Laboratorio No. 1

Leidy D. Galindo Acuña

Universidad ECCI

Seminario Big Data y Gerencia de datos

LABORATORIO #1

Desarrollo

Se importan librerías y se crea el DataFrame a traves de un conjunto de datos. Un DataFrame es básicamente una estructura de datos tabular, con filas y columnas. Las filas tienen un índice específico para acceder a ellas, que puede ser cualquier nombre o valor.

```python
# Importación de librerias
import pandas as pd
import numpy as np
import matplotlib. pyplot as plt
```

El resultado de la estructura de datos de DataFrame puede verse como una hoja de cálculo

```python
# Se contruye el dataframe
data = {'year': [2010, 2011, 2012,
                 2010, 2011, 2012,
                 2010, 2011, 2012],
        'team': ['FCBarcelona', 'FCBarcelona',
                 'FCBarcelona', 'RMadrid',
                 'RMadrid', 'RMadrid',
                 'ValenciaCF', 'ValenciaCF',
                 'ValenciaCF'],
        'wins': [30, 28, 32, 29, 32, 26, 21, 17, 19],
        'draws': [6, 7, 4, 5, 4, 7, 8, 10, 8],
        'losses': [2, 3, 2, 4, 2, 5, 9, 11, 11]}

football = pd.DataFrame(data, columns=['year', 'team', 'wins', 'draws', 'losses'])

football
```

	year	team	wins	draws	losses
0	2010	FCBarcelona	30	6	2
1	2011	FCBarcelona	28	7	3
2	2012	FCBarcelona	32	4	2
3	2010	RMadrid	29	5	4
4	2011	RMadrid	32	4	2
5	2012	RMadrid	26	7	5
6	2010	ValenciaCF	21	8	9
7	2011	ValenciaCF	17	10	11
8	2012	ValenciaCF	19	8	11

LABORATORIO #1

Se carga un repositorio de datos en formato CSV otorgando permisos de acceso a Drive, especificando la ruta donde está guardado el archivo y el nombre de las columnas del mismo.

```
[3]  1 from google.colab import drive
     2 drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
1 edu = pd.read_csv("/content/drive/MyDrive/data/educ_figdp_1_Data.csv",
2
3 na_values = ':',
4 usecols = ["TIME","GEO","Value"])
5
6 edu
```

	TIME	GEO	Value
0	2000	European Union (28 countries)	NaN
1	2001	European Union (28 countries)	NaN
2	2002	European Union (28 countries)	5.00
3	2003	European Union (28 countries)	5.03
4	2004	European Union (28 countries)	4.95
...	...	...	...
379	2007	Finland	5.90
380	2008	Finland	6.10
381	2009	Finland	6.81
382	2010	Finland	6.85
383	2011	Finland	6.76

384 rows × 3 columns

Se realiza una consulta utilizando el método tail(), que devuelve las últimas cinco filas de forma predeterminada. Además se filtra por la comuna ‘Value’ para traer los datos mayores a 6.5

```
1 edu[edu['Value'] > 6.5].tail()
```

	TIME	GEO	Value
286	2010	Malta	6.74
287	2011	Malta	7.96
381	2009	Finland	6.81
382	2010	Finland	6.85
383	2011	Finland	6.76

LABORATORIO #1

Se filtra en la columna ‘Value’ los valores nulos.

```
1 # Se filtran los valores nulos
2 edu[edu["Value"].isnull()].head()
```

	TIME	GEO	Value
0	2000	European Union (28 countries)	NaN
1	2001	European Union (28 countries)	NaN
36	2000	Euro area (18 countries)	NaN
37	2001	Euro area (18 countries)	NaN
48	2000	Euro area (17 countries)	NaN

Con la función max(), se obtienen los valores máximos de cada columna

```
1 # Obteniendo los máximos para cada columna
2 edu. max(axis = 0)
```

```
TIME      2011
GEO      Spain
Value     8.81
dtype: object
```

Se compara la función max() de Pandas y la de Python. En Python, los valores NaN se propagan a través de todas las operaciones sin generar una excepción. Por el contrario, las operaciones de Pandas excluyen los valores NaN que representan datos faltantes.

```
[10] 1 print("Pandas max function:", edu['Value'].max())
     2 print("Python max function:", max(edu['Value']))
```

```
Pandas max function: 8.81
Python max function: nan
```

Aplicación de la raíz cuadrada a cada columna.

LABORATORIO #1

```
# raiz cuadradada a los valores
s = edu["Value"]. apply (np.sqrt)

s.head()
```

```
0         NaN
1         NaN
2    2.236068
3    2.242766
4    2.224860
Name: Value, dtype: float64
```

Aplicación de la lambda a cada columna

```
# Aplicación de lambda
s = edu["Value"]. apply ( lambda d: d**2)

s.head()
```

```
0         NaN
1         NaN
2    25.0000
3    25.3009
4    24.5025
Name: Value, dtype: float64
```

Utilizando el operador de asignación = se pueden establecer nuevos valores en un DataFrame. Se asigna la Serie que resulta de dividir la columna ‘Value’ por el valor máximo en la misma columna a una nueva columna denominada ‘ValueNorm’

```
# Se dan nuevos valores al dataframe, a partir de calculo matematico
edu['ValueNorm'] = edu['Value']/edu['Value']. max ()

edu.tail()
```

	TIME	GEO	Value	ValueNorm
379	2007	Finland	5.90	0.669694
380	2008	Finland	6.10	0.692395
381	2009	Finland	6.81	0.772985
382	2010	Finland	6.85	0.777526
383	2011	Finland	6.76	0.767310

Si queremos eliminar esta columna del DataFrame, podemos usar la función axis; esto elimina las filas indicadas si el axis = 0, o las columnas indicadas si axis = 1.

```
[14]  1 # Se Borran columnas ValueNorm
      2 edu.drop('ValueNorm', axis = 1, inplace = True)
      3
      4 edu.head()
```

	TIME	GEO	Value
0	2000	European Union (28 countries)	NaN
1	2001	European Union (28 countries)	NaN
2	2002	European Union (28 countries)	5.00
3	2003	European Union (28 countries)	5.03
4	2004	European Union (28 countries)	4.95

Se ordena el DataFrame utilizando la siguiente función especificando el orden ascendente de la columna ‘Value’.

```
1 # Ordenamientos
2 edu.sort_values(by = 'Value', ascending = False ,
3
4 inplace = True)
5
6 edu.head()
```

	TIME	GEO	Value
130	2010	Denmark	8.81
131	2011	Denmark	8.75
129	2009	Denmark	8.74
121	2001	Denmark	8.44
122	2002	Denmark	8.44

Se agrupan todos los datos por país, independientemente del año utilizando la función groupby

LABORATORIO #1

```python
# Agrupamientos
group = edu[["GEO", "Value"]].groupby('GEO').mean()
group.head()
```

GEO	Value
Austria	5.618333
Belgium	6.189091
Bulgaria	4.093333
Cyprus	7.023333
Czech Republic	4.168333

Se realiza la transformación de la disposición de los datos, redistribuyendo los índices y columnas usando la función pivot_table para realizar una mejor manipulación de los datos.

```python
# Filtrado de tabla y pivoteo por las columnas GEO y TIME
filtered_data = edu[edu["TIME"] > 2005]

pivedu = pd. pivot_table( filtered_data , values = 'Value',

index = ['GEO'] ,
columns = ['TIME'])

pivedu.head()
```

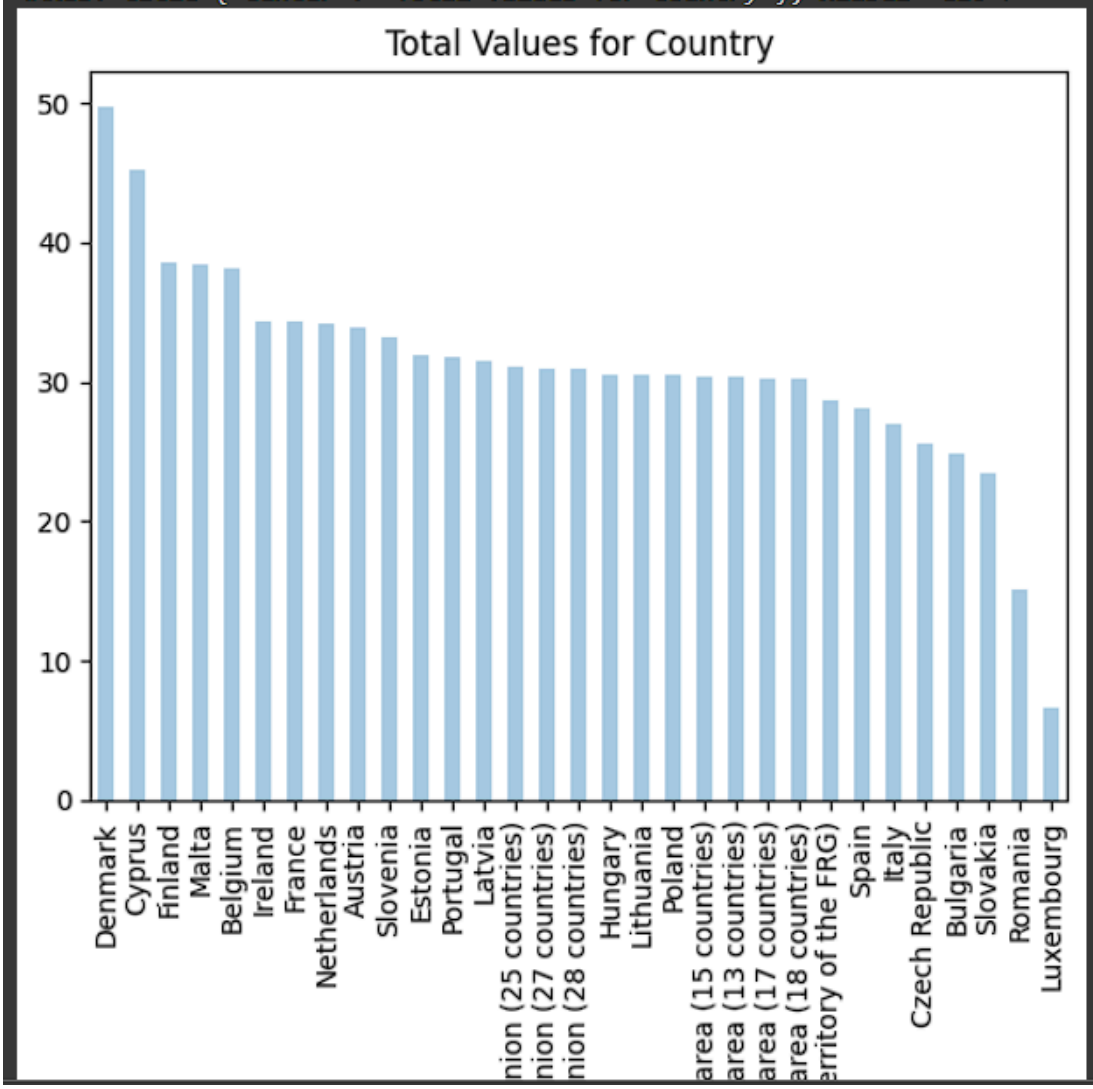
GEO \ TIME	2006	2007	2008	2009	2010	2011
Austria	5.40	5.33	5.47	5.98	5.91	5.80
Belgium	5.98	6.00	6.43	6.57	6.58	6.55
Bulgaria	4.04	3.88	4.44	4.58	4.10	3.82
Cyprus	7.02	6.95	7.45	7.98	7.92	7.87
Czech Republic	4.42	4.05	3.92	4.36	4.25	4.51

Al utilizar la biblioteca para gráficos Matplotlib, , si grafican los valores acumulados para cada país durante los últimos 6 años:

LABORATORIO #1

```
1 # Graficas
2 totalSum = pivedu. sum(axis = 1).sort_values(ascending = False)
3 totalSum. plot(kind = 'bar', style = 'b', alpha = 0.4,
4 title = "Total Values for Country")
```

```
<Axes: title={'center': 'Total Values for Country'}, xlabel='GEO'>
```



LABORATORIO #1

```
my_colors = ['b', 'r', 'g', 'y', 'm', 'c']
ax = pivedu. plot(kind = 'barh',
stacked = True ,
color = my_colors)

ax.legend(loc = 'center left', bbox_to_anchor = (1, .5))
```

```
<matplotlib.legend.Legend at 0x7dabaaf054b0>
```