wells.drop(['SELISIH','Well'], axis = 1)
```

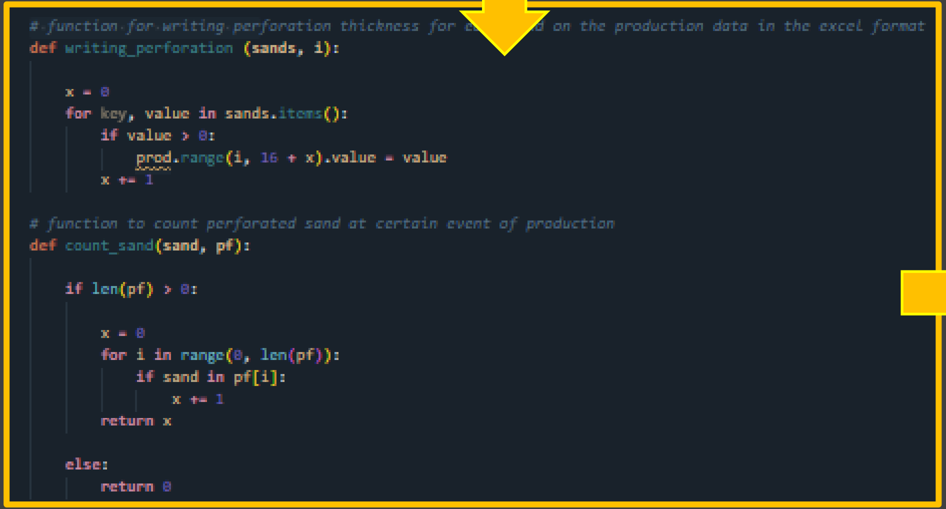


```python
# Markering Telisa and Sihapas
import warnings
warnings.filterwarnings("ignore")

# blank dataframe to contain marker from top completion and bottom completion
df = pd.DataFrame()
# start markering!
for x,y in list(zip(['TOP', 'DEPTH'], ['TOP_WAY', 'BOTTOM_WAY'])):
    # initialising iterator
    i = 0
    # looping through 13 wells
    for well in all_well:
        # initialising temporary dataframe
        temp = perfo_wells[perfo_wells['UWI'] == well]
        temp = temp.dropna(axis = 1)
        sand_list = list(temp.columns[5:])

        j = 0
        while j < len(temp):

            for k in range(0, len(sand_list)):

                # adressing one marker only
                if len(sand_list) == 1:
                    if temp[x][i+j] >= temp[sand_list[k]][i+j]:
                        df.loc[i+j,y] = str(sand[sand.index(sand_list[k])])
                        break
                    else:
                        df.loc[i+j,y] = str(sand[sand.index(sand_list[k]) - 1])
                        break
                else:
                    # addressing first sand
                    if sand_list[k] == sand_list[0]:
                        if temp[x][i+j] < temp[sand_list[k]][i+j]:
                            df.loc[i+j,y] = str(sand[sand.index(sand_list[k]) - 1])
                            break

                    # addressing last sand
                    elif sand_list[k] == sand_list[-1]:

                        if temp[x][i+j] >= temp[sand_list[k]][i+j]:
                            df.loc[i+j,y] = str(sand_list[sand_list.index(sand_list[k])])
                            break
                        else:
                            df.loc[i+j,y] = str(sand_list[sand_list.index(sand_list[k]) - 1])
                            break

                    # addressing all sand except first and last sand
                    else:
                        if temp[x][i+j] < temp[sand_list[k]][i+j]:
                            df.loc[i+j,y] = str(sand_list[sand_list.index(sand_list[k]) - 1])
                            break

            j += 1
        i += j
```

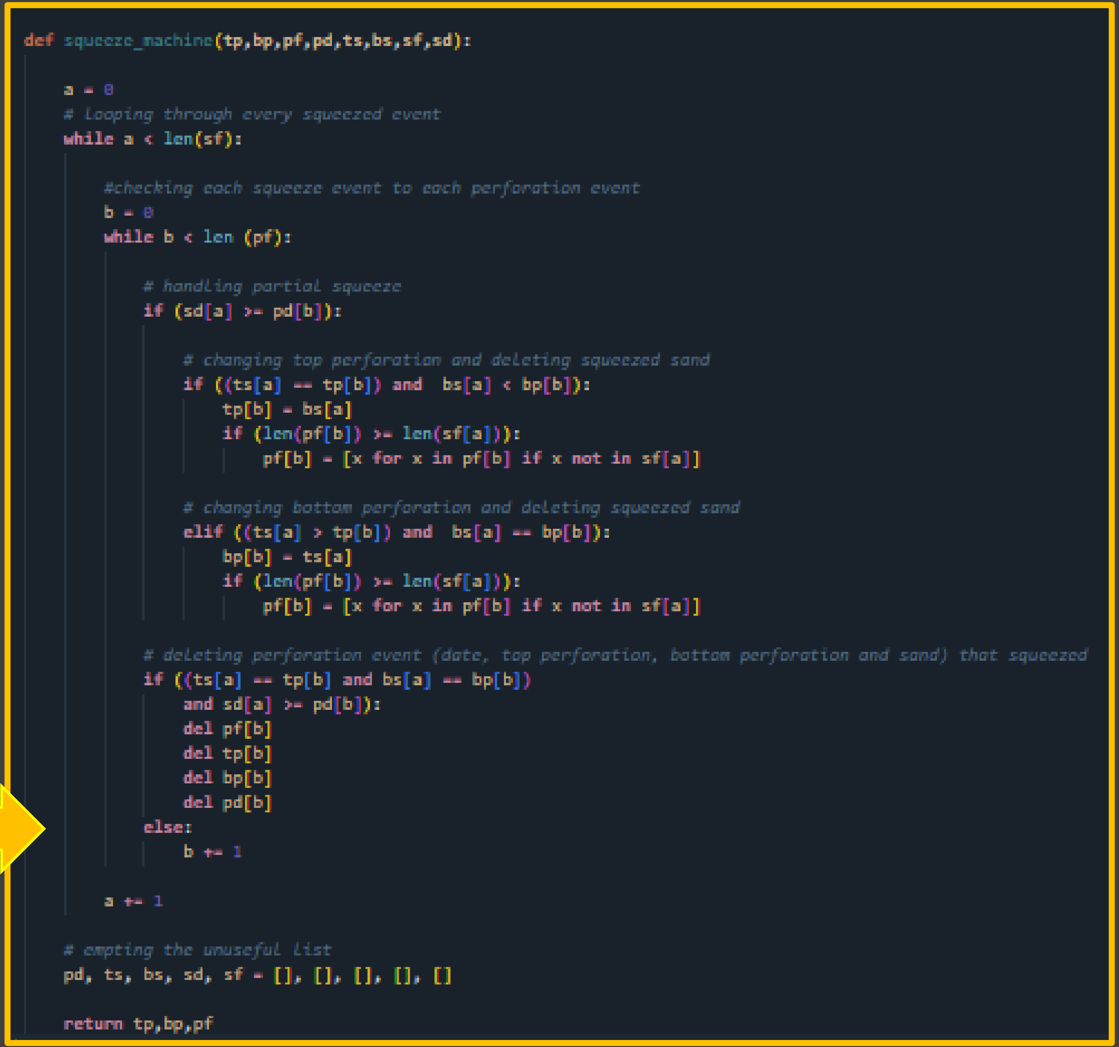
Code Time! (2)



```python
# dropping marker columns and adding TOP_WAY and BOTTOM_WAY
perfo_wells['TOP_WAY'] = df['TOP_WAY']
perfo_wells['BOTTOM_WAY'] = df['BOTTOM_WAY']
perfo_wells['MARKER'] = np.nan # containing final marker

i = 0
# markering final marker
for well in all_well:

    # initialising temporary dataframe
    temp = perfo_wells[perfo_wells['UWI'] == well]
    temp = temp.dropna(axis = 1)
    temp.drop(['TOP_WAY', 'BOTTOM_WAY'], axis=1, inplace=True)
    sand_list = list(temp.columns[5:])

    #combining TOP_WAY and BOTTOM_WAY
    j = 0
    while j < len(temp):

        if (
            (perfo_wells['TOP_WAY'][i+j] not in sand_list and  perfo_wells['BOTTOM_WAY'][i+j] not in sand_list) or
            (perfo_wells['TOP_WAY'][i+j] not in sand_list or perfo_wells['BOTTOM_WAY'][i+j] not in sand_list)
            ):
            perfo_wells['MARKER'][i+j] = sand[sand.index(perfo_wells['TOP_WAY'][i+j]):sand.index(perfo_wells['BOTTOM_WAY'][i+j])+1]
        else:
            perfo_wells['MARKER'][i+j] = sand_list[sand_list.index(perfo_wells['TOP_WAY'][i+j]):sand_list.index(perfo_wells['BOTTOM_WAY'][i+j])+1]

