**Semana 3: Trabajo Experiencial y Cocreación Pandas**

**Objetivo:** Que los estudiantes adquieran habilidades prácticas en la manipulación y análisis de datos utilizando pandas.

**Actividad 1: Carga y Manipulación de Datos con pandas**

**Descripción:** Los estudiantes aprenderán a cargar datos en un DataFrame de pandas desde un conjunto de datos y realizarán operaciones básicas de manipulación de datos.

**Pasos a seguir:**

1. **Importar librerías necesarias y cargue de datos**
   1. Definir las librerías que utilizará para su análisis, siendo necesaria principalmente la de pandas.
   2. Importar los datos en la sección de Archivos
   3. Cargar datos con los que va a trabajar en un DataFrame de pandas.

**Ej:**

import pandas as pd

import numpy as np

import plotly.express as px

#Cargar datos

df1=pd.read\_excel("/sample\_data/Aportes\_Energético\_Largo\_Plazo.xlsx")

# Mostrar las primeras filas del DataFrame

print(data.head(10))

print(data.tail(10))

1. **Exploración Básica de los Datos:**
   1. Utilizar métodos de pandas para explorar y describir el DataFrame.
   2. **Ejemplo:** Describir el DataFrame y mostrar las estadísticas básicas.

**Ej:**

print(df1.describe()) # Estadísticas descriptivas

print(df1.info()) # Información sobre las columnas y los tipos de datos

1. **Operaciones sobre las columnas:**
   1. Convertir columnas a su tipo de dato real.
   2. Dividir una columna en más columnas de ser necesario para facilitar el análisis.

**Ej:**

# Convertir la columna 'FechaInicio' al tipo datetime

df2['FechaInicio'] = pd.to\_datetime(df2['FechaInicio'])

# Dividir la columna de fechas en año y mes

df2['Año\_Inicio'] = df2['FechaInicio'].dt.year

df2['Mes\_Inicio'] = df2['FechaInicio'].dt.month

df2['Dia\_Inicio'] = df2['FechaInicio'].dt.day

1. **Limpieza de datos:**
   1. Identificar valores nulos que se encuentran en las diferentes columnas del conjunto de datos.
   2. Eliminar los registros (filas) que tengan datos nulos en una columna específica o que tenga valores nulos en varias columnas.
   3. Eliminar datos duplicados

**EJ:**

# Identificar valores nulos

valores\_nulos = df2.isnull().sum()

valores\_nulos

# Eliminar valores nulos de una columna

df2**.**dropna**(**subset**=[**'AportesProyectados'**],** inplace**=**True**)**

#Eliminar valores nulos de todo el DataFrame

df2.dropna()

# Eliminar filas duplicadas

df2 = df2.drop\_duplicates()

1. **Calcular correlación**
   1. Como las correlaciones se realizan sobre tipo de datos numéricos, creamos un dataFrame con las columnas numéricas de nuestro conjunto de datos .
   2. Aplicamos la correlación de pearson o de spearman o ambas.
   3. Crear matriz de correlación que muestre mejor los resultados.

**EJ:**

# Seleccionar solo las columnas numéricas del DataFrame

df\_numericas = df2.select\_dtypes(include=[np.number])

# Calcular la matriz de correlación de Pearson (el método por defecto)

correlacion\_pearson = df\_numericas.corr()

correlacion\_spearman = df\_numericas.corr(method= 'spearman')

# Matriz de correlación

figCorrelacion = px.imshow(correlacion\_pearson , text\_auto=True, aspect="auto", title='Matriz de Correlación')

figCorrelacion.show()

1. **Exploración de datos:**
   1. Identificar la cantidad de aportes proyectados para cada mes y año.
   2. Obtener el mes de cada año en el que los aportes son mayores.
   3. Obtener el mes de cada año en el que los aportes son menores.
   4. Obtener la cantidad de registros por etapa y hacer un gráfico de pastel con estos datos.
   5. Agrupar el conjunto de datos de acuerdo al año de inicio y al año fin, y ordenarlos de acuerdo al total de AportesProyectados.
   6. Explorar y realizar al menos 2 consultas para obtener información que desee del conjunto de datos y graficarlo.

**EJ:**

#Cantidad de aportes por mes y año

aportes\_mes = df2.groupby(['Año\_Inicio','Mes\_Inicio']).agg({'Etapa': 'count', 'AportesProyectados': 'sum'}).reset\_index()

aportes\_mes.rename(columns={'Etapa': 'Cantidad Registros'}, inplace=True)

# Ordenar los resultados de acuerdo al año y mes

aportes\_mes = aportes\_mes.sort\_values(by=['Año\_Inicio','Mes\_Inicio']

aportes\_mes

# Mes de cada año donde se realizan más aportes

max =aportes\_mes.groupby('Año\_Inicio')['AportesProyectados'].idxmax()

mes\_mas\_costoso = aportes\_mes.loc[max]

mes\_mas\_costoso

# Mes de cada año donde se realizan menos aportes

min =aportes\_mes.groupby('Año\_Inicio')['AportesProyectados'].idxmin()

mes\_menos\_costoso = aportes\_mes.loc[min]

mes\_menos\_costoso

#Contar registros por etapa

conteo\_por\_etapa = f2.groupby('Etapa')['Etapa'].count().reset\_index(name='Cantidad')

conteo\_por\_etapa

#Graficar

fig = px.pie(conteo\_por\_etapa, values='Cantidad', names='Etapa', title='Cantidad de Registros por Etapa')

fig.show()

# Agrupar por año y mes de inicio y fin

aportes\_i\_f= df2.groupby(['Año\_Inicio', 'Mes\_Inicio', 'Año\_Fin','Mes\_Fin']).agg({'Etapa': 'count', 'AportesProyectados':'sum'}).reset\_index()

aportes\_i\_f.rename(columns={'Etapas': 'Cantidad'})

#Ordenarlo de forma descendente

aportes\_i\_f = aportes\_i\_f.sort\_values(by='AportesProyectados', ascending=False)

aportes\_i\_f

**EJ:**

# Realizar consultas con una condición fija

# Se define la condición

condicion = df2['AportesProyectados'] > 30000000

# Filtramos el dataframe

registros\_filtrados = df2[condicion]

registros\_filtrados

# Realizar consultas con varias condiciones

# Definir condiciones

condicion1 = df2['AportesProyectados'] > 30000000

condicion2 = df2['Año\_Inicio'] == 2023

condicion3 = df2['Año\_Fin'] == 2024

# Combinar condiciones con operadores lógicos AND (&) OR (|)

condicion\_combinada = condicion1 & (condicion2 | condicion3)

# Filtramos el dataframe

registros\_filtrados2 = df2[condicion\_combinada]

registros\_filtrados2