Importing Library

In [1]:

```
import lasio
import numpy as np
import pandas as pd
```

Importing Lasio File

In [2]:

```
# Importing Lasio
las = lasio.read("E:\Kuliah\Tugas\Semester 7\Well Logging\Python Processing\MLD-05-B1,
 OH 12.25Inch Combined.las")
```

In [3]:

```
# Making dataframe 0
df = las.df()
```

Indexing

Indexing dilakukan untuk menyesuaikan Index data dengan variable

In [4]:

```
LLD0 = df['LLD']
LLS0 = df['LLS']
DT0 = df['DT']
SP0 = df['SP']
MSFL0 = df['MSFL']
GR0 = df['GR']
NPHI0 = df['NPHI']
DRHO0 = df['DRHO']
PEF0 = df['PEF']
RHOB0 = df['RHOB']
CAL0 = df['CALI']
```

Making dataframe 1

In [5]:

```
df1 = df.iloc[::-1]
```

Change data index from feet to m

In [6]:

```
df1.index *= 0.3048
```

Selecting depth (m to m)

In [7]:

```
df1 = df1[2496:2562] # Kelompok 7 2496-2562
df1
```

Out[7]:

DEPT	LLD	LLS	DT	SP	MSFL	GR	NPHI	DRHO	PEF	RHC
2496.0072	1.7785	2.0131	84.6	-539.2500	1.7964	108.1791	0.2802	0.0260	3.7884	2.558
2496.1596	1.6537	1.9631	86.3	-539.1250	0.6928	115.0081	0.3596	0.0337	4.1716	2.575
2496.3120	2.1105	2.2254	90.8	-539.0625	1.3091	114.4738	0.3628	0.0433	4.6224	2.616
2496.4644	6.4171	5.8067	95.8	-539.0000	4.0725	100.5820	0.2693	0.0545	4.6148	2.672
2496.6168	22.7664	18.2884	97.9	-538.8750	18.7516	78.4085	0.1283	0.0584	4.8341	2.720
...	...	...	...	...	...	...	...	...	...	
2561.3868	15.4643	14.2839	58.6	-519.4375	5.0668	43.6011	0.0538	0.0270	6.6120	2.580
2561.5392	18.7549	16.4898	58.5	-519.8125	3.6888	42.4693	0.0527	0.0306	6.3558	2.582
2561.6916	23.3354	19.3205	59.6	-520.0625	5.0954	40.8807	0.0400	0.0336	6.3235	2.582
2561.8440	29.2556	22.0448	60.5	-520.9375	4.5132	39.8121	0.0272	0.0488	6.5895	2.605
2561.9964	38.2383	27.0460	61.9	-520.8125	14.0219	37.9421	0.0164	0.0615	6.6092	2.624

434 rows × 11 columns

Set The Dataset

In [8]:

```
y = df1.index
y1 = df1.index.values
```

In [9]:

```
LLD = df1['LLD']
LLS = df1['LLS']
DT = df1['DT']
SP = df1['SP']
MSFL = df1['MSFL']
GR = df1['GR']
NPHI = df1['NPHI']
DRHO = df1['DRHO']
PEF = df1['PEF']
RHOB = df1['RHOB']
CAL = df1['CALI']
```

In [10]:

```
# Fixing SP to -80_20
df1['SP'] = SP + 505
```

Import Kuantitatif

In [11]:

```
dfk = pd.read_csv('E:\Kuliah\Tugas\Semester 7\Well Logging\Python Processing\kuantia.csv', sep = ';',index_col = 'depth')
dfk = dfk.iloc[::-1]
dfk1 = dfk[2496:2562]
yk = dfk1.index.values
```

Measuring Sw Metode 1

In [12]:

```
sw = []
sWa = ((1/(dfk1['F'].values**2))*
      (dfk1['Rw'].values/dfk1['Rt'].values))**(1/2)
sWa = sWa*100
```

Measuring Sw Metode 2, Porosity Density, dan Porosity Sonic

In [13]:

```
# Porosity Density
Por = []
Por = (2.65 - dfk1.RHOB)/(2.65 - 1.1)
df1['Porosity Density'] = Por.values*100
```

In [14]:

```
# Sw Calculation
#Sw = [len(Por)]
#Sw = ((df1.LLD.values/dfk1.Rw.values)*Por**2)**(-1/2)
#Sw = Sw/10
#df1['Water Saturation'] = Sw.values*100
```

In [15]:

```
# Porosity Sonic
Ps = []
Ps = (df1.DT - 55.5)/(189 - 55.5)
df1['Porosity Sonic'] = Ps.values*100
```

Perhitungan

Bitsize

In [16]:

```
BS = np.zeros(len(GR))
BS[:] = 12.5
```

Vshale Measurement

In [17]:

```
Vsh=[]
GRmin = 10 #Rio,2020 for Baturaja
GRmax = 110 #Rio,2020 for Baturaja
IGR = (GR-GRmin)/(GRmax-GRmin) #Index Gamma Ray
```

In [18]:

```
Vsh = 0.083*(2**(3.7*(IGR)-1))
Vsh = Vsh*100 #Persen
df1['Vshale'] = Vsh
```

Baseline

Gamma Ray Baseline

In [19]:

```
grb_min = np.zeros(len(GR))
grb_min[:] = GRmin

grb_max = np.zeros(len(GR))
grb_max[:] = GRmax

grbmed = (grb_max+grb_min)/2
```

SP Baseline

In [20]:

```
spb = np.zeros(len(SP))
spb[:] = 0
```

Calliper Baseline

In [21]:

```
calb = np.zeros(len(CAL))
calb[:] = BS[:]
```

Plot

Total Plot

In [22]:

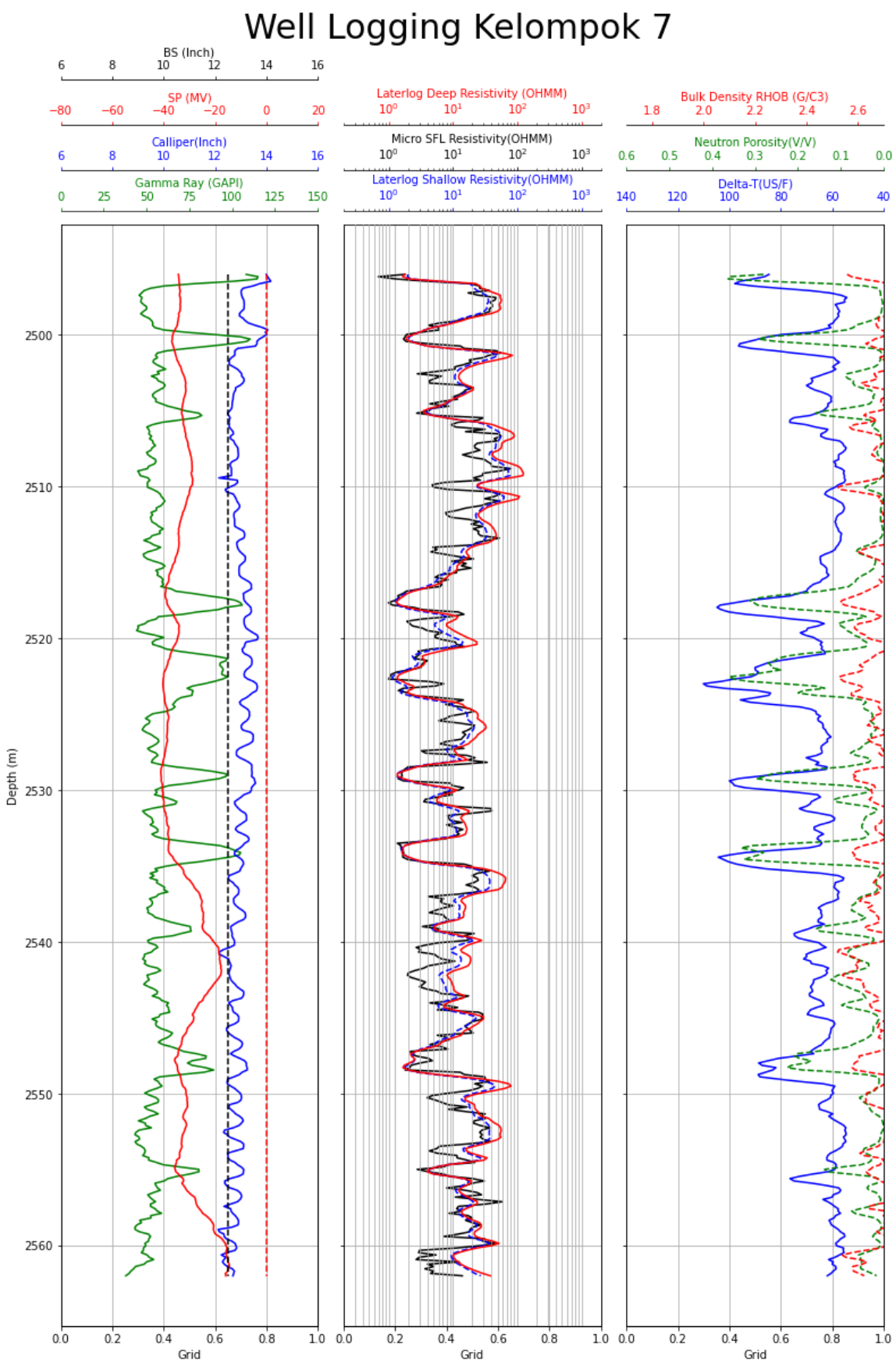
```
import matplotlib as mpl
from mpl_toolkits.axes_grid1 import make_axes_locatable
import matplotlib.ticker as ticker
import matplotlib.pyplot as plt
```

```
In C:\Python36\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t.mplstyle:
The text.latex.preview rcparam was deprecated in Matplotlib 3.3 and will b
e removed two minor releases later.
In C:\Python36\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t.mplstyle:
The mathtext.fallback_to_cm rcparam was deprecated in Matplotlib 3.3 and w
ill be removed two minor releases later.
In C:\Python36\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t.mplstyle: Support for setting the 'mathtext.fallback_to_cm' rcParam is d
eprecated since 3.3 and will be removed two minor releases later; use 'mat
htext.fallback : 'cm' instead.
In C:\Python36\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t.mplstyle:
The validate_bool_maybe_none function was deprecated in Matplotlib 3.3 and
will be removed two minor releases later.
In C:\Python36\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t.mplstyle:
The savefig.jpeg_quality rcparam was deprecated in Matplotlib 3.3 and will
be removed two minor releases later.
In C:\Python36\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t.mplstyle:
The keymap.all_axes rcparam was deprecated in Matplotlib 3.3 and will be r
emoved two minor releases later.
In C:\Python36\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t.mplstyle:
The animation.avconv_path rcparam was deprecated in Matplotlib 3.3 and wil
l be removed two minor releases later.
In C:\Python36\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_tes
t.mplstyle:
The animation.avconv_args rcparam was deprecated in Matplotlib 3.3 and wil
l be removed two minor releases later.
```

In [23]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 3,
                           figsize=(13,20), sharey=True)
fig.subplots_adjust(top=0.80, wspace=0.1)
fig.suptitle('Well Logging Kelompok 7', fontsize=30,
             y = 0.93)
ax[0].invert_yaxis()

#track 1
ax[0].grid(True, which = "both")
ax[0].set_ylabel("Depth (m)")
ax[0].set_xlabel("Grid")

ax01 = ax[0].twiny() # Gamma Ray
ax01.plot(GR, y, '-g')
ax01.set_xlim(0,150)
ax01.set_xlabel('Gamma Ray (GAPI)', color='g')
ax01.tick_params(axis = 'x', colors = 'g')
ax01.spines['top'].set_position(('outward', 12))

ax02 = ax[0].twiny() # Caliper
ax02.plot(CAL, y, '-b')
ax02.set_xlim(6,16)
ax02.set_xlabel('Calliper(Inch)', color='b')
ax02.tick_params(axis = 'x', colors = 'b')
ax02.spines['top'].set_position(('outward', 48))

ax03 = ax[0].twiny() # SP
ax03.plot(SP, y, '-r')
ax03.plot(spb, y, '--r') #Baseline
ax03.set_xlim(-80,20)
ax03.set_xlabel('SP (MV)', color='r')
ax03.tick_params(axis = 'x', colors = 'r')
ax03.spines['top'].set_position(('outward', 88))

ax04 = ax[0].twiny() # Bit Size
ax04.plot(BS, y, '--k')
ax04.set_xlim(6,16)
ax04.set_xlabel('BS (Inch)', color='black')
ax04.tick_params(axis = 'x', colors = 'black')
ax04.spines['top'].set_position(('outward', 128))

# track 2
ax[1].grid(True, which = "both")
ax[1].set_xlabel("Grid")

ax11 = ax[1].twiny() #MSFL
ax11.set_xlim(0.2,2000)
ax11.semilogx(MSFL, y, color = 'black')
ax11.set_xscale('log')
ax11.set_xlabel('Micro SFL Resistivity(OHMM)', color='black')
ax11.tick_params(axis = 'x', colors = 'black')
ax11.spines['top'].set_position (('outward', 48))

ax12 = ax[1].twiny() #LLS
ax12.set_xlim(0.2,2000)
ax12.semilogx(LLS, y, '--b')
ax12.set_xscale('log')
ax12.set_xlabel('Laterlog Shallow Resistivity(OHMM)', color='b')
```

```
ax12.tick_params(axis = 'x', colors = 'b')
ax12.spines['top'].set_position(('outward', 12))
ax12.grid(True, which = "both")

ax13 = ax[1].twiny() #LLD
ax13.set_xlim(0.2,2000)
ax13.semilogx(LLD, y, 'r')
ax13.set_xscale('log')
ax13.set_xlabel('Laterlog Deep Resistivity (OHMM)', color='r')
ax13.tick_params(axis = 'x', colors = 'r')
ax13.spines['top'].set_position(('outward', 88))

#track 3
ax[2].grid(True, which = "both")
ax[2].set_xlabel("Grid")

ax31 = ax[2].twiny() #DT
ax31.plot(DT, y, 'b')
ax31.set_xlim(140,40)
ax31.set_xlabel('Delta-T(US/F)', color='b')
ax31.tick_params(axis = 'x', colors = 'b')
ax31.spines['top'].set_position(('outward', 12))

ax32 = ax[2].twiny() #NPHI
ax32.plot(NPHI, y, '--g')
ax32.set_xlim(0.6,0)
ax32.set_xlabel('Neutron Porosity(V/V)', color='g')
ax32.tick_params(axis = 'x', colors = 'g')
ax32.spines['top'].set_position(('outward', 48))

ax33 = ax[2].twiny() #RHOB
ax33.plot(RHOB, y, '--r')
ax33.set_xlim(1.7,2.7)
ax33.set_xlabel('Bulk Density RHOB (G/C3)', color='r')
ax33.tick_params(axis = 'x', colors = 'r')
ax33.spines['top'].set_position(('outward', 88))
```


Well Logging Kelompok 7
BS (Inch)
SP (MV)
Calliper(Inch)
Gamma Ray (GAPI)
Laterlog Deep Resistivity (OHMM)
Micro SFL Resistivity(OHMM)
Laterlog Shallow Resistivity(OHMM)
Bulk Density RHOB (G/C3)
Neutron Porosity(V/V)
Delta-T(US/F)
Depth (m)
Grid

Interpretasi Kualitatif

Gamma Ray Picks

In [24]:

```
gr_top_pick = [np.min(y1)-1, 2497, 2499, 2501, 2504.5, 2506, 2510,
               2519, 2521, 2523, 2528.5, 2529.5, 2533.8, 2535.5, 2538,
               2539.5, 2547, 2550,2554,2555.8,
               np.max(y1)]
gr_f = [1,3,1,3,3,1,1,1,1,1,1,1,1,3,2,3,1,3,1,3]

gr_label = ('Shale', 'Shaly Sand', 'Sandstone')

gr_facies = np.zeros(len(GR))

for i in range(len(y1)):
    for j in range(len(gr_top_pick)-1):
        if y1[i] > gr_top_pick[j] and y1[i] <= gr_top_pick[j+1]:
            gr_facies[i]=gr_f[j]


df1['FACIES'] = gr_facies
```

In [25]:

```
df1
```

Out[25]:

DEPT	LLD	LLS	DT	SP	MSFL	GR	NPHI	DRHO	PEF	RHOE
2496.0072	1.7785	2.0131	84.6	-34.2500	1.7964	108.1791	0.2802	0.0260	3.7884	2.558(
2496.1596	1.6537	1.9631	86.3	-34.1250	0.6928	115.0081	0.3596	0.0337	4.1716	2.575(
2496.3120	2.1105	2.2254	90.8	-34.0625	1.3091	114.4738	0.3628	0.0433	4.6224	2.616(
2496.4644	6.4171	5.8067	95.8	-34.0000	4.0725	100.5820	0.2693	0.0545	4.6148	2.672(
2496.6168	22.7664	18.2884	97.9	-33.8750	18.7516	78.4085	0.1283	0.0584	4.8341	2.720(
...	...	...	...	...	...	...	...	...	...	..
2561.3868	15.4643	14.2839	58.6	-14.4375	5.0668	43.6011	0.0538	0.0270	6.6120	2.580'
2561.5392	18.7549	16.4898	58.5	-14.8125	3.6888	42.4693	0.0527	0.0306	6.3558	2.582(
2561.6916	23.3354	19.3205	59.6	-15.0625	5.0954	40.8807	0.0400	0.0336	6.3235	2.582'
2561.8440	29.2556	22.0448	60.5	-15.9375	4.5132	39.8121	0.0272	0.0488	6.5895	2.605-
2561.9964	38.2383	27.0460	61.9	-15.8125	14.0219	37.9421	0.0164	0.0615	6.6092	2.624'

434 rows × 15 columns

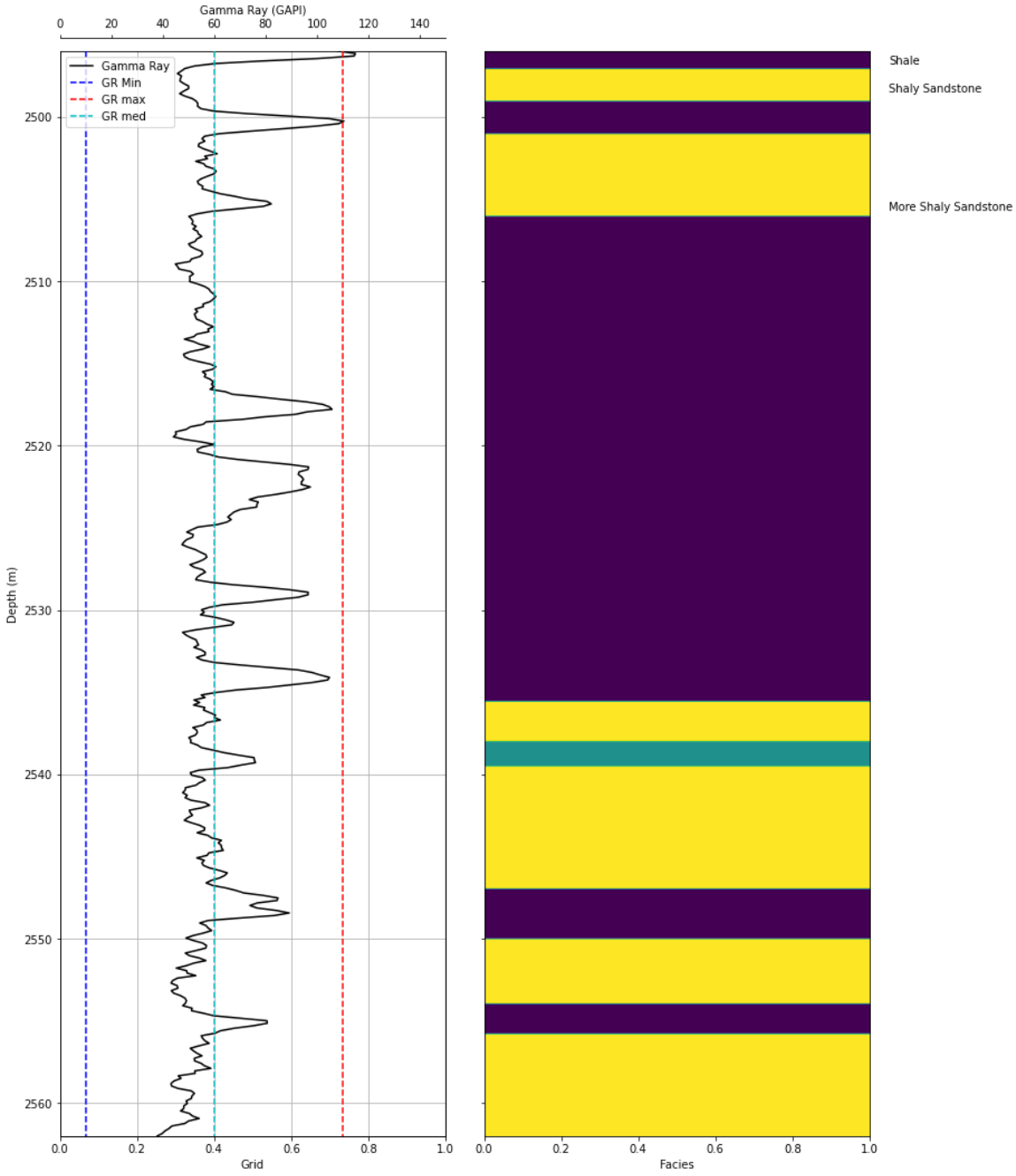
In [26]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 2,
                       figsize=(13,20), sharey=True)
fig.subplots_adjust(top=0.80, wspace=0.1)
fig.suptitle('Facies from GR', fontsize=30,
          y = 0.93)
ax[0].invert_yaxis()

#track 1
ax[0].grid(True, which = "both")
ax[0].set_ylabel("Depth (m)")
ax[0].set_xlabel("Grid")

ax01 = ax[0].twiny() # Gamma Ray
ax01.plot(GR, y, '-k', label = 'Gamma Ray')
# ax01.plot(grb_min, y, '--b', label = 'GR Min')
# ax01.plot(grb_max, y, '--r', label = 'GR max')
# grb2_min = np.zeros(len(GR))
# grb2_min[:] = np.min(GR)
# grb2_max = np.zeros(len(GR))
# grb2_max[:] = np.max(GR)
# grb2 = grb2_max-grb2_min

ax01.plot(grb_min, y, '--b', label = 'GR Min')
ax01.plot(grb_max, y, '--r', label = 'GR max')
ax01.plot(grbmed, y, '--c', label = 'GR med')
# ax01.plot(grb2, y, '--g', label = 'Mid')

ax01.legend()
ax01.set_xlim(0,150)
ax01.set_xlabel('Gamma Ray (GAPI)', color='k')
ax01.tick_params(axis = 'x', colors = 'k')
ax01.spines['top'].set_position(('outward', 12))

#track 2
GR_F = np.vstack((gr_facies, gr_facies)).T
ax[1].set_xlabel('Facies')
# cmap = mpl.colors.ListedColormap(['r', 'g', 'b'])
ax[1].imshow(GR_F, aspect = 'auto', extent = [0, 1, max(y), min(y)])
style = dict(size=10, color='gray')
ax[1].text(1.05, np.min(y1)+0.8, "Shale")
ax[1].text(1.05, np.min(y1)+2.5, "Shaly Sandstone")
ax[1].text(1.05, np.min(y1)+9.7, "More Shaly Sandstone")
```