        j += 1

    i += j

# sorting the completion data
perfo_wells = perfo_wells.sort_values(by = ['UWI', 'DATE', 'STATUS']).reset_index(drop=True)
```

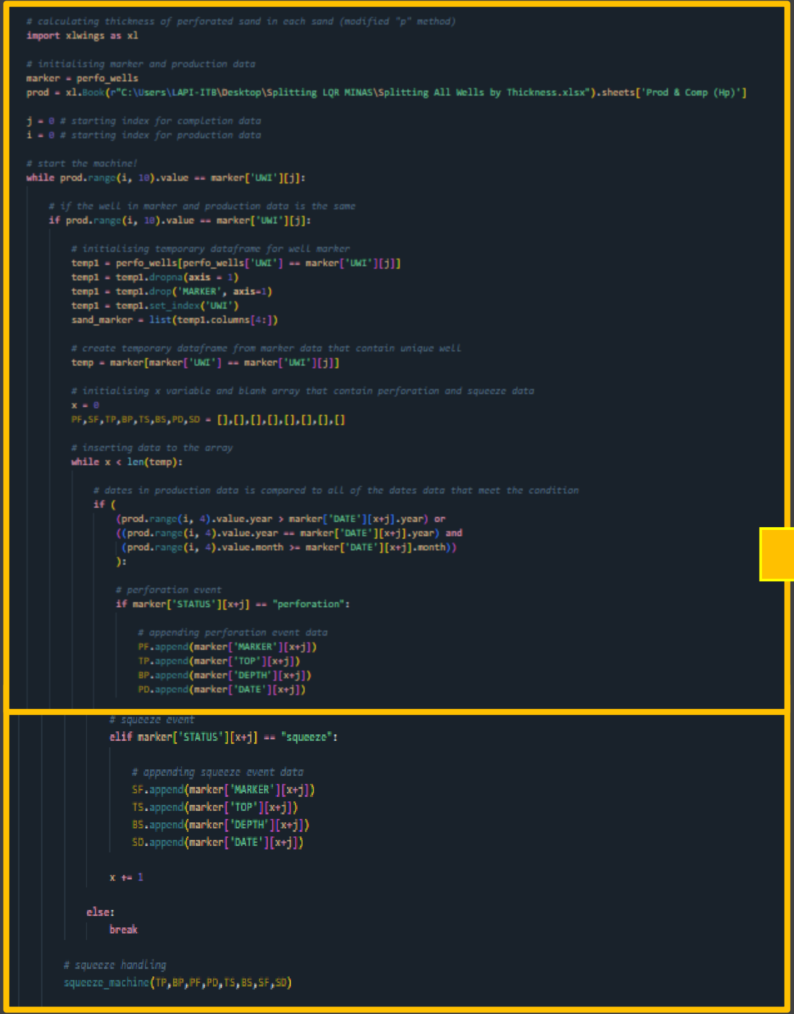


```python
# function for writing perforation thickness for e[illegible]d on the production data in the excel format
def writing_perforation (sands, i):

    x = 0
    for key, value in sands.items():
        if value > 0:
            prod.range(i, 16 + x).value = value
        x += 1

# function to count perforated sand at certain event of production
def count_sand(sand, pf):

    if len(pf) > 0:

        x = 0
        for i in range(0, len(pf)):
            if sand in pf[i]:
                x += 1
        return x

    else:
        return 0
```

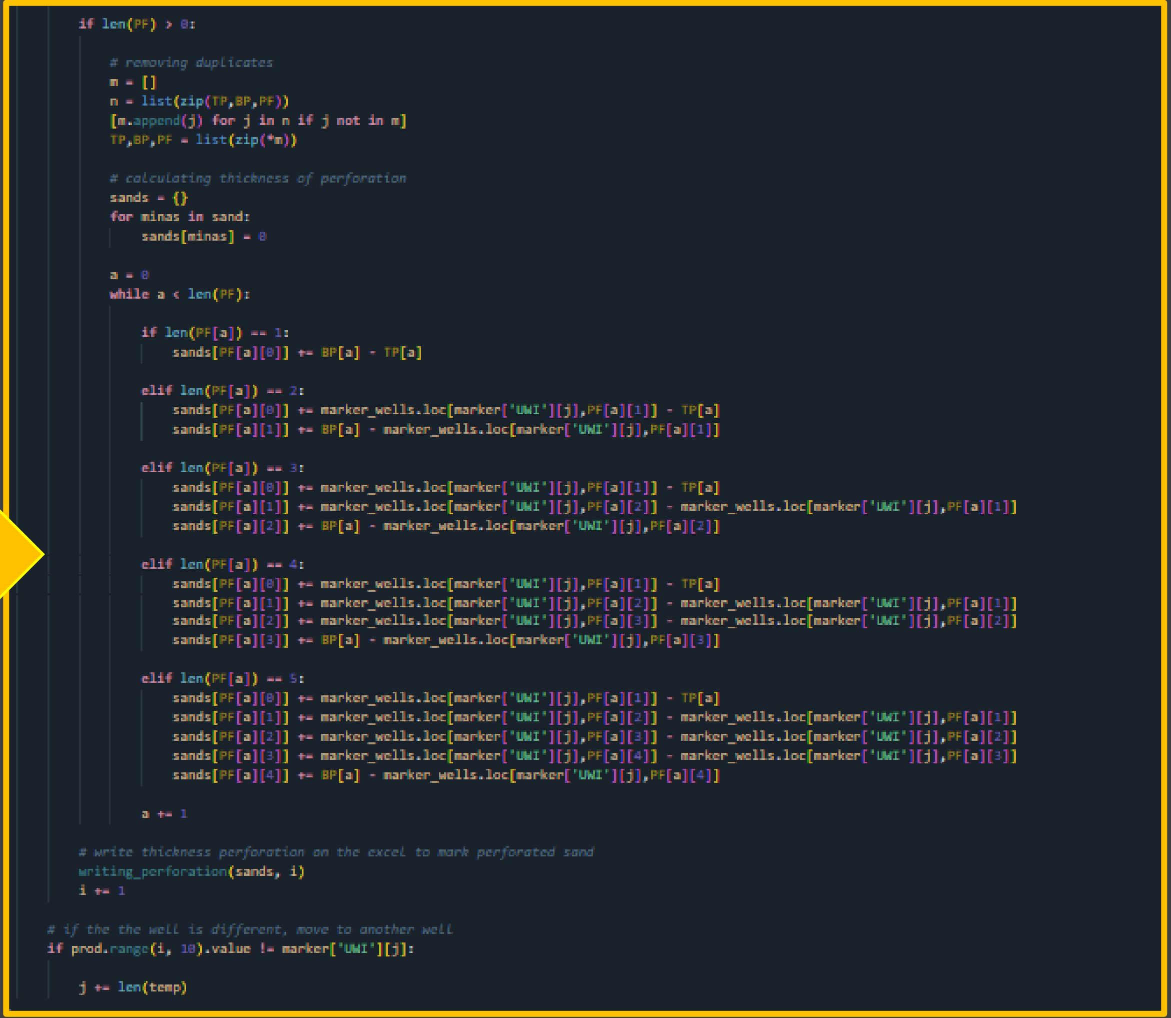


```python
def squeeze_machine(tp,bp,pf,pd,ts,bs,sf,sd):

    a = 0
    # looping through every squeezed event
    while a < len(sf):

        #checking each squeeze event to each perforation event
        b = 0
        while b < len (pf):

            # handling partial squeeze
            if (sd[a] >= pd[b]):

                # changing top perforation and deleting squeezed sand
                if ((ts[a] == tp[b]) and  bs[a] < bp[b]):
                    tp[b] = bs[a]
                    if (len(pf[b]) >= len(sf[a])):
                        pf[b] = [x for x in pf[b] if x not in sf[a]]

                # changing bottom perforation and deleting squeezed sand
                elif ((ts[a] > tp[b]) and  bs[a] == bp[b]):
                    bp[b] = ts[a]
                    if (len(pf[b]) >= len(sf[a])):
                        pf[b] = [x for x in pf[b] if x not in sf[a]]

            # deleting perforation event (date, top perforation, bottom perforation and sand) that squeezed
            if ((ts[a] == tp[b] and bs[a] == bp[b])
                and sd[a] >= pd[b]):
                del pf[b]
                del tp[b]
                del bp[b]
                del pd[b]
            else:
                b += 1

        a += 1

    # empting the unuseful list
    pd, ts, bs, sd, sf = [], [], [], [], []

    return tp,bp,pf
```

Code Time! (3)

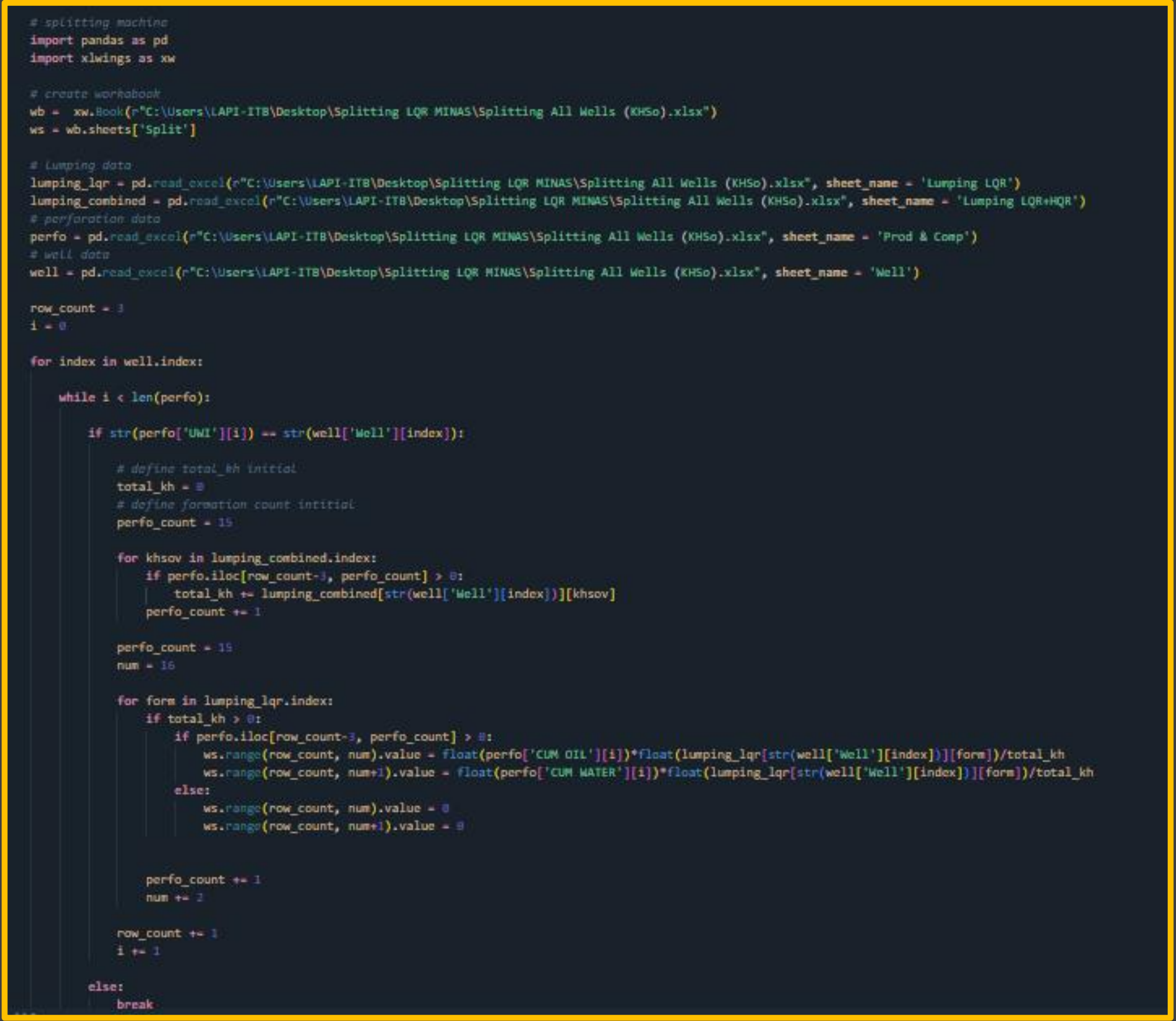


```python
# calculating thickness of perforated sand in each sand (modified "p" method)
import xlwings as xl

# initialising marker and production data
marker = perfo_wells
prod = xl.Book(r"C:\Users\LAPI-ITB\Desktop\Splitting LQR MINAS\Splitting All Wells by Thickness.xlsx").sheets['Prod & Comp (Hp)']

j = 0 # starting index for completion data
i = 0 # starting index for production data

# start the machine!
while prod.range(i, 10).value == marker['UWI'][j]:

    # if the well in marker and production data is the same
    if prod.range(i, 10).value == marker['UWI'][j]:

        # initialising temporary dataframe for well marker
        temp1 = perfo_wells[perfo_wells['UWI'] == marker['UWI'][j]]
        temp1 = temp1.dropna(axis = 1)
        temp1 = temp1.drop('MARKER', axis=1)
        temp1 = temp1.set_index('UWI')
        sand_marker = list(temp1.columns[4:])

        # create temporary dataframe from marker data that contain unique well
        temp = marker[marker['UWI'] == marker['UWI'][j]]

        # initialising x variable and blank array that contain perforation and squeeze data
        x = 0
        PF,SF,TP,BP,TS,BS,PD,SD = [],[],[],[],[],[],[],[]

        # inserting data to the array
        while x < len(temp):

            # dates in production data is compared to all of the dates data that meet the condition
            if (
                (prod.range(i, 4).value.year > marker['DATE'][x+j].year) or
                ((prod.range(i, 4).value.year == marker['DATE'][x+j].year) and
                 (prod.range(i, 4).value.month >= marker['DATE'][x+j].month))
                ):

                # perforation event
                if marker['STATUS'][x+j] == "perforation":

                    # appending perforation event data
                    PF.append(marker['MARKER'][x+j])
                    TP.append(marker['TOP'][x+j])
                    BP.append(marker['DEPTH'][x+j])
                    PD.append(marker['DATE'][x+j])
                # squeeze event
                elif marker['STATUS'][x+j] == "squeeze":

                    # appending squeeze event data
                    SF.append(marker['MARKER'][x+j])
                    TS.append(marker['TOP'][x+j])
                    BS.append(marker['DEPTH'][x+j])
                    SD.append(marker['DATE'][x+j])

                x += 1

            else:
                break

        # squeeze handling
        squeeze_machine(TP,BP,PF,PD,TS,BS,SF,SD)
```



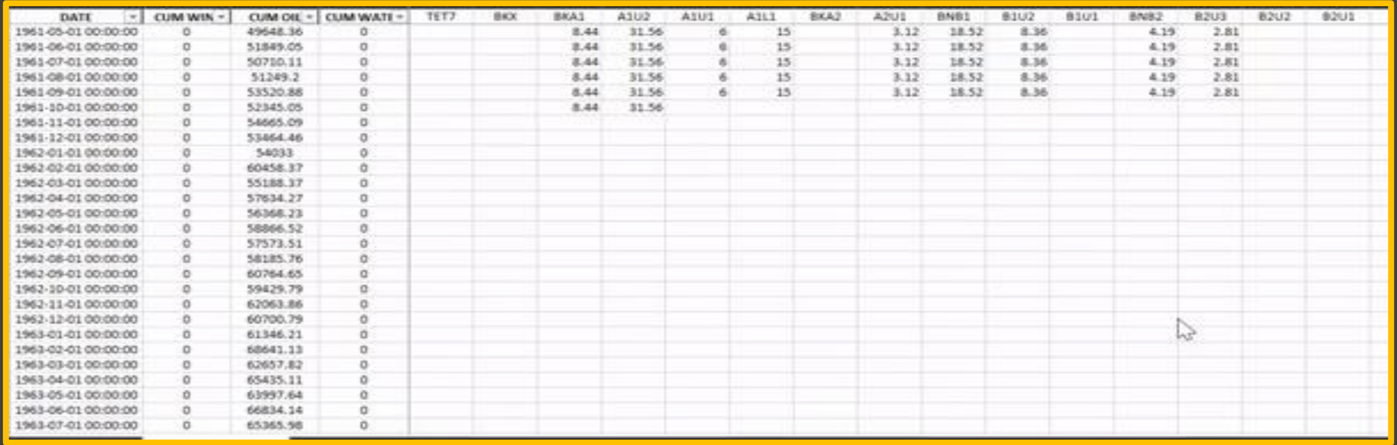
```python
        if len(PF) > 0:

            # removing duplicates
            m = []
            n = list(zip(TP,BP,PF))
            [m.append(j) for j in n if j not in m]
            TP,BP,PF = list(zip(*m))

            # calculating thickness of perforation
            sands = {}
            for minas in sand:
                sands[minas] = 0

            a = 0
            while a < len(PF):

                if len(PF[a]) == 1:
                    sands[PF[a][0]] += BP[a] - TP[a]

                elif len(PF[a]) == 2:
                    sands[PF[a][0]] += marker_wells.loc[marker['UWI'][j],PF[a][1]] - TP[a]
                    sands[PF[a][1]] += BP[a] - marker_wells.loc[marker['UWI'][j],PF[a][1]]

                elif len(PF[a]) == 3:
                    sands[PF[a][0]] += marker_wells.loc[marker['UWI'][j],PF[a][1]] - TP[a]
                    sands[PF[a][1]] += marker_wells.loc[marker['UWI'][j],PF[a][2]] - marker_wells.loc[marker['UWI'][j],PF[a][1]]
                    sands[PF[a][2]] += BP[a] - marker_wells.loc[marker['UWI'][j],PF[a][2]]

                elif len(PF[a]) == 4:
                    sands[PF[a][0]] += marker_wells.loc[marker['UWI'][j],PF[a][1]] - TP[a]
                    sands[PF[a][1]] += marker_wells.loc[marker['UWI'][j],PF[a][2]] - marker_wells.loc[marker['UWI'][j],PF[a][1]]
                    sands[PF[a][2]] += marker_wells.loc[marker['UWI'][j],PF[a][3]] - marker_wells.loc[marker['UWI'][j],PF[a][2]]
                    sands[PF[a][3]] += BP[a] - marker_wells.loc[marker['UWI'][j],PF[a][3]]

                elif len(PF[a]) == 5:
                    sands[PF[a][0]] += marker_wells.loc[marker['UWI'][j],PF[a][1]] - TP[a]
                    sands[PF[a][1]] += marker_wells.loc[marker['UWI'][j],PF[a][2]] - marker_wells.loc[marker['UWI'][j],PF[a][1]]
                    sands[PF[a][2]] += marker_wells.loc[marker['UWI'][j],PF[a][3]] - marker_wells.loc[marker['UWI'][j],PF[a][2]]
                    sands[PF[a][3]] += marker_wells.loc[marker['UWI'][j],PF[a][4]] - marker_wells.loc[marker['UWI'][j],PF[a][3]]
                    sands[PF[a][4]] += BP[a] - marker_wells.loc[marker['UWI'][j],PF[a][4]]

                a += 1

        # write thickness perforation on the excel to mark perforated sand
        writing_perforation(sands, i)
        i += 1

    # if the the well is different, move to another well
    if prod.range(i, 10).value != marker['UWI'][j]:

        j += len(temp)
```

Code Time! (Final Step)

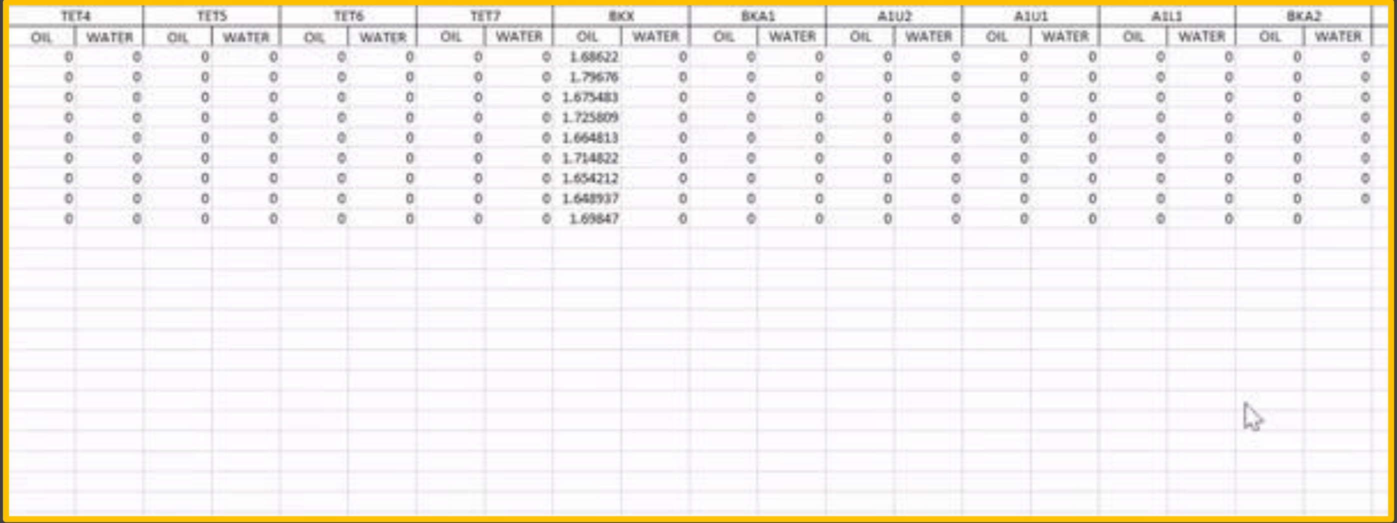


```python
# splitting machine
import pandas as pd
import xlwings as xw

# create workbook
wb =  xw.Book(r"C:\Users\LAPI-ITB\Desktop\Splitting LQR MINAS\Splitting All Wells (KHSo).xlsx")
ws = wb.sheets['Split']

# Lumping data
lumping_lqr = pd.read_excel(r"C:\Users\LAPI-ITB\Desktop\Splitting LQR MINAS\Splitting All Wells (KHSo).xlsx", sheet_name = 'Lumping LQR')
lumping_combined = pd.read_excel(r"C:\Users\LAPI-ITB\Desktop\Splitting LQR MINAS\Splitting All Wells (KHSo).xlsx", sheet_name = 'Lumping LQR+HQR')
# perforation data
perfo = pd.read_excel(r"C:\Users\LAPI-ITB\Desktop\Splitting LQR MINAS\Splitting All Wells (KHSo).xlsx", sheet_name = 'Prod & Comp')
# well data
well = pd.read_excel(r"C:\Users\LAPI-ITB\Desktop\Splitting LQR MINAS\Splitting All Wells (KHSo).xlsx", sheet_name = 'Well')

row_count = 3
i = 0

for index in well.index:

    while i < len(perfo):

        if str(perfo['UWI'][i]) == str(well['Well'][index]):

            # define total_kh initial
            total_kh = 0
            # define formation count intitial
            perfo_count = 15

            for khsov in lumping_combined.index:
                if perfo.iloc[row_count-3, perfo_count] > 0:
                    total_kh += lumping_combined[str(well['Well'][index])][khsov]
                perfo_count += 1

            perfo_count = 15
            num = 16

            for form in lumping_lqr.index:
                if total_kh > 0:
                    if perfo.iloc[row_count-3, perfo_count] > 0:
                        ws.range(row_count, num).value = float(perfo['CUM OIL'][i])*float(lumping_lqr[str(well['Well'][index])][form])/total_kh
                        ws.range(row_count, num+1).value = float(perfo['CUM WATER'][i])*float(lumping_lqr[str(well['Well'][index])][form])/total_kh
                    else:
                        ws.range(row_count, num).value = 0
                        ws.range(row_count, num+1).value = 0


                perfo_count += 1
                num += 2

            row_count += 1
            i += 1

        else:
            break
```

Result



DATE	CUM WIN	CUM OIL	CUM WATE	TET7	BKX	BKA1	A1U2	A1U1	A1L1	BKA2	A2U1	BNB1	B1U2	B1U1	BNB2	B2U3	B2U2	B2U1
1961-05-01 00:00:00	0	49648.36	0			8.44	31.56	6	15		3.12	18.52	8.36		4.19	2.81		
1961-06-01 00:00:00	0	51849.05	0			8.44	31.56	6	15		3.12	18.52	8.36		4.19	2.81		
1961-07-01 00:00:00	0	50710.11	0			8.44	31.56	6	15		3.12	18.52	8.36		4.19	2.81		
1961-08-01 00:00:00	0	51249.2	0			8.44	31.56	6	15		3.12	18.52	8.36		4.19	2.81		
1961-09-01 00:00:00	0	53520.88	0			8.44	31.56	6	15		3.12	18.52	8.36		4.19	2.81		
1961-10-01 00:00:00	0	52345.05	0			8.44	31.56											
1961-11-01 00:00:00	0	54665.09	0															
1961-12-01 00:00:00	0	53464.46	0															
1962-01-01 00:00:00	0	54033	0															
1962-02-01 00:00:00	0	60458.37	0															
1962-03-01 00:00:00	0	55188.37	0															
1962-04-01 00:00:00	0	57634.27	0															
1962-05-01 00:00:00	0	56368.23	0															
1962-06-01 00:00:00	0	58866.52	0															
1962-07-01 00:00:00	0	57573.51	0															
1962-08-01 00:00:00	0	58185.76	0															
1962-09-01 00:00:00	0	60764.65	0															
1962-10-01 00:00:00	0	59429.79	0															
1962-11-01 00:00:00	0	62063.86	0															
1962-12-01 00:00:00	0	60700.79	0															
1963-01-01 00:00:00	0	61346.21	0															
1963-02-01 00:00:00	0	68641.13	0															
1963-03-01 00:00:00	0	62657.82	0															
1963-04-01 00:00:00	0	65435.11	0															
1963-05-01 00:00:00	0	63997.64	0															
1963-06-01 00:00:00	0	66834.14	0															
1963-07-01 00:00:00	0	65365.98	0															



TET4		TET5		TET6		TET7		BKX		BKA1		A1U2		A1U1		A1L1		BKA2	
OIL	WATER	OIL	WATER	OIL	WATER	OIL	WATER	OIL	WATER	OIL	WATER	OIL	WATER	OIL	WATER	OIL	WATER	OIL	WATER
0	0	0	0	0	0	0	0	1.68622	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.79676	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.675483	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.725809	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.664813	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.714822	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.654212	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.648937	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.69847	0	0	0	0	0	0	0	0	0	0	

Thank You