# Advanced data management concepts

**1- Verify if the Pandas and Bumpy libraries are installed in the DrillingAnalytics condo environment.**

conda list pandas

conda list numpy

Jupyter-Lab

**2- Load libraries into the notebook**

import pandas as pd

import numpy as np

**3- Declare a dictionary with drilling parameters**

data = {

'Depth (m)': np.random.uniform(1000, 5000, 10),

'Total Vertical Depth (m)': np.random.uniform(1500, 8000, 10),

'ROP (m/h)': np.random.uniform(5, 25, 10),

'Weight On Bit Driller (klbm)': np.random.uniform(2000, 6000, 10),

'Surface Torque (klbf.ft)': np.random.uniform(1000, 4000, 10),

'Surface RPM': np.random.uniform(80, 150, 10)

}

print(data)

**4- Transfer matrix to a pandas dataframe**

df = pd.DataFrame(data)

print("Matriz Pandas:")

print(df)

**5- Detect lines where ROP is greater than 15 m/h**

high\_rop\_rows = df[df['ROP (m/h)'] > 15]

print("Rows where ROP exceeds 15:")

print(high\_rop\_rows)

**6- Print working directory**

import os

print(“Directorio de trabajo:", os.getcwd())

**7- Import a CSV file**

file\_path = 'Data//ASCII//Well-CSV-1\_GSS\_DDRL\_201229\_5618.0m\_TD.csv'

df = pd.read\_csv(file\_path)

print(“CSV file dataframe:")

print(df)

**8- Confirm the Lasio library is installed**

Open a new Terminal/Command windows

Navigate to the DataScience folder

Confirm which condo environments are available: condo env list

If the active environment is not DrillingAnalytics, then activate it: conda activate DrillingAnalytics

Run conda env list and determine if the correct environment is active

Determine if Lasio is installed: conda list lasio

If lasio is not installed, then install with pip

Once again confirm is lasio is installed using conda list lasio

**9- Import time depth files**

import lasio

las\_file\_path = 'Data/LAS/Ironbark-1\_L601\_PWD\_FNL\_5618\_210107.las'

wellDepth = lasio.read(las\_file\_path)

print("\nData:")

print(wellDepth.df())

las\_file\_path = 'Data/LAS/Ironbark-1\_L601\_PWD\_FNL\_TIME\_210107.las'

wellTime = lasio.read(las\_file\_path)

print("\nData:")

print(wellTime.df())

**10- Import WITSML file and print available curves**

from bs4 import BeautifulSoup

WITSML\_file = r"Data/WITSML/1.xml"

with open(WITSML\_file) as f:

data = f.read()

data\_xml = BeautifulSoup(data, 'xml')

# Print the tags in the file

temp = set([str(tag.name) for tag in data\_xml.find\_all()])

print (“\n".join(temp))

**11- Load defines curve into a dataframe: md, tvd, inc, azi, dispNs, dispEw**

columns = ['md', 'tvd', 'incl', 'azi', 'dispNs', 'dispEw']

df = pd.DataFrame()

for col in columns:

df[col] = [float(x.text) for x in data\_xml.find\_all(col)]

print(df)

**10- Plot well trajectory**

import matplotlib

matplotlib.use('TkAgg')

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

## Plot the trajectory

fig = plt.figure()

ax = fig.add\_subplot(projection='3d')

# define the axis parameters

ax.plot(df['dispNs'], df['dispEw'], df['tvd']\*-1, '-r', linewidth = 2)

# format the plot

ax.set\_xlabel('dispNs', size=20, labelpad=30)

ax.set\_ylabel('dispEw', size=20, labelpad=30)

ax.set\_zlabel('tvd', size=20, labelpad=30)

ax.tick\_params(labelsize=15)

#set plot aspect ratio

ax.get\_proj = lambda: np.dot(Axes3D.get\_proj(ax), np.diag([0.5, 0.5,1.5, 1]))

fig.show()

**11- Load time LAS file: Barossa-6\_24hrs Time Ascii\_010517.LAS**

las\_file\_path = 'Data/LAS/Barossa-6\_24hrs Time Ascii\_010517.LAS'

wellTime1 = lasio.read(las\_file\_path)

print("\nData:")

df1=wellTime1.df()

df1.head()

df1.tail()

**12- Load time LAS file: Barossa-6\_24hrs Time Ascii\_020517.LAS**

las\_file\_path = 'Data/LAS/Barossa-6\_24hrs Time Ascii\_020517.LAS'

wellTime1 = lasio.read(las\_file\_path)

print("\nData:")

df2=wellTime1.df()

df2.head()

**13- Concatenate two data frames**

result\_vertical = pd.concat([df1, df2], ignore\_index=False)

# Print the result

print("Result after vertical concatenation:")

result\_vertical.head()

result\_vertical.tail()

**14- Is the head of result\_vertical the same as the head of df1?**

print("Es la cabeza de result\_vertical igual a la cabeza de df1? " + str(result\_vertical.head().equals(df1.head())))

**15- Is the tail of result\_vertical the same as the tail of df2?**

print("Es la cola de result\_vertical igual a la cola de df2? " + str(result\_vertical.tail().equals(df2.tail())))

**16- How many rows in df1 and df2?**

print("Numero de lineas en df1: "+str(len(df1)) + ", Numero de lineas en df2: ", str(len(df2)))

**17- How many rows in the concatenated dataframe?**

print("Numero de lineas en result\_vertical : “+str(len(result\_vertical)))

**18- Why hasn’t the column ETIM not incremented after concatenating?**

df\_reset = result\_vertical.reset\_index()

df\_reset.tail()

# Plot the values of the 'Values' column horizontally

df\_reset['ETIM'][-200:].plot(kind='line', marker='o', linestyle='-', color='skyblue')

# Customize the plot

plt.title('ETIM plot')

plt.xlabel('Index')

plt.ylabel('ETIM')

plt.show()

**19- Create a dataframe with the following information: 'Depth': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000], 'ROP': [45, 55, 60, 40, 70, 30, 80, 20, 90, 25], and create groups for ROP greater and smaller than 50.**

data = {

'Depth': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000],

'ROP': [45, 55, 60, 40, 70, 30, 80, 20, 90, 25]

}

df = pd.DataFrame(data)

# Create groups based on the condition ROP > 50

groups = df.groupby(df['ROP'] > 50)

# Print the groups

for name, group in groups:

print(f"Group: {name}")

print(group)

print(“\n")

**20- Load a WITSML file and determine if there is ROP information: 1-2.xml**

from bs4 import BeautifulSoup

WITSML\_file = r"Data/WITSML/1-2.xml"

with open(WITSML\_file) as f:

data = f.read()

data\_xml = BeautifulSoup(data, 'xml')

# Print the tags in the file

temp = set([str(tag.name) for tag in data\_xml.find\_all()])

print (“\n".join(temp))

**21- Create a dataframe with all the data**

tags\_at\_level = data\_xml.find\_all(lambda tag: len(tag.find\_parents()) == 4)

df=pd.DataFrame()

ilitho=-1

row={}

for index,tag in enumerate(tags\_at\_level):

if tag.name not in df.columns:

df[tag.name]=0

if tag.name=="typeLithology":

ilitho+=1

#print("New Horizon - "+str(ilitho))

#print(row)

if ilitho:

df=pd.concat([df,pd.DataFrame([row])],ignore\_index=True)

row={}

try:

row[tag.name]=float(tag.text)

except:

row[tag.name]=tag.text

#print(f"{ilitho} - {tag.name}: {tag.text}")

df.head(10)

**22- Convert ROP from meters per second to meters per hour**

df[‘ropAv’]=df['ropAv']\*3600

**23- Create two groups, one with ROP greater than 10 m/hr, and another of less than 10 m/hr**

# Create groups based on the condition ROP > 10

groups = df.groupby(df['ropAv'] > 10)

print("Grupo True")

groups.get\_group(True).head()

print("Grupo False")

groups.get\_group(False).head()