Out[26]:

```
Text(1.05, 2505.7072, 'More Shaly Sandstone')
```

Facies from GR



Calliper

In [27]:

```
cal_top_pick = [np.min(y-1), 2500.5, 2510, 2539, 2543, 2557,

                np.max(y1)]
cal_f = [1,2,1,2,1,2]

cal_label = ('Impermeable', 'Permeable')
cal_facies = np.zeros(len(CAL))

for i in range(len(y1)):
    for j in range(len(cal_top_pick)-1):
        if y1[i] > cal_top_pick[j] and y1[i] <= cal_top_pick[j+1]:
            cal_facies[i]=cal_f[j]
```

Triple Pick

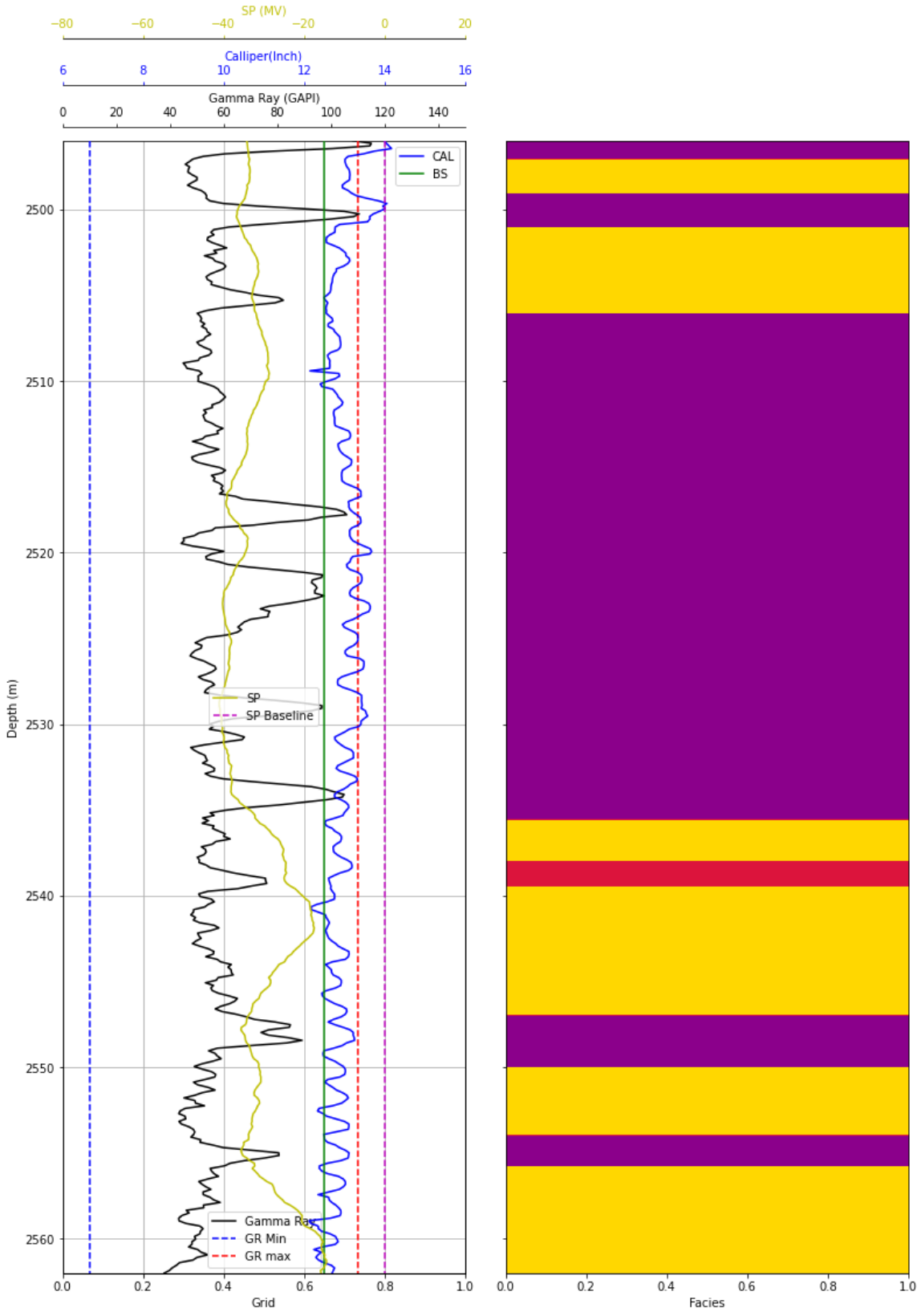
In [28]:

```
tp_top_pick = [np.min(y1)-1, 2497, 2499, 2501, 2504.5, 2506, 2516,
               2519, 2521, 2523, 2528.5, 2529.5, 2533.8, 2535.5, 2538,
               2539.5, 2547, 2550,2554,2555.8,
               np.max(y1)]
tp_f = [1,3,1,3,2,3,1,3,2,3,2,3,1,3,2,3,2,3,2,3]

tp_label = ('reservoir', 'not interested zone')

tp_facies = np.zeros(len(GR))

for i in range(len(y1)):
    for j in range(len(tp_top_pick)-1):
        if y1[i] > tp_top_pick[j] and y1[i] <= tp_top_pick[j+1]:
            tp_facies[i]=gr_f[j]
```

In [29]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 2,
                           figsize=(13,20), sharey=True)
fig.subplots_adjust(top=0.80, wspace=0.1)
fig.suptitle('Facies from SP', fontsize=30,
             y = 0.93)
ax[0].invert_yaxis()

#track 1
ax[0].grid(True, which = "both")
ax[0].set_ylabel("Depth (m)")
ax[0].set_xlabel("Grid")


ax01 = ax[0].twiny() # Gamma Ray
ax01.plot(GR, y, '-k', label = 'Gamma Ray')
ax01.plot(grb_min, y, '--b', label = 'GR Min')
ax01.plot(grb_max, y, '--r', label = 'GR max')
ax01.legend()
ax01.set_xlim(0,150)
ax01.set_xlabel('Gamma Ray (GAPI)', color='k')
ax01.tick_params(axis = 'x', colors = 'k')
ax01.spines['top'].set_position(('outward', 12))

ax02 = ax[0].twiny() # Caliper
ax02.plot(CAL, y, '-b', label = 'CAL')
ax02.plot(BS, y, '-g', label = 'BS')
ax02.set_xlim(6,16)
ax02.legend(loc = 'upper right')
ax02.set_xlabel('Calliper(Inch)', color='b')
ax02.tick_params(axis = 'x', colors = 'b')
ax02.spines['top'].set_position(('outward', 48))

ax03 = ax[0].twiny() # SP
ax03.plot(SP, y, '-y', label = 'SP')
ax03.plot(spb, y, '--m', label = 'SP Baseline') #Baseline

ax03.set_xlim(-80,20)
ax03.legend(loc = 10)
ax03.set_xlabel('SP (MV)', color='y')
ax03.tick_params(axis = 'x', colors = 'y')
ax03.spines['top'].set_position(('outward', 88))

#track 2
TP_F = np.vstack((tp_facies, tp_facies)).T
ax[1].set_xlabel('Facies')
cmap = mpl.colors.ListedColormap(['darkmagenta', 'crimson', 'gold'])
ax[1].imshow(TP_F, aspect = 'auto', extent = [0, 1, max(y), min(y)], cmap = cmap)
style = dict(size=10, color='gray')
```

Facies from SP



Resistivity Interpretation

Resistivitty Top Pick

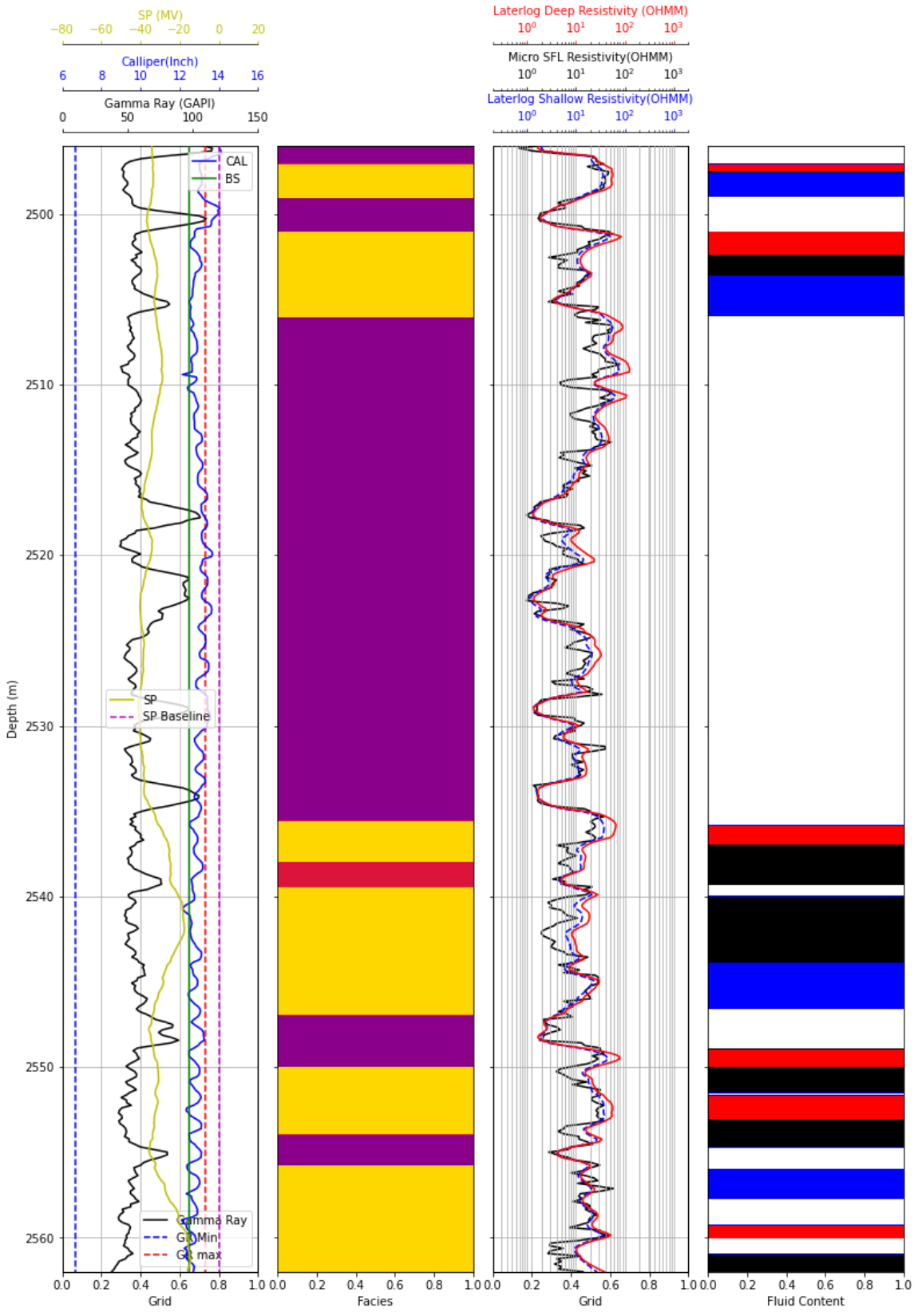
In [30]:

```
res_top_pick = [np.min(y1)-1 ,2496.95,2497, 2497.4,2499,
                2501, 2502.3,2503.5,2506, 2535.8, 2537,
                2539.3,2540, 2544, 2546.7,2549,
                2550,2551.5,2551.7,2553.1,2554.8,
                2556,2557.8,2559.3,2560.1,2561,
                np.max(y1)]
res_f = [4,2,1,3,4,1,2,3,4,1,2,4,2,3,4,1,2,4,1,2,4,3,4,1,4,2]

res_label = ('reservoir', 'not interested zone')

res_facies = np.zeros(len(GR))

for i in range(len(y1)):
    for j in range(len(res_top_pick)-1):
        if y1[i] > res_top_pick[j] and y1[i] <= res_top_pick[j+1]:
            res_facies[i]=res_f[j]
```

In [31]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 4,
                           figsize=(13,20), sharey=True)
fig.subplots_adjust(top=0.80, wspace=0.1)
# fig.suptitle('', fontsize=30,
              # y = 0.93)
ax[0].invert_yaxis()

#track 1
ax[0].grid(True, which = "both")
ax[0].set_ylabel("Depth (m)")
ax[0].set_xlabel("Grid")


ax01 = ax[0].twiny() # Gamma Ray
ax01.plot(GR, y, '-k', label = 'Gamma Ray')
ax01.plot(grb_min, y, '--b', label = 'GR Min')
ax01.plot(grb_max, y, '--r', label = 'GR max')
ax01.legend()
ax01.set_xlim(0,150)
ax01.set_xlabel('Gamma Ray (GAPI)', color='k')
ax01.tick_params(axis = 'x', colors = 'k')
ax01.spines['top'].set_position(('outward', 12))

ax02 = ax[0].twiny() # Caliper
ax02.plot(CAL, y, '-b', label = 'CAL')
ax02.plot(BS, y, '-g', label = 'BS')
ax02.set_xlim(6,16)
ax02.legend(loc = 'upper right')
ax02.set_xlabel('Calliper(Inch)', color='b')
ax02.tick_params(axis = 'x', colors = 'b')
ax02.spines['top'].set_position(('outward', 48))

ax03 = ax[0].twiny() # SP
ax03.plot(SP, y, '-y', label = 'SP')
ax03.plot(spb, y, '--m', label = 'SP Baseline') #Baseline

ax03.set_xlim(-80,20)
ax03.legend(loc = 10)
ax03.set_xlabel('SP (MV)', color='y')
ax03.tick_params(axis = 'x', colors = 'y')
ax03.spines['top'].set_position(('outward', 88))

#track 2
TP_F = np.vstack((tp_facies, tp_facies)).T
ax[1].set_xlabel('Facies')
cmap = mpl.colors.ListedColormap(['darkmagenta', 'crimson', 'gold'])
ax[1].imshow(TP_F, aspect = 'auto', extent = [0, 1, max(y), min(y)], cmap = cmap)
style = dict(size=10, color='gray')

#Track 3
ax[2].grid(True, which = "both")
ax[2].set_xlabel("Grid")

ax11 = ax[2].twiny() #MSFL
ax11.set_xlim(0.2,2000)
ax11.semilogx(MSFL, y, color = 'black')
ax11.set_xscale('log')
ax11.set_xlabel('Micro SFL Resistivity(OHMM)', color='black')
ax11.tick_params(axis = 'x', colors = 'black')
```

```
ax11.spines['top'].set_position (('outward', 48))

ax12 = ax[2].twiny() #LLS
ax12.set_xlim(0.2,2000)
ax12.semilogx(LLS, y, '--b')
ax12.set_xscale('log')
ax12.set_xlabel('Laterlog Shallow Resistivity(OHMM)', color='b')
ax12.tick_params(axis = 'x', colors = 'b')
ax12.spines['top'].set_position(('outward', 12))
ax12.grid(True, which = "both")

ax13 = ax[2].twiny() #LLD
ax13.set_xlim(0.2,2000)
ax13.semilogx(LLD, y, 'r')
ax13.set_xscale('log')
ax13.set_xlabel('Laterlog Deep Resistivity (OHMM)', color='r')
ax13.tick_params(axis = 'x', colors = 'r')
ax13.spines['top'].set_position(('outward', 88))

#track 4
RES_F = np.vstack((res_facies, res_facies)).T
ax[3].set_xlabel('Fluid Content')
cmap = mpl.colors.ListedColormap(['red', 'black', 'blue', 'white'])
ax[3].imshow(RES_F, aspect = 'auto', extent = [0, 1, max(y), min(y)], cmap = cmap)
style = dict(size=10, color='gray')
```


SP (MV)
−80 −60 −40 −20 0 20
Calliper(Inch)
6 8 10 12 14 16
Gamma Ray (GAPI)
0 50 100 150
Laterlog Deep Resistivity (OHMM)
Micro SFL Resistivity(OHMM)
Laterlog Shallow Resistivity(OHMM)
CAL
BS
SP
SP Baseline
Gamma Ray
GR Min
GR max
Depth (m)
2500
2510
2520
2530
2540
2550
2560
Grid
Facies
Grid
Fluid Content

Total Plot

Pick

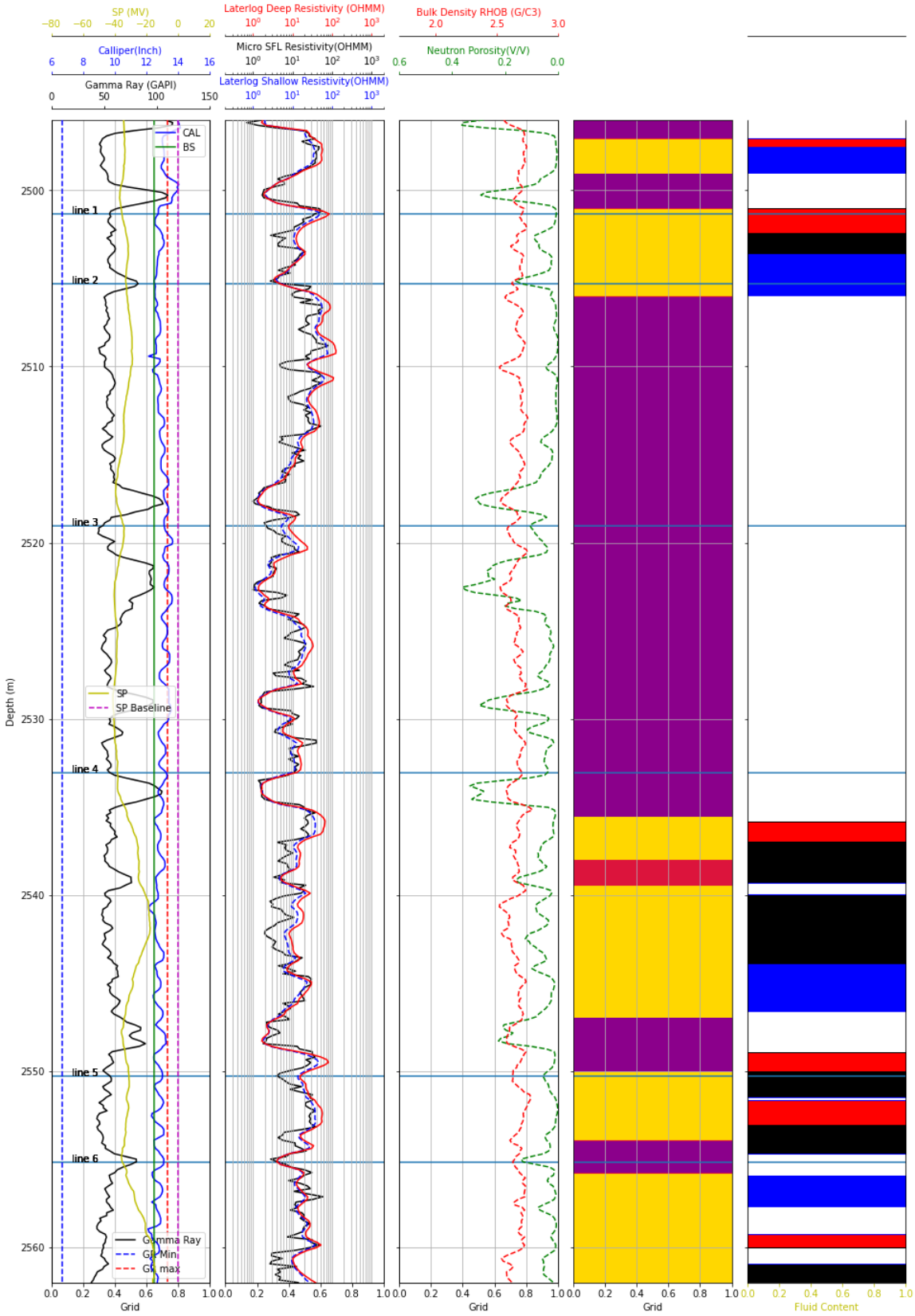
In [32]:

```
depth_pick = [2501.3412000000003,
              2505.3036,
              2519.0196,
              2533.0404000000003,
              2550.2616000000003,
              2555.1384000000003]
label_depth_pick = ['line 1', 'line 2', 'line 3', 'line 4', 'line 5', 'line 6']
```

In [33]:

```
fig, ax = plt.subplots(nrows = 1, ncols = 5, figsize=(16,25), sharey=True)
fig.subplots_adjust(top=0.80, wspace=0.1)
# fig.suptitle('', fontsize=30,
              # y = 0.93)
ax[0].invert_yaxis()

#track 1
ax[0].grid(True, which = "both")
ax[0].set_ylabel("Depth (m)")
ax[0].set_xlabel("Grid")


ax01 = ax[0].twiny() # Gamma Ray
ax01.plot(GR, y, '-k', label = 'Gamma Ray')
ax01.plot(grb_min, y, '--b', label = 'GR Min')
ax01.plot(grb_max, y, '--r', label = 'GR max')
ax01.legend()
ax01.set_xlim(0,150)
ax01.set_xlabel('Gamma Ray (GAPI)', color='k')
ax01.tick_params(axis = 'x', colors = 'k')
ax01.spines['top'].set_position(('outward', 12))

ax02 = ax[0].twiny() # Caliper
ax02.plot(CAL, y, '-b', label = 'CAL')
ax02.plot(BS, y, '-g', label = 'BS')
ax02.set_xlim(6,16)
ax02.legend(loc = 'upper right')
ax02.set_xlabel('Calliper(Inch)', color='b')
ax02.tick_params(axis = 'x', colors = 'b')
ax02.spines['top'].set_position(('outward', 48))

ax03 = ax[0].twiny() # SP
ax03.plot(SP, y, '-y', label = 'SP')
ax03.plot(spb, y, '--m', label = 'SP Baseline') #Baseline

ax03.set_xlim(-80,20)
ax03.legend(loc = 10)
ax03.set_xlabel('SP (MV)', color='y')
ax03.tick_params(axis = 'x', colors = 'y')
ax03.spines['top'].set_position(('outward', 88))

#track 2
TP_F = np.vstack((tp_facies, tp_facies)).T
ax[3].set_xlabel('Facies')
cmap = mpl.colors.ListedColormap(['darkmagenta', 'crimson', 'gold'])
ax[3].imshow(TP_F, aspect = 'auto', extent = [0, 1, max(y), min(y)], cmap = cmap)
style = dict(size=10, color='gray')

#Track 3
ax[3].grid(True, which = "both")
ax[3].set_xlabel("Grid")

ax11 = ax[1].twiny() #MSFL
ax11.set_xlim(0.2,2000)
ax11.semilogx(MSFL, y, color = 'black')
ax11.set_xscale('log')
ax11.set_xlabel('Micro SFL Resistivity(OHMM)', color='black')
ax11.tick_params(axis = 'x', colors = 'black')
ax11.spines['top'].set_position (('outward', 48))
```

```
ax12 = ax[1].twiny() #LLS
ax12.set_xlim(0.2,2000)
ax12.semilogx(LLS, y, '--b')
ax12.set_xscale('log')
ax12.set_xlabel('Laterlog Shallow Resistivity(OHMM)', color='b')
ax12.tick_params(axis = 'x', colors = 'b')
ax12.spines['top'].set_position(('outward', 12))
ax12.grid(True, which = "both")

ax13 = ax[1].twiny() #LLD
ax13.set_xlim(0.2,2000)
ax13.semilogx(LLD, y, 'r')
ax13.set_xscale('log')
ax13.set_xlabel('Laterlog Deep Resistivity (OHMM)', color='r')
ax13.tick_params(axis = 'x', colors = 'r')
ax13.spines['top'].set_position(('outward', 88))

#track 4
RES_F = np.vstack((res_facies, res_facies)).T
#ax[4].set_xlabel('Fluid Content')
cmap = mpl.colors.ListedColormap(['red', 'black', 'blue', 'white'])
# cmap = mpl.colors.ListedColormap(['white'])
ax[4].imshow(RES_F, aspect = 'auto', extent = [0, 1, max(y), min(y)], cmap = cmap)
style = dict(size=10, color='gray')
ax[4].set_xlabel('Fluid Content', color='y')
ax[4].tick_params(axis = 'x', colors = 'y')
ax[4].spines['top'].set_position(('outward', 88))

#track 5
ax[2].grid(True, which = "both")
ax[2].set_xlabel("Grid")

ax32 = ax[2].twiny() #NPHI
ax32.plot(NPHI, y, '--g')
ax32.set_xlim(0.6,0)
ax32.set_xlabel('Neutron Porosity(V/V)', color='g')
ax32.tick_params(axis = 'x', colors = 'g')
ax32.spines['top'].set_position(('outward', 48))

ax33 = ax[2].twiny() #RHOB
ax33.plot(RHOB, y, '--r')
ax33.set_xlim(1.7,3)
ax33.set_xlabel('Bulk Density RHOB (G/C3)', color='r')
ax33.tick_params(axis = 'x', colors = 'r')
ax33.spines['top'].set_position(('outward', 88))

for i in np.arange(0,5,1):
    for j in np.arange(0,len(depth_pick),1):
        ax[i].axhline(depth_pick[j])
        ax[0].text(0.13, depth_pick[j], label_depth_pick[j])

plt.savefig('E:\Kuliah\Tugas\Semester 7\Well Logging\Python Processing\Export\Export_pi
ck.jpg')
```


SP (MV)
Laterlog Deep Resistivity (OHMM)
Bulk Density RHOB (G/C3)
Calliper(Inch)
Micro SFL Resistivity(OHMM)
Neutron Porosity(V/V)
Gamma Ray (GAPI)
Laterlog Shallow Resistivity(OHMM)
CAL
BS
line 1
line 2
line 3
line 4
line 5
line 6
SP
SP Baseline
Gamma Ray
GR Min
GR max
Depth (m)
Grid
Grid
Grid
Fluid Content

In [34]:

```
df_q = df1.loc[depth_pick[:], :].copy()
df_q
```

Out[34]:

DEPT	LLD	LLS	DT	SP	MSFL	GR	NPHI	DRHO	PEF	RHOB
2501.3412	84.5538	50.1523	66.6	-35.2500	36.3449	55.2902	0.0050	0.0252	5.8359	2.7063
2505.3036	4.7408	5.1179	68.6	-32.7500	5.4675	82.2578	0.1379	0.0238	5.0707	2.6233
2519.0196	8.0065	5.2441	63.1	-34.2500	2.1314	49.1549	0.1055	0.0229	4.8352	2.6047
2533.0404	12.9207	9.7950	65.2	-38.1875	10.5874	55.1665	0.0377	0.0541	5.2735	2.7056
2550.2616	15.5800	13.7034	60.7	-30.8750	3.9807	55.5766	0.0564	0.0220	5.0469	2.6291
2555.1384	3.9965	4.2046	61.6	-35.2500	4.1402	80.6568	0.1320	0.0223	4.7959	2.6310

In [35]:

```
df_q = df_q.drop(columns = ['Vshale', 'FACIES'])
df_q
```

Out[35]:

DEPT	LLD	LLS	DT	SP	MSFL	GR	NPHI	DRHO	PEF	RHOB
2501.3412	84.5538	50.1523	66.6	-35.2500	36.3449	55.2902	0.0050	0.0252	5.8359	2.7063
2505.3036	4.7408	5.1179	68.6	-32.7500	5.4675	82.2578	0.1379	0.0238	5.0707	2.6233
2519.0196	8.0065	5.2441	63.1	-34.2500	2.1314	49.1549	0.1055	0.0229	4.8352	2.6047
2533.0404	12.9207	9.7950	65.2	-38.1875	10.5874	55.1665	0.0377	0.0541	5.2735	2.7056
2550.2616	15.5800	13.7034	60.7	-30.8750	3.9807	55.5766	0.0564	0.0220	5.0469	2.6291
2555.1384	3.9965	4.2046	61.6	-35.2500	4.1402	80.6568	0.1320	0.0223	4.7959	2.6310

In [ ]:

```

```

Hasil Plot

In [58]:

```
#por_ND
#por_ND =
df_q['por_ND'] = [0.5, 12, 9, 3, 5.1, 11]
df_q

df_hasil = df_q
df_q.drop('por_ND', axis = 1)
df_hasil = df_hasil.drop(['LLD', 'LLS', 'DT', 'SP', 'MSFL',
                'GR', 'NPHI', 'DRHO', 'PEF', 'RHOB',
                'CALI'], axis = 1)
df_hasil
```

Out[58]:

DEPT	Porosity Density	Porosity Sonic	por_ND	por_SD
2501.3412	-3.632258	8.314607	0.5	11.0
2505.3036	1.722581	9.812734	12.0	14.5
2519.0196	2.922581	5.692884	9.0	14.3
2533.0404	-3.587097	7.265918	3.0	12.8
2550.2616	1.348387	3.895131	5.1	5.0
2555.1384	1.225806	4.569288	11.0	4.0

In [59]:

```
df_hasil['por_SD'] = [11, 14.5, 14.3, 12.8, 5, 4]
df_hasil
```

Out[59]:

DEPT	Porosity Density	Porosity Sonic	por_ND	por_SD
2501.3412	-3.632258	8.314607	0.5	11.0
2505.3036	1.722581	9.812734	12.0	14.5
2519.0196	2.922581	5.692884	9.0	14.3
2533.0404	-3.587097	7.265918	3.0	12.8
2550.2616	1.348387	3.895131	5.1	5.0
2555.1384	1.225806	4.569288	11.0	4.0

In [60]:

```
df_hasil['por_SN'] = [6, 14.5, 11, 6, 7, 10]
df_hasil
```

Out[60]:

DEPT	Porosity Density	Porosity Sonic	por_ND	por_SD	por_SN
2501.3412	-3.632258	8.314607	0.5	11.0	6.0
2505.3036	1.722581	9.812734	12.0	14.5	14.5
2519.0196	2.922581	5.692884	9.0	14.3	11.0
2533.0404	-3.587097	7.265918	3.0	12.8	6.0
2550.2616	1.348387	3.895131	5.1	5.0	7.0
2555.1384	1.225806	4.569288	11.0	4.0	10.0

Formation Resistivity Factor

$F = \frac{1}{\Phi^2}$

In [62]:

```
# Mencari Fungsi Porositas
df_hasil['por_F'] = 1 / ((df_q['por_ND']/100)**2)
df_hasil
```

Out[62]:

DEPT	Porosity Density	Porosity Sonic	por_ND	por_SD	por_SN	por_F
2501.3412	-3.632258	8.314607	0.5	11.0	6.0	40000.000000
2505.3036	1.722581	9.812734	12.0	14.5	14.5	69.444444
2519.0196	2.922581	5.692884	9.0	14.3	11.0	123.456790
2533.0404	-3.587097	7.265918	3.0	12.8	6.0	1111.111111
2550.2616	1.348387	3.895131	5.1	5.0	7.0	384.467512
2555.1384	1.225806	4.569288	11.0	4.0	10.0	82.644628

Perhitungan RLLD

In [78]:

```
df_RLLD = df_q[['LLD']]
Rm = 0.05
df_RLLD['RLLD/Rm'] = df_RLLD['LLD']/Rm
df_RLLD['RLLDcor/RLLD'] = [1, 1.05, 1.05, 1.03, 1.01, 1.06]
df_RLLD['RLLDcor'] = df_RLLD['RLLDcor/RLLD'] * df_RLLD['LLD']
df_RLLD
```

```
C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing import
s until
C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

Out[78]:

DEPT	LLD	RLLD/Rm	RLLDcor/RLLD	RLLDcor
2501.3412	84.5538	1691.076	1.00	84.553800
2505.3036	4.7408	94.816	1.05	4.977840
2519.0196	8.0065	160.130	1.05	8.406825
2533.0404	12.9207	258.414	1.03	13.308321
2550.2616	15.5800	311.600	1.01	15.735800
2555.1384	3.9965	79.930	1.06	4.236290

Perhitungan RLLS

In [79]:

```
df_RLLS = df_q[['LLS']]
df_RLLS['RLLS/Rm'] = df_RLLS['LLS'] / Rm
df_RLLS['RLLScor/RLLS'] = [1.21, 1.15, 1.15, 1.16, 1.155, 1.15]
df_RLLS['RLLScor'] = df_RLLS['RLLScor/RLLS'] * df_RLLS['LLS']
df_RLLS
```

```
C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing import
s until
C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

Out[79]:

DEPT	LLS	RLLS/Rm	RLLScor/RLLS	RLLScor
2501.3412	50.1523	1003.046	1.210	60.684283
2505.3036	5.1179	102.358	1.150	5.885585
2519.0196	5.2441	104.882	1.150	6.030715
2533.0404	9.7950	195.900	1.160	11.362200
2550.2616	13.7034	274.068	1.155	15.827427
2555.1384	4.2046	84.092	1.150	4.835290

Perhitungan RMSFL

In [137]:

```
df_RMSFL = df_q[['MSFL']]
Rmc = 0.15
df_RMSFL['MSFL/Rmc'] = df_RMSFL['MSFL'] / Rmc
df_RMSFL['MSFLcor/MSFL'] = [0.91, 0.81, 0.8, 0.84, 0.8, 0.8]
df_RMSFL['MSFLcor'] = df_RMSFL['MSFLcor/MSFL'] * df_RMSFL['MSFL']
df_RMSFL
```

```
C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing import
s until
C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
C:\Users\Viraldi\AppData\Roaming\Python\Python36\site-packages\ipykernel_l
auncher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

Out[137]:

DEPT	MSFL	MSFL/Rmc	MSFLcor/MSFL	MSFLcor
2501.3412	36.3449	242.299333	0.91	33.073859
2505.3036	5.4675	36.450000	0.81	4.428675
2519.0196	2.1314	14.209333	0.80	1.705120
2533.0404	10.5874	70.582667	0.84	8.893416
2550.2616	3.9807	26.538000	0.80	3.184560
2555.1384	4.1402	27.601333	0.80	3.312160

Perhitungan Formation Water Resistivity

Menggunakan Rwa

In [110]:

```
df_calculation = df_RLLD[['RLLDcor']]
df_calculation = df_calculation.rename(columns = {'RLLDcor':'Rt(RLLDcor)'})
df_calculation['F'] = df_hasil['por_F']
df_calculation['Rwa(ohmm)'] = df_calculation['Rt(RLLDcor)']/df_calculation['F']
df_calculation
df_calculation
```

Out[110]:

DEPT	Rt(RLLDcor)	F	Rwa(ohmm)
2501.3412	84.553800	40000.000000	0.002114
2505.3036	4.977840	69.444444	0.071681
2519.0196	8.406825	123.456790	0.068095
2533.0404	13.308321	1111.111111	0.011977
2550.2616	15.735800	384.467512	0.040929
2555.1384	4.236290	82.644628	0.051259

Perhitungan Water Saturation

$$S_w = \sqrt{\frac{1 \times R_{wa}}{R_t \times N_\Phi^2}},\ R_{wa} = \frac{R_t}{F},\ HC_{content} = 1 - S_w$$

In [136]:

```
df_calculation['Sw'] = np.sqrt((1/(df_q['NPHI']**2)*(df_calculation['Rwa(ohmm)'])/df_ca
lculation['Rt(RLLDcor)']))
df_calculation['HCcontent'] = (1 - df_calculation['Sw'])*100
df_calculation
```

Out[136]:

DEPT	Rt(RLLDcor)	F	Rwa(ohmm)	Sw	HCcontent
2501.3412	84.553800	40000.000000	0.002114	1.000000	0.000000
2505.3036	4.977840	69.444444	0.071681	0.870196	12.980421
2519.0196	8.406825	123.456790	0.068095	0.853081	14.691943
2533.0404	13.308321	1111.111111	0.011977	0.795756	20.424403
2550.2616	15.735800	384.467512	0.040929	0.904255	9.574468
2555.1384	4.236290	82.644628	0.051259	0.833333	16.666667

In [133]:

```
#latex editor
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: