

Ingeniería de Computación y Sistemas

# INGENIERÍA DE DATOS

Mg. Ing. Carlos Edwin Julca Castillo

**UPAO**  
UNIVERSIDAD PRIVADA ANTENOR ORREGO

# UNIDAD II

## SESIÓN 9: Librería PANDAS

**UPAO**

UNIVERSIDAD PRIVADA ANTENOR ORREGO

# CASO PRÁCTICO: Ordenar Columnas

# Método `sort_values()`:

`DataFrame.sort_values()`: Ordena un DataFrame en orden ascendente o descendente de una columna.

## SINTAXIS:

```
pandas.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
```

- **by:** Único/Lista de nombres de columnas para ordenar el DataFrame.
- **axis:** 0 o índice para filas y 1 o columnas para Columna.
- **ascending:** Valor booleano que ordena el DataFrame en orden ascendente si es Verdadero.
- **inplace:** Valor booleano que realiza los cambios en el DataFrame si es Verdadero.
- **kind:** Cadena (String) que puede tener tres entradas ('quicksort', 'mergesort' o 'heapsort') del algoritmo utilizado para ordenar el DataFrame.
- **na\_position:** Toma dos entradas de cadena, 'last' o 'first' para establecer la posición de los valores nulos. El valor predeterminado es "last".

# Método `sort_values()`:

```
DF.sort_values("Estudiante", axis=0, ascending=True, inplace=True, na_position='last')
```

```
DF.sort_values("Estudiante")
```

	Estudiante	Edad	Promedio	Ciclo	Curso
0	Juan	19	15	V	Ingeniería de Datos
1	Carlos	20	12	VI	Base de Datos
2	Rosa	18	8	V	Algoritmos
3	Jean	18	18	IV	Tesis
4	Lourdes	20	10	IX	Proyecto de Tesis
5	Mario	18	17	V	Programación



	Estudiante	Edad	Promedio	Ciclo	Curso
1	Carlos	20	12	VI	Base de Datos
3	Jean	18	18	IV	Tesis
0	Juan	19	15	V	Ingeniería de Datos
4	Lourdes	20	10	IX	Proyecto de Tesis
5	Mario	18	17	V	Programación
2	Rosa	18	8	V	Algoritmos

# Método `sort_values()`:

```
df.sort_values("Edad")
```

	Edad	Renuncia	ViajesNegocios	TarifaDiaria	Departamento
458	18	Yes	Travel_Frequently	1306	Sales
728	18	No	Non-Travel	287	Research & Development
302	18	No	Travel_Rarely	812	Sales
1312	18	No	Non-Travel	1431	Research & Development
973	18	No	Non-Travel	1124	Research & Development
...	...	...	...	...	...
537	60	No	Travel_Rarely	1179	Sales
1210	60	No	Travel_Rarely	370	Research & Development
428	60	No	Travel_Frequently	1499	Sales
880	60	No	Travel_Rarely	696	Sales
412	60	No	Travel_Rarely	422	Research & Development

1480 rows × 35 columns

# Método `sort_values()`:

Ordenar por varias columnas:

```
df.sort_values(["Edad", "Profesion"], axis=0, ascending=True, inplace=True)  
df
```

	Edad	Renuncia	ViajesNegocios	TarifaDiaria	Departamento	DistanciaDesdeCasa	NivelEducacion	Profesion
297	18	Yes	Travel_Rarely	230	Research & Development	3.0	3	Life Sciences
728	18	No	Non-Travel	287	Research & Development	5.0	2	Life Sciences
973	18	No	Non-Travel	1124	Research & Development	1.0	3	Life Sciences
458	18	Yes	Travel_Frequently	1306	Sales	5.0	3	Marketing
829	18	Yes	Non-Travel	247	Research & Development	8.0	1	Medical
...	...	...	...	...	...	...	...	...
412	60	No	Travel_Rarely	422	Research & Development	7.0	3	Life Sciences
428	60	No	Travel_Frequently	1499	Sales	28.0	3	Marketing
537	60	No	Travel_Rarely	1179	Sales	16.0	4	Marketing
880	60	No	Travel_Rarely	696	Sales	7.0	4	Marketing
1210	60	No	Travel_Rarely	370	Research & Development	1.0	4	Medical

# Método `sort_values()`:

Ordenar de manera Descendente la Edad y por Orden Ascendente la Profesión:

```
df.sort_values(["Edad", "Profesion"], axis=0, ascending=[False, True], inplace=True)  
df
```

	Edad	Renuncia	ViajesNegocios	TarifaDiaria	Departamento	DistanciaDesdeCasa	NivelEducacion	Profesion
412	60	No	Travel_Rarely	422	Research & Development	7.0	3	Life Sciences
428	60	No	Travel_Frequently	1499	Sales	28.0	3	Marketing
537	60	No	Travel_Rarely	1179	Sales	16.0	4	Marketing
880	60	No	Travel_Rarely	696	Sales	7.0	4	Marketing
1210	60	No	Travel_Rarely	370	Research & Development	1.0	4	Medical
...	...	...	...	...	...	...	...	...
458	18	Yes	Travel_Frequently	1306	Sales	5.0	3	Marketing
829	18	Yes	Non-Travel	247	Research & Development	8.0	1	Medical
1312	18	No	Non-Travel	1431	Research & Development	14.0	3	Medical
302	18	No	Travel_Rarely	812	Sales	10.0	3	Medical
1154	18	Yes	Travel_Frequently	544	Sales	3.0	2	Medical



# Práctica



Ordenar por 3 Columnas:

- **Edad:** Ascendente
- **Departamento:** Descendente
- **Profesión:** Ascendente

# CASO PRÁCTICO: Agregación

# AGREGACIÓN

Pandas nos proporciona una variedad de funciones agregadas . Estas funciones ayudan a realizar diversas actividades en los conjuntos de datos. Estas funciones son:



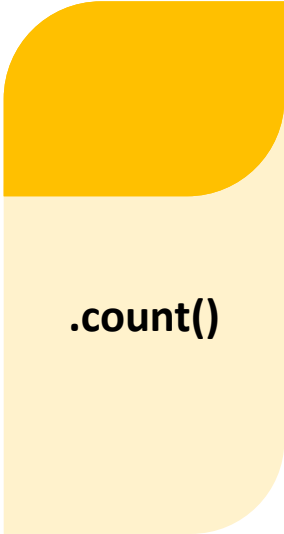
**.count()**

**.sum()**

**.min()  
.max()**

**.mean()  
.median()**

# AGREGACIÓN



**.count()**

# Método `count()`:

**DataFrame.count()**: Muestra la cantidad total de elementos que no son NaN de una columna del DataFrame.

```
df[['Departamento']].count()
```

```
Departamento    1485  
dtype: int64
```

```
df.count()
```

```
Edad                1485  
Renuncia            1484  
ViajesNegocios      1477  
TarifaDiaria        1485  
Departamento       1484  
DistanciaDesdeCasa  1476  
NivelEducacion      1485  
Profesion           1480  
RecuentoEmpleados   1484  
NumeroEmpleado      1485  
SatisfaccionAmbienteTrabajo 1485  
Genero              1478  
TarifaPorHora       1485  
ParticipacionTrabajo 1485  
NivelTrabajo        1485  
RolTrabajador       1483  
Satisfaccionlaboral  1485  
EstadoCivil         1475  
IngresosMensuales   1485  
TarifaMensual        1485  
NumeroEmpresasTrabajo 1485  
MayorDe18           1485  
TiempoExtra         1477  
PorcentajeAumentoSalarial 1485  
CalificacionRendimiento 1485  
SatisfaccionRelacionLaboral 1485  
HorasEstandar       1485  
NivelParticipacionAcciones 1485  
AñosLaboralesTotales 1485  
NroCapacitacionUltimoAño 1485  
EquilibrioVidaLaboral 1485  
AñosEmpresa         1485  
AñosRolActual        1483  
AñosDesdeUltimaPromocion 1484  
AñosComoJefe         1485  
dtype: int64
```

# Método `count()`:

Contar los valores de las filas.

```
df.count(axis=1)
```

```
0      35
1      35
2      35
3      35
4      33
..
1480   33
1481   32
1482   31
1483   31
1484   31
Length: 1485, dtype: int64
```

```
df.count(axis="columns")
```

# Método `count()`:

Contar los empleados mayores de 50 años y mostrarlos.

```
Empleados = df[df['Edad']>50]['Renuncia'].count()
print ("Empleados mayores de 50 años: ", Empleados)
```

Empleados mayores de 50 años: 143

Mostramos los Empleados

```
df[df['Edad']>50]
```

	Edad	Renuncia	ViajesNegocios	TarifaDiaria	Departamento
7	59	No	Travel_Rarely	1324	Research & Development
19	53	No	Travel_Rarely	1219	Sales
26	53	No	Travel_Rarely	1282	Research & Development
64	59	No	Travel_Rarely	1435	Sales
66	55	No	Travel_Rarely	836	Research & Development
...	...	...	...	...	...
1402	55	No	Travel_Rarely	189	Human Resources
1407	54	No	Travel_Rarely	157	Research & Development
1435	52	No	Non-Travel	585	Sales
1442	56	No	Non-Travel	667	Research & Development
1445	56	Yes	Travel_Rarely	310	Research & Development

143 rows × 35 columns

# Método `value_counts()`:

**DataFrame.value\_counts()**: Muestra el número de veces que se repite un valor.

```
df[['Departamento']].value_counts()
```

```
Departamento
Research & Development    975
Sales                    447
Human Resources           63
Name: count, dtype: int64
```

```
df[['Edad']].value_counts()
```

```
Edad
35    83
34    77
31    70
36    69
29    68
32    61
33    60
30    60
38    58
40    57
37    52
27    48
28    48
42    46
45    42
39    42
41    40
26    39
44    33
46    33
43    32
50    30
24    26
25    26
47    26
49    24
55    22
48    19
53    19
51    19
52    18
22    18
54    18
56    14
58    14
23    14
21    13
20    11
59    10
19     9
18     8
60     5
57     4
Name: count, dtype: int64
```



# Práctica



Contar el total de empleados mayores de 30 años, que trabajen en el departamento de Recursos Humanos y que tengas menos de 6 años como jefe.

Mostrar a los empleados.

```
Empleados = df[(df['Edad']>30) &
                (df['Departamento']=='Human Resources') &
                (df['AñosComoJefe']<6)][['Edad']].count()
print ("El número de Empleados: ", Empleados)
```

El número de Empleados: 31

```
# Mostramos los empleados
df[(df['Edad']>30) &
   (df['Departamento']=='Human Resources') &
   (df['AñosComoJefe']<6)]
```



	Edad	Renuncia	ViajesNegocios	TarifaDiaria	Departamento	DistanciaDesdeCasa	NivelEducacion
<b>80</b>	46	No	Travel_Rarely	945	Human Resources	5.0	2
<b>101</b>	37	Yes	Travel_Rarely	807	Human Resources	6.0	4
<b>106</b>	59	No	Non-Travel	1420	Human Resources	2.0	4
<b>113</b>	54	No	Non-Travel	142	Human Resources	26.0	3
<b>233</b>	59	No	Travel_Rarely	818	Human Resources	6.0	2
<b>311</b>	31	No	Travel_Rarely	106	Human Resources	2.0	3

31 rows × 35 columns

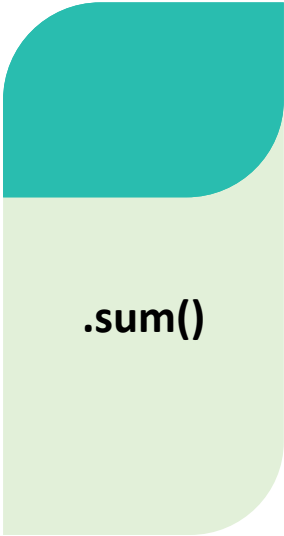
# Práctica



Mostrar cuantos Empleados son:

- Solteros
- Casados
- Divorciados.

# AGREGACIÓN



`.sum()`

# Método `sum()`:

**DataFrame.sum():** Devuelve la suma de los valores de una columna del DataFrame, cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena .

Suma de los valores de una columna específica

```
df[['TarifaDiaria']].sum()
```

```
TarifaDiaria    1193397
dtype: int64
```

```
df.sum()
```

```
Edad                                                    54807
Renuncia                                                YesNoYesNoYesNoNoNoNoNoNoNoNoNoNoNoNoNoNoNoNo...
ViajesNegocios                                          Travel_RarelyTravel_FrequentlyTravel_RarelyTra...
TarifaDiaria                                            1193397
Departamento                                           SalesResearch & DevelopmentResearch & Developm...
DistanciaDesdeCasa                                    13546.0
NivelEducacion                                         4326
Profesion                                               Life SciencesLife SciencesOtherLife SciencesOt...
RecuentoEmpleados                                     1485.0
NumeroEmpleado                                         1527011
SatisfaccionAmbienteTrabajo                            4046
Genero                                                  FemaleMaleMaleFemaleMaleMaleMaleFemaleMaleMale...
TarifaPorHora                                           97821
ParticipacionTrabajo                                   4056
NivelTrabajo                                            3063
RolTrabajador                                           Sales ExecutiveResearch ScientistLaboratory Te...
SatisfaccionLaboral                                    4053
EstadoCivil                                             SingleMarriedSingleMarriedSingleMarriedSingleM...
IngresosMensuales                                     9652980
TarifaMensual                                           21317557
NumeroEmpresasTrabajo                                  3990
MayorDel18                                             YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY...
TiempoExtra                                             YesNoYesYesYesNoNoYesNoNoNoNoYesNoNoYesNoYesYe...
PorcentajeAumentoSalarial                             22583
CalificacionRendimiento                                4682
SatisfaccionRelacionLaboral                           4020
HorasEstandar                                          118800
NivelParticipacionAcciones                             1175
AñosLaboralesTotales                                  16734
NroCapacitacionUltimoAño                               4166
EquilibrioVidaLaboral                                  4104
AñosEmpresa                                             10407
AñosRolActual                                           6291.0
AñosDesdeUltimaPromocion                               3256.0
AñosComoJefe                                             6129
dtype: object
```

# Método `cumsum()`:

**DataFrame.cumsum()**: Devuelve la suma acumulada de los valores de una columna del DataFrame, cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena .

Suma acumulada de los valores  
de una columna específica

```
df[['Edad']].cumsum()
```

	Edad
0	41
1	90
2	127
3	160
4	197
...	...
1480	54682
1481	54717
1482	54739
1483	54774
1484	54807

1485 rows × 1 columns

```
df.cumsum()
```

	Edad	Renuncia
0	41	Yes
1	90	YesNo
2	127	YesNoYes
3	160	YesNoYesNo
4	197	YesNoYesNoYes
...	...	...
1480	54682	YesNoYesNoYesNoNoNoNoNoNoNoNoNoYesNoNoNoNoNo...
1481	54717	YesNoYesNoYesNoNoNoNoNoNoNoNoNoYesNoNoNoNoNo...
1482	54739	YesNoYesNoYesNoNoNoNoNoNoNoNoNoYesNoNoNoNoNo...

# AGREGACIÓN

A vertical rounded rectangle with a purple top half and a light purple bottom half. The text `.min()` and `.max()` is centered in the light purple section.

`.min()`  
`.max()`

# Método `min()`:

**DataFrame.min():** Encuentra el valor mínimo de los valores de una columna del DataFrame.

Valor mínimo de una columna específica

```
df[['Edad']].min()
```

```
Edad    18  
dtype: int64
```

```
df.min()
```

```
Edad    18  
Renuncia    No  
ViajesNegocios    Non-Travel  
TarifaDiaria    102  
Departamento    Human Resources  
DistanciaDesdeCasa    1.0  
NivelEducacion    1  
Profesion    Human Resources  
RecuentoEmpleados    1.0  
NumeroEmpleado    1  
SatisfaccionAmbienteTrabajo    1  
Genero    Female  
TarifaPorHora    30  
ParticipacionTrabajo    1  
NivelTrabajo    1  
RolTrabajador    Healthcare Representative  
SatisfaccionLaboral    1  
EstadoCivil    Divorced  
IngresosMensuales    1009  
TarifaMensual    2094  
NumeroEmpresasTrabajo    0  
MayorDe18    Y  
TiempoExtra    No  
PorcentajeAumentoSalarial    11  
CalificacionRendimiento    3  
SatisfaccionRelacionLaboral    1  
HorasEstandar    80  
NivelParticipacionAcciones    0  
AñosLaboralesTotales    0  
NroCapacitacionUltimoAño    0  
EquilibrioVidaLaboral    1  
AñosEmpresa    0  
AñosRolActual    0.0  
AñosDesdeUltimaPromocion    0.0  
AñosComoJefe    0  
dtype: object
```



# Método `max()`:

**DataFrame.`max()`:** Encuentra el valor máximo de los valores de una columna del DataFrame.

Valor máximo de una columna específica

```
edadMaxima = df[['Edad']].max()

print("La edad máxima de todos los empleados es: ", edadMaxima)

La edad máxima de todos los empleados es: Edad    60
dtype: int64
```

`df.max()`

```
Edad    60
Renuncia    Yes
ViajesNegocios    Travel_Rarely
TarifaDiaria    1499
Departamento    Sales
DistanciaDesdeCasa    29.0
NivelEducacion    5
Profesion    Technical Degree
RecuentoEmpleados    1.0
NumeroEmpleado    2068
SatisfaccionAmbienteTrabajo    4
Genero    Male
TarifaPorHora    100
ParticipacionTrabajo    4
NivelTrabajo    5
RolTrabajador    Sales Representative
SatisfaccionLaboral    4
EstadoCivil    Single
IngresosMensuales    19999
TarifaMensual    26999
NumeroEmpresasTrabajo    9
MayorDe18    Y
TiempoExtra    Yes
PorcentajeAumentoSalarial    25
CalificacionRendimiento    4
SatisfaccionRelacionLaboral    4
HorasEstandar    80
NivelParticipacionAcciones    3
AñosLaboralesTotales    40
NroCapacitacionUltimoAño    6
EquilibrioVidaLaboral    4
AñosEmpresa    40
AñosRolActual    18.0
AñosDesdeUltimaPromocion    15.0
AñosComoJefe    17
dtype: object
```

# AGREGACIÓN

**.mean()**  
**.median()**

# Método `mean()`:

**DataFrame.`mean()`:** Calcula el promedio de una columna del DataFrame.

```
df[['Edad']].mean()
```

```
Edad    36.907071  
dtype: float64
```

```
df[['AñosComoJefe']].mean()
```

```
AñosComoJefe    4.127273  
dtype: float64
```

# Método `median()`:

**DataFrame.`median()`:** Calcula la mediana de los valores de una columna del DataFrame.

```
df[['Edad']].median()
```

```
Edad    36.0  
dtype: float64
```

```
df[['TarifaMensual']].median()
```

```
TarifaMensual    14369.0  
dtype: float64
```

# CASO PRÁCTICO: Agrupación

# Método `groupby()`:

**DataFrame.groupby():** Divide los datos de un DataFrame en grupos según algunos criterios. Esto permite implementar varias funciones sobre ellos.

```
df.groupby("Departamento")
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001E64D895A50>
```

# Método `groupby()`:

`DataFrame.groupby('Columnas').groups`: Devuelve un diccionario que resultan de todas las combinaciones de los valores de las columnas con nombres en la lista `columnas`, y de los valores de las listas de los nombres de las filas del DataFrame.

```
df.groupby("Departamento").groups
```

```
{'Human Resources': [80, 101, 106, 113, 135, 140, 233, 311, 351, 423, 441, 454, 478, 494, 511, 536, 539, 552, 600, 614, 630, 634, 655, 656, 760, 790, 827, 836, 864, 879, 924, 944, 957, 963, 1000, 1006, 1036, 1040, 1065, 1097, 1108, 1155, 1166, 1201, 1223, 1229, 1244, 1246, 1247, 1270, 1281, 1290, 1298, 1313, 1314, 1324, 1330, 1348, 1380, 1401, 1402, 1412, 1451], 'Research & Development': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 23, 24, 25, 26, 27, 29, 31, 32, 33, 35, 36, 39, 41, 42, 43, 45, 46, 48, 50, 51, 52, 54, 56, 58, 59, 60, 61, 62, 63, 65, 66, 67, 68, 69, 70, 72, 73, 74, 75, 76, 78, 79, 81, 82, 84, 85, 86, 88, 89, 91, 94, 96, 100, 102, 103, 104, 105, 107, 109, 110, 111, 112, 114, 115, 117, 119, 121, 123, 124, 126, 127, 129, 130, 131, 136, 137, 141, 142, 143, 144, ...], 'Sales': [0, 19, 22, 28, 30, 34, 37, 38, 40, 44, 47, 49, 53, 55, 57, 64, 71, 77, 83, 87, 90, 92, 93, 95, 97, 98, 99, 108, 116, 118, 120, 122, 125, 128, 132, 133, 134, 138, 139, 145, 152, 153, 155, 159, 160, 168, 169, 172, 175, 179, 183, 206, 211, 213, 216, 217, 219, 220, 224, 227, 228, 229, 234, 236, 238, 239, 242, 255, 262, 264, 266, 274, 278, 282, 283, 293, 294, 296, 298, 301, 302, 304, 307, 320, 321, 322, 328, 329, 332, 336, 339, 340, 344, 348, 350, 353, 355, 356, 358, 359, ...]}
```

Human Resources

Research & Development

Sales

# Método `groupby()`:

`DataFrame.groupby('Columnas').get_group(valores):`

Devuelve un DataFrame con las filas del DataFrame de un grupo concreto.

```
df.groupby("Departamento").get_group('Human Resources')
```

	Edad	Renuncia	ViajesNegocios	TarifaDiaria	Departamento
80	46	No	Travel_Rarely	945	Human Resources
101	37	Yes	Travel_Rarely	807	Human Resources
106	59	No	Non-Travel	1420	Human Resources
113	54	No	Non-Travel	142	Human Resources
135	26	No	Travel_Rarely	1355	Human Resources
...	...	...	...	...	...
1380	27	Yes	Travel_Frequently	1337	Human Resources
1401	38	No	Travel_Frequently	1444	Human Resources
1402	55	No	Travel_Rarely	189	Human Resources
1412	25	No	Travel_Rarely	309	Human Resources
1451	35	No	Travel_Rarely	1146	Human Resources

63 rows × 35 columns



# Práctica



Crear agrupaciones por los valores de la columna Estado Civil.

Mostrar el grupo de empleados divorciados.

# APLICAR FUNCIONES DE AGREGACIÓN

Obtener el Promedio por Departamento de todas las columnas que tienen valores numéricos

```
df.groupby("Departamento").mean(numeric_only = True)
```

	Edad	TarifaDiaria	DistanciaDesdeCasa	NivelEducacion
Departamento				
Human Resources	37.809524	751.539683	8.698413	2.968254
Research & Development	37.017436	808.565128	9.082051	2.898462
Sales	36.539150	800.221477	9.346756	2.937360

3 rows × 26 columns

# APLICAR FUNCIONES DE AGREGACIÓN

Obtener la Desviación Estandar por Departamento de todas las columnas que tienen valores numéricos

```
df.groupby("Departamento").std(numeric_only = True)
```

	Edad	TarifaDiaria	DistanciaDesdeCasa	NivelEducacion
Departamento				
Human Resources	9.226290	426.203943	8.115368	0.983218
Research & Development	9.157682	403.487173	8.084731	1.026821
Sales	9.022774	402.347292	8.087422	1.031180

3 rows × 26 columns

# APLICAR FUNCIONES DE AGREGACIÓN

Agrupar por Departamento y devolver el valor mínimo de todas las columnas que tienen valores numéricos

```
df.groupby("Departamento").min(numeric_only = True)
```

	Edad	TarifaDiaria	DistanciaDesdeCasa	NivelEducacion
Departamento				
Human Resources	19	106	1.0	1
Research & Development	18	102	1.0	1
Sales	18	107	1.0	1

3 rows × 26 columns

Agrupar por Departamento y devolver el valor máximo de todas las columnas que tienen valores numéricos

```
df.groupby("Departamento").max(numeric_only = True)
```

	Edad	TarifaDiaria	DistanciaDesdeCasa	NivelEducacion
Departamento				
Human Resources	59	1444	26.0	5
Research & Development	60	1496	29.0	5
Sales	60	1499	29.0	5

3 rows × 26 columns

# APLICAR FUNCIONES DE AGREGACIÓN

Obtener la Suma por Departamento de todas las columnas que tienen valores numéricos

```
df.groupby("Departamento").sum(numeric_only = True)
```

	Edad	TarifaDiaria	DistanciaDesdeCasa	NivelEducacion	RecuentoEmpleados
Departamento					
Human Resources	2382	47347	548.0	187	63.0
Research & Development	36092	788351	8855.0	2826	975.0
Sales	16333	357699	4178.0	1313	447.0

3 rows × 6 columns

# APLICAR FUNCIONES DE AGREGACIÓN

Utilizando el método `df.groupby().describe()`

```
df.groupby("Departamento").describe()
```

		Edad							
		count	mean	std	min	25%	50%	75%	max
Departamento									
Human Resources		63.0	37.809524	9.226290	19.0	30.5	37.0	44.0	59.0
Research & Development		975.0	37.017436	9.157682	18.0	30.0	36.0	43.0	60.0
Sales		447.0	36.539150	9.022774	18.0	30.0	35.0	42.0	60.0

3 rows × 208 columns

# APLICAR FUNCIONES DE AGREGACIÓN

Utilizando el método `df.groupby().describe().transpose()`

```
df.groupby("Departamento").describe().transpose()
```

	Departamento	Human Resources	Research & Development	Sales
Edad	count	63.000000	975.000000	447.000000
	mean	37.809524	37.017436	36.539150
	std	9.226290	9.157682	9.022774
	min	19.000000	18.000000	18.000000
	25%	30.500000	30.000000	30.000000
...	...	...	...	...
AñosComoJefe	min	0.000000	0.000000	0.000000
	25%	2.000000	2.000000	2.000000
	50%	3.000000	3.000000	3.000000
	75%	6.500000	7.000000	7.000000
	max	10.000000	17.000000	17.000000

208 rows × 3 columns

# APLICAR FUNCIONES DE AGREGACIÓN

Utilizando el método `df.groupby().describe().transpose()`, para mostrar la información de un departamento en particular

```
df.groupby("Departamento").describe().transpose()['Sales']
```

```
Edad      count    447.000000  
          mean     36.539150  
          std       9.022774  
          min     18.000000  
          25%     30.000000  
          ...  
AñosComoJefe  min      0.000000  
              25%      2.000000  
              50%      3.000000  
              75%      7.000000  
              max     17.000000  
Name: Sales, Length: 208, dtype: float64
```



# APLICAR FUNCIONES DE AGREGACIÓN

## Método `groupby().agg()`:

**DataFrame.** `groupby(columnas).agg(funciones)`: Devuelve un DataFrame con el resultado de aplicar las funciones de agregación a cada uno de los DataFrames que resultan de dividir el DataFrame.

```
df.groupby("Departamento").agg(['min'])
```

Departamento	Edad	Renuncia	ViajesNegocios	TarifaDiaria	DistanciaDesdeCasa
	min	min	min	min	min
Human Resources	19	No	Non-Travel	106	1.0
Research & Development	18	No	Non-Travel	102	1.0
Sales	18	No	Non-Travel	107	1.0

3 rows × 34 columns

```
df.groupby("Departamento").agg(['min', 'max'])
```

Departamento	Edad		Renuncia		ViajesNegocios		TarifaDiaria	
	min	max	min	max	min	max	min	max
Human Resources	19	59	No	Yes	Non-Travel	Travel_Rarely	106	1444
Research & Development	18	60	No	Yes	Non-Travel	Travel_Rarely	102	1496
Sales	18	60	No	Yes	Non-Travel	Travel_Rarely	107	1499

3 rows × 68 columns

# Práctica



Agrupar el DataFrame por Profesión y aplicar las funciones de agregación.



# ¡Gracias!

**UPAO**  
UNIVERSIDAD PRIVADA ANTENOR ORREGO