# Ejercitación módulo Pandas

Módulo Pandas - Parte práctica

**Profesor Adjunto:** Mag. Bioing. Baldezzari Lucas

V2022

---

## 1. Primeros pasos con Pandas

En este ejercicio trabajará con datos estimados de personas en condición de calle en 2018 en algunos estados de Estados Unidos.

- La columna *individuos* representa personas que viven solas.
- La columna *miembrosFamilia* es el número de personas en condición de calle que poseen hijos/as.
- La columna *poblacionTotal* representa el total de personas que habitan en un estado.

### 1.1 Cargando un DataFrame y chequeando datos

- A) Usted dispone de un archivo llamado *sinhogares.csv* dentro de la carpeta */datasets*. Cargue el archivo en un dataframe llamado *sinhogares*
- B) Utilice los siguientes métodos para conocer un poco más los datos cargados.
  - .head()
  - .info()
  - .describe()
  - .shape
- C) En base a lo anterior, ¿cuantas observaciones/filas posee el data set?

In [4]:
```
## TODO 1.1
#Punto A
import pandas as pd

sh = pd.read_csv("datasets/sinhogares.csv")
sh.shape

#Punto b
sh.head()
sh.info()
sh.describe()
sh.shape

#Punto C
#Posee 51 filas ashee
sh.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Unnamed: 0       51 non-null     int64
```

```
 1   region           51 non-null     object
 2   estado           51 non-null     object
 3   individuos       51 non-null     int64
 4   miembrosFamilia  51 non-null     int64
 5   poblacionTotal   51 non-null     int64
dtypes: int64(4), object(2)
memory usage: 2.5+ KB
```

Out[4]:
```
(51, 6)
```

## 1.2 Valores, columnas e índices del dataframe

Para entender mejor al dataframe investigue los atributos.

- .values
- .columns
- .index

¿Cuales son las columnas que forman el DataFrame? ¿Cómo son los índices (numéricos o tienen nombre)?

In [16]:
```
## TODO 1.2
sh.values
sh.columns #Columnas del DataFrame
```

Out[16]:
```
Index(['Unnamed: 0', 'region', 'estado', 'individuos', 'miembrosFamilia',
       'poblacionTotal'],
      dtype='object')
```

In [17]:
```
sh.index #Indices númericos
```

Out[17]:
```
RangeIndex(start=0, stop=51, step=1)
```

## 1.3 Ordenando el dataframe

- A) Ordene los datos dentro del dataframe *sinhogares* por la columna *individuos* y guardela en un archivo llamado *sinhogares_Ind*. Imprima los primeros 5 datos de *sinhogares_Ind*.
- B) En un dataframe llamado *sinhogares_Fam* guarde el dataframe original pero ordenado por la columna *miembrosFamilia* de mayor a menor. Imprima los primeros 5 datos de *sinhogares_Fam*.
- C) Ordene los datos del dataframe primero por *region* (de forma ascendente) y luego por *miembrosFamilia* (de manera descendente). Guarde los datos en un dataframe e imprima los primeros cinco valores.

In [15]:
```
## TODO 1.3
#Punto A
sinhogares_Ind = sh.sort_values("individuos", ascending = True).head(5)

#Punto B
sinhogares_Fam = sh.sort_values("miembrosFamilia", ascending = True).head(5)

#PuntoC
sinhogares_reg_fam = sh.sort_values(["region","miembrosFamilia"], ascending = [True,Fa
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
```

```
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Unnamed: 0       51 non-null     int64
 1   region           51 non-null     object
 2   estado           51 non-null     object
 3   individuos       51 non-null     int64
 4   miembrosFamilia  51 non-null     int64
 5   poblacionTotal   51 non-null     int64
dtypes: int64(4), object(2)
memory usage: 2.5+ KB
```

## 1.4 Obteniendo subsets a partir de columnas

Recuerde que podemos obtener trozos de nuestro dataframe haciendo

```
subset1 = dataframe["columnaInteres"]
subset1 = dataframe[["columnaInteres","otraColumna"]]
```

- A partir del dataframe *sinhogares* cree un dataframe llamado *indiv_estados* que este formado por las columnas *individuos* y *poblacionTotal*. Imprima el head del dataframe.

In [19]:
```
## TODO 1.4
indiv_estados = sh[["individuos","poblacionTotal"]].head(5)
indiv_estados
```

Out[19]:

| | individuos | poblacionTotal |
|---|---|---|
| **0** | 2570 | 4887681 |
| **1** | 1434 | 735139 |
| **2** | 7259 | 7158024 |
| **3** | 2280 | 3009733 |
| **4** | 109008 | 39461588 |

## 1.5 Obteniendo subsets a partir de filas aplicando filtros

- A) Cree un dataframe que contenga los datos de las personas en condición de calle que superen los 10mil individuos por estado.
- B) Cree un dataframe que contenga los datos de las personas en condición de calle menores a los mil individuos y que correspondan a la región "Pacific". Imprima el dataframe creado.

In [37]:
```
## TODO 1.5
#Punto A

filtro1 = sh["individuos"]>10000
filtro1

#Punto B
filtro2 = sh[(sh["individuos"]<1000) & (sh["region"]=="Pacific")]
filtro2
```

Out[37]:

| | Unnamed: 0 | region | estado | individuos | miembrosFamilia | poblacionTotal |
|---|---|---|---|---|---|---|

In [36]:

```
#A
filtro1=sh["individuos"]>10000
filtro1
#B
filtro2=sh["region"] == "Pacific"
filtro2
subSet2Filtros = sh[ filtro1 & filtro2 ]
subSet2Filtros
```

Out[36]:

| | Unnamed: 0 | region | estado | individuos | miembrosFamilia | poblacionTotal |
|---|---|---|---|---|---|---|
| **4** | 4 | Pacific | California | 109008 | 20964 | 39461588 |
| **37** | 37 | Pacific | Oregon | 11139 | 3337 | 4181886 |
| **47** | 47 | Pacific | Washington | 16424 | 5880 | 7523869 |

## 1.6 Obteniendo subsets a partir de datos categóricos

Utilizando *.isin()* se pide,

- A) Cree un subset que contenga los datos para las regiones *South Atlantic* y *Mid-Atlantic*. Imprima el subset.
- B) Cree un subset que contenga los datos para los estados *California*, *Arizona*, *Nevada* y *Utah*.

In [52]:

```
## TODO 1.6

#PuntoA
regionFil = sh["region"].isin(["South Atlantic", "Mid-Atlantic"]) #Se que es un subset
region = sh[regionFil]
#region
print(region)

estadosFil = sh["estado"].isin(["California", "Arizona", "Nevada" , "Utah"])
estados = sh[estadosFil]
#estados
print(estados)
```

```
    Unnamed: 0          region                estado  individuos  \
7            7  South Atlantic              Delaware         708
8            8  South Atlantic  District of Columbia        3770
9            9  South Atlantic               Florida       21443
10          10  South Atlantic               Georgia        6943
20          20  South Atlantic              Maryland        4914
30          30    Mid-Atlantic            New Jersey        6048
32          32    Mid-Atlantic              New York       39827
33          33  South Atlantic        North Carolina        6451
38          38    Mid-Atlantic          Pennsylvania        8163
40          40  South Atlantic        South Carolina        3082
46          46  South Atlantic              Virginia        3928
48          48  South Atlantic         West Virginia        1021

    miembrosFamilia  poblacionTotal
7               374          965479
8              3134          701547
9              9587        21244317
10             2556        10511131
20             2230         6035802
30             3350         8886025
32            52070        19530351
33             2817        10381615
```

```
38             5349        12800922
40              851         5084156
46             2047         8501286
48              222         1804291
    Unnamed: 0    region      estado  individuos  miembrosFamilia  \
2            2  Mountain     Arizona        7259             2606
4            4   Pacific  California      109008            20964
28          28  Mountain      Nevada        7058              486
44          44  Mountain        Utah        1904              972

    poblacionTotal
2          7158024
4         39461588
28         3027341
44         3153550
```

# 2. Un poco de estadística

Para este ejercicio se trabajará con los datos de venta de la cadena de supermercados *Walmart* que van desde enero de 2010 a diciembre de 2012.

Las columnas dentro del set de datos representa:

- Store: Numero de supermercado
- Type: Tipo de tienda.
- Department: Departamento donde se realizó la venta.
- Date: Semana de venta
- weekly_sales: Ventas semanales
- is_holiday: Indica si esa semana hubieron días festivos.
- Temperature: Temperatura del día de las ventas.
- Fuel_Price: Costo del combustible en la región
- Unemployment: Representa el porcentaje de la fuerza laboral sin trabajo.

Cargamos el archivo *walmart.csv* (se encuentra dentro del directorio */datastes/*) en un dataframe llamado *walmart* y vemos su encabezado

In [134]:
```
walmart = pd.read_csv("datasets/walmart.csv")
walmart.head()
```

Out[134]:

| | store | type | department | date | weekly_sales | is_holiday | temperature | fuel_price | unempl |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | A | 1 | 2/5/2010 | 24924.50 | False | 5.727778 | 0.679451 | |
| **1** | 1 | A | 1 | 3/5/2010 | 21827.90 | False | 8.055556 | 0.693452 | |
| **2** | 1 | A | 1 | 4/2/2010 | 57258.43 | False | 16.816667 | 0.718284 | |
| **3** | 1 | A | 1 | 5/7/2010 | 17413.94 | False | 22.527778 | 0.748928 | |
| **4** | 1 | A | 1 | 6/4/2010 | 17558.09 | False | 27.050000 | 0.714586 | |

## 2.1 Medias y medianas

- A) ¿Cuál es el promedio de ventas semanales totales? ¿Cuál es la mediana?
- B) Obtenga el promedio de ventas semanales para cada año del data set. Idem para la mediana.

- C) ¿Cuando se hizo la última venta según este data set? ¿Y cuando fue la primera?

In [62]:
```
## TODO 2.1
#A ¿Cuál es el promedio de ventas semanales totales? ¿Cuál es la mediana?
promedio = walmart["weekly_sales"].mean()
mediana = walmart["weekly_sales"].median()
print(f"El promedio de ventas semanales totales es: {promedio}, y la mediana de ventas

#B - Obtenga el promedio de ventas semanales para cada año del data set. Idem para la
walmart['year'] = pd.DatetimeIndex(walmart['date']).year
promedio2=walmart[["year","weekly_sales"]].groupby("year").agg(["mean","median"])
promedio2

#C ¿Cuando se hizo la última venta según este data set? ¿Y cuando fue la primera?
walmart.sort_values("date", ascending=True).head(1) #Primera Venta
walmart.sort_values("date", ascending=False).head(1) #Ultima Venta
```

```
El promedio de ventas semanales totales es: 23843.95014850566, y la mediana de ventas
es : 12049.064999999999
```

Out[62]:

| | store | type | department | date | weekly_sales | is_holiday | temperature | fuel_price | une |
|---|---|---|---|---|---|---|---|---|---|
| **6820** | 19 | A | 48 | 9/9/2011 | 197.0 | True | 20.155556 | 1.038197 | |

## 2.2 Rango intercuartil

- Defina una función llamada *iqr(columna)* que devuelva el rango intercuartil (IQR). Esta función recibirá una *columna* de un dataframe con valores numéricos a partir de los cuales obtendrá los cuartiles $Q_1$ y $Q_3$ para devolver el $IQR = Q_3 - Q_1$.
- Luego con la función *.agg()* aplique las funciones *iqr()* y *np.median()* a las columnas *temperature*, *fuel_price* y *unemployment* del set de datos. Imprima el dataframe obtenido.

In [69]:
```
## TODO 2.2
import numpy as np
def irq(columna):
    return columna.quantile(0.75) - columna.quantile(0.25)

print(walmart[['temperature',"fuel_price", "unemployment"]].agg([irq,np.median])) ##Im
```

```
        temperature  fuel_price  unemployment
irq       16.583333    0.073176         0.565
median    16.966667    0.743381         8.099
```

## 2.3 Eliminando duplicados y contando

Utilizando *.drop_duplicates()* se pide:

- A) Elimine los datos duplicados a partir de las columnas *store* y *type*. Almacene el dataframe en *tiposTiendas*.
- B) Elimine los datos duplicados a partir de las columnas *store* y *departament*. Almacene el dataframe en *deptsTiendas*.
- C) Genere un subset llamado *fechaVacaciones* a partir de filtrar los datos en donde los valores de la columna *is_holiday* son True y remueva los duplicados a partir de la columna *date*.

Imprima los dataframe creados.

In [142]:
```
## TODO 2.3


tiposTiendas =  walmart.drop_duplicates(subset = ["store", "type"])
deptsTiendas = walmart.drop_duplicates(subset = ["store", "department"])
filtro1 = walmart["is_holiday"] == True
filtro2 = walmart["date"]

#fechaVacaciones = walmart.drop_duplicates(subset = [filtro1, filtro2])
```

## 2.4 Contando variables categóricas

A partir de los dataframe *tiposTiendas* y *deptsTiendas* se pide,

- A) Cuente cuántos tipos de tiendas hay dentro de *tiposTiendas*. También imprima las proporciones (usando *normalize = True*)
- B) Cuente cuántos departamentos diferentes tenemos en *deptsTiendas* y ordene los datos. También imprima las proporciones ordeandas.

In [146]:
```
## TODO 2.4
```

## 2.5 Proporciones de ventas por cada tipo de tienda

Calcule las proporciones de ventas semanales realizadas por cada tienda. Tenga en cuenta que las proporciones son el resultado de la venta de un tipo de tienda dividida las ventas totales.

Utilizando las herramientas vistas hasta ahora cuente cuantos tipos de tiendas hay en el set de datos. Luego utilice estos tipos para *filtrar* el dataset por cada tipo de tienda y use la función *.sum()* para sumar las ventas semanales del tipo particular de tienda selecionado. Finalmente divida cada una de las ventas semanales por las ventas totales.

Guarde las proporciones en una lista.

Ejemplo para tiendas del tipo A

```
ventasA = walmart[walmart["type"] == "A"]["weekly_sales"].sum()/totales
```

Donde *totales* es la suma de todas las ventas semanales del set de datos.

In [147]:
```
## TODO 2.5
walmart
```

Out[147]:

| | store | type | department | date | weekly_sales | is_holiday | temperature | fuel_price | u |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | A | 1 | 2/5/2010 | 24924.50 | False | 5.727778 | 0.679451 | |
| **1** | 1 | A | 1 | 3/5/2010 | 21827.90 | False | 8.055556 | 0.693452 | |
| **2** | 1 | A | 1 | 4/2/2010 | 57258.43 | False | 16.816667 | 0.718284 | |
| **3** | 1 | A | 1 | 5/7/2010 | 17413.94 | False | 22.527778 | 0.748928 | |
| **4** | 1 | A | 1 | 6/4/2010 | 17558.09 | False | 27.050000 | 0.714586 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **10769** | 39 | A | 99 | 12/9/2011 | 895.00 | False | 9.644444 | 0.834256 | |

| | store | type | department | date | weekly_sales | is_holiday | temperature | fuel_price | u |
|---|---|---|---|---|---|---|---|---|---|
| **10770** | 39 | A | 99 | 2/3/2012 | 350.00 | False | 15.938889 | 0.887619 | |
| **10771** | 39 | A | 99 | 6/8/2012 | 450.00 | False | 27.288889 | 0.911922 | |
| **10772** | 39 | A | 99 | 7/13/2012 | 0.06 | False | 25.644444 | 0.860145 | |
| **10773** | 39 | A | 99 | 10/5/2012 | 915.00 | False | 22.250000 | 0.955511 | |

10774 rows × 9 columns

# Función *.groupby()*

## 2.6 Haciendo cálculos con ***groupby()***

Implemente el punto 2.5 pero usando la función *groupby()*.

> Much easier! Isn't it?

In [13]:
```
## TODO 2.6
```

## 2.7 Datos a partir de agrupamiento múltiple

- A) Obtenga los mínimos, máximos, las medias (usando np.mean) y las medianas (usando np.median) de las ventas semanales para cada tipo de tienda.
- B) Obtenga los mínimos, máximos, las medias (usando np.mean) y las medianas (usando np.median) de las columnas *unemployment* y *fuel_price* para cada tipo de tienda.

**Hint:** Utilice *.groupby()* para agrupar datos y *.agg()* para aplicar las funciones.

In [14]:
```
## TODO 2.7
```

# Tablas dinámicas

En esta parte utilizaremos la función *.pivot_table()*.

## 2.8 Datos a partir de agrupamiento múltiple

- A) Obtenga la media y la mediana (con funciones de Numpy) de las ventas semanales por cada tipo de tienda usando *.pivot_table()* y almacene la tabla en *mediasYmedianas*.
- B) Use *.pivot_table()* para crear una tabla donde muestre las ventas promedios por cada tipo de tienda, donde las columnas correspondan a *is_holiday*. Guarde esta tabla en *mediasXVacaciones*.

In [79]:
```
## TODO 2.8
walmart
#mediasYmedianas = walmart.pivot_table(values = "weekly_sales", columns="type", aggfur
mediasYmedianas = walmart.pivot_table(values = "weekly_sales", columns="type", aggfunc
                                      index = "department")
mediasYmedianas

mediasXVacaciones = ___
```

Out[79]:

| | | | | | mean | | |
|---|---|---|---|---|---|---|---|
| type | A | B | C | D | A | B | |
| department | | | | | | | |
| 1 | 31190.167154 | 44050.626667 | NaN | 16113.010000 | 24780.020 | 31986.360 | N |
| 2 | 68084.234264 | 112958.526667 | 45800.135000 | 48754.470000 | 68976.370 | 112812.985 | 45800.1 |
| 3 | 17132.907344 | 30580.655000 | 40735.080000 | 10457.723333 | 13396.805 | 23145.625 | 40735.( |
| 4 | 44278.216378 | 51219.654167 | 54855.110000 | 37542.993333 | 42796.260 | 51485.930 | 54855.1 |
| 5 | 34842.088923 | 63236.875000 | 27107.950000 | 39793.990000 | 30299.045 | 60400.660 | 27107.9 |
| ... | ... | ... | ... | ... | ... | ... | |
| 95 | 123813.756538 | 77082.102500 | 131735.775000 | NaN | 123194.880 | 76416.640 | 131735.7 |
| 96 | 21367.042857 | 9528.538333 | NaN | NaN | 25187.875 | 9503.140 | N |
| 97 | 28510.668615 | 5828.873333 | 25910.160000 | NaN | 27240.870 | 5856.705 | 25910.1 |
| 98 | 12889.786357 | 217.428333 | 12257.806667 | NaN | 12651.700 | 34.100 | 12144.3 |
| 99 | 379.123659 | NaN | NaN | NaN | 167.000 | NaN | N |

80 rows × 8 columns

# 3. Temperatura a lo largo de los años

En esta parte trabajará con un set de datos que contiene valores de temperatura para diferentes países a lo largo de varios años. Usted cuenta con un archivo llamado *temperatures.csv* dentro de la carpeta *datasets/*.

In [84]:
```
t = pd.read_csv("datasets/temperatures.csv")
```

## 3.1 Datos básicos

Cargue el archivo en un dataframe llamado *temperaturas* y encuentre.

1. Los 5 países con mayores temperaturas promedios en los últimos 5 años.
2. Las 10 ciudades con mayores temperaturas promedios en los últimos 5 años.
3. El país que haya sufrido el mayor aumento de temperatura desde el primer registro hasta el último registro dentro del set de datos. ¿Cuantos grados en promedio ha aumentado o disminuído la temperatura este país?

In [88]:
```
## TODO 3.1
t
#t.sort_values("country", "avg_temp_c")
```

Out[88]:

| | date | city | country | avg_temp_c |
|---|---|---|---|---|
| 0 | 2000-01-01 | Abidjan | Costa de Marfil | 27.293 |
| 1 | 2000-02-01 | Abidjan | Costa de Marfil | 27.685 |
| 2 | 2000-03-01 | Abidjan | Costa de Marfil | 29.061 |

| | date | city | country | avg_temp_c |
|---|---|---|---|---|
| **3** | 2000-04-01 | Abidjan | Costa de Marfil | 28.162 |
| **4** | 2000-05-01 | Abidjan | Costa de Marfil | 27.547 |
| **...** | ... | ... | ... | ... |
| **16402** | 2013-04-01 | Xian | China | 12.563 |
| **16403** | 2013-05-01 | Xian | China | 18.979 |
| **16404** | 2013-06-01 | Xian | China | 23.522 |
| **16405** | 2013-07-01 | Xian | China | 25.251 |
| **16406** | 2013-08-01 | Xian | China | 24.528 |

16407 rows × 4 columns

## 3.2 Subdatos con .loc[ ]

1. Crear una lista llamada *ciudades* que contengan las ciudades de "Montreal" y "Toronto".
2. Utilizando corchetes cree un subset de datos filtrando con la lista creada anteriormente. Nota: Puede ayudarse con *isin()*.
3. Utilice *.set_index()* para crear un nuevo dataframe llamado *tempCity* con los índices a partir de las ciudades. Luego use la función .loc[] para crear un subset de datos que contenga las ciudades en la lista *ciudades*

¿Cual de las formas de crear subsets les parece más facil y/o intuitiva?

In [18]:
```
## TODO 3.2
```

## 3.3 Trabajando con índices multinivel

A partir del dataframe *temperaturas* setee los índices *country* y *city* y asigne el dataframe a *tempCC*. Luego cree una lista de tuplas de tal manera de tener los pares país/ciudad de Brazil/Rio de Janeiro, Canada/Toronto y China/Wuhan. Finalmente, imprima un subset de datos de *tempCC* donde solo se muestren los datos correspondientes a los pares mencionados anteriormente. Utilice .loc[].

In [19]:
```
## TODO 3.3

tempCC = ___
```

## 3.4 Slicing

Ordene el dataframe *tempCC* usando *.sort_index()* luego,

1. Imprima un slice contemplando los países de Brasil hasta Dominican Republic.
2. Imprimia un slice contemplando las ciudades de Belo Horizonte hasta Bogotá.
3. Cree un dataframe llamado *tempsDates* el cual reciba el dataframe *temperaturas* pero con la columna *date* seteada como índice. Ordene *tempsDates* por índice. Luego imprima un slice que contemple las fechas 2010-07 y 2011-01.

In [20]:
```

```

```
## TODO 3.4
```

### 3.5 Tabla dinámica por ciudad y año

Ahora creará una tabla dinámica para obtener información de la variación de temperatura por cada ciudad dentro del set de datos.

1. En primer lugar, agregue una columna llamada *year* al set *temperaturas* que contenga los años a partir de la columna *date* del dataframe. Es posible obtener los días, los mesos o los años de un dato del tipo *fecha* haciendo

```
dataframe["columa"].dt.component
```

Así por ejemplo, tomando el dataframe *temperaturas* podríamos agregar una columna *year* haciendo,

```
temperaturas["year"] = temperaturas["date"].dt.year
```

**IMPORTANTE**: Si al intentar hacer lo anterior usted recibe un error del tipo

```
AttributeError: Can only use .dt accessor with datetimelike values
```

Debe ejecutar la siguiente linea de código para formatear las fechas.

```
t['date'] = pd.to_datetime(t.date, format='%Y-%m-%d')
```

1. Cree una tabla pivote que tome los valores de *avg_temp_c*, donde los índices estén formados por *country* y *city* (en ese orden) y las columnas contengan los años de la columna *year* agregada en el paso anterior. Guarde la tabla en *tablaTemps*. Imprima *tablaTemps*.

In [21]:
```
## TODO 3.5
tablaTemps = ___
```

### 3.6 Variación de temperatura para la ciudad de Wuhan, China.

A continuación veremos la variación de temperatura promedio desde el año 2000 al 2013 para la ciudad de Wuhan. Una forma de hacer esto es siguiendo estos pasos.

- Utiliando el dataframe *temperaturas* con la columna *year* agregada, primeramente filtre con [] los datos correspondientes a *China*. Luego cree una *pivote_table()* donde sus valores estén formados por los datos de las columnas *avg_temp_c*, los índices correspondan a la columna *city* y las columnas correspondan a *year*.
- Imprima la tabla. Si todo esta OK, Lla misma debe contener las temperaturas promedios de las ciudades de China que figuren en el dataset desde el año 2000 al año 2013. Guarde esta tabla en *tempsChinaCities*.
- Utilizando *tempsChinaCities.loc[]* obtenga las variaciones de temperatura correspondientes a la ciudad de *Wuhan*. Guarde estos datos en *WuhanTemps*.
- Finalmente, descomente las lineas para graficar.

In [22]:
```
## TODO 3.6
tempsChinaCities = ___
WuhanTemps = ___
```

In [23]:
```
```

```
## # Descomentar para graficar
# import matplotlib.pyplot as plt

# plt.plot(WuhanTemps, label = "Wuhan")
# plt.xlabel("Año")
# plt.ylabel("Temperatura [°C]")
# plt.title("Temperaturas promedios desde 2000 a 2013")
# plt.legend()
# plt.show()
```

# 4. Graficación básica

En este ejercicio se trabajará con un set de datos que contiene las ventas semanales de *paltas* en tiendas de EEUU. Estas ventas están separadas por año, por tipo (convencional u orgnánico) y por tamaño (small, large, extra_large).

Cargue los datos de *paltas.csv* que se encuentra dentro de *datasets/*. Imvestigue el dataframe con los métodos vistos.

## 4.1 Paltas populares

- Obtenga el número total de paltas vendidas por cada tamaño (*size*). Puede usar *.groupby()* para agrupar por tamaