Conceptos avanzados de manejo de datos

1- Verificar si la libreria Pandas y numpy esta instalada en el ambiente DrillingAnalytics. Ejecutar Jupyter-Lab desde el directorio /DataScienceComodoro2023

```
conda list pandas
conda list numpy
cd ~/DataScienceComodoro2023
Jupyter-Lab
```

2- Cargar las librerías en el notebook

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

3- Crear una estructura de datos for datos de perforación - diccionario

```
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- Transferir datos a una matriz Pandas

```
df = pd.DataFrame(data)

print("Matriz Pandas:")
print(df)
```

5- Detectar lineas en donde la ROP ha excedido 15 m/h

```
high_rop_rows = df[df['ROP (m/h)'] > 15]

print("Rows where ROP exceeds 15:")
print(high_rop_rows)
```

6- Imprimir el nombre del activo de trabajo

```
import os

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

7- Importar data de un archivo CSV

```
file_path = 'Data//ASCII//Well-CSV-1_GSS_DDRL_201229_5618.0m_TD.csv'
```

```
df = pd.read_csv(file_path)

print("Matriz de un archivo CSV:")
print(df)
```

8- Corroborar si la libreria lasio esta instalada

Abrir un nueva ventana Terminal/Command
Navegar al directorio DataScienceComodoro2023
Confirmar cual ambiente conda esta activo con: conda env list
Si el ambiente conda activo no es DrillingAnalytics, entonces activar ambient con: conda activate DrillingAnalytics
Correr conda env list nuevamente y determinar si el ambiente correcto esta activado
Determinar si Casio esta instalado en este ambiente con: conda list lasio
Si la libreria lasio no esta instalada, installar con pip install lasio
Una vez la instalación es completada, confirmar que lasio esta instalado con conda list lasio

9- Importar archivos las en profundidad y tiempo

```
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- Importar un archivo WITSML é imprimir las curvas disponibles

```
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- Cargar estas curvas a un 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- Adelanto, gráfica de la trayectoria de un pozo

```
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- Importar el registro LAS en tiempo: 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- Importar el registro LAS en tiempo: 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- Concatenar los dos dataframes

```
result_vertical = pd.concat([df1, df2], ignore_index=False)

# Print the result
print("Result after vertical concatenation:")
result_vertical.head()

result_vertical.tail()
```

14- Es la cabeza de result_vertical igual a la cabeza de df1?

```
print("Es la cabeza de result_vertical igual a la cabeza de df1? " +
str(result_vertical.head().equals(df1.head())))
```

15- Es la cola de result_vertical igual a la cola de df2?

```
print("Es la cola de result_vertical igual a la cola de df2? " +
str(result_vertical.tail().equals(df2.tail())))
```

16- Cuantas líneas tines los dataframes df1 y df2 respectivamente?

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

17- Cuantas líneas tiene el dataframe con la concatenación?

```
print("Numero de lineas en result_vertical : “+str(len(result_vertical)))
```

18- Por qué la columna ETIM no ha incrementado después de completar la concatenación?

```
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- Crear un dataframe con la siguiente información: 'Depth': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000], 'ROP': [45, 55, 60, 40, 70, 30, 80, 20, 90, 25], y cree grupos para las lectura de ROP mayores de 50, y menores de 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- Cargar el archivo WITSML y determinar si hay información de ROP: 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- Crear un dataframe con los valores de ROP y otros parámetros en el registro

```
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- Converting ROP de metros por segundo a metros por hora

```
    df[‘ropAv’]=df['ropAv']*3600
```

23- Crear dos grupos, uno con ROP mayores de 10 m/hr y otro de menos de 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()
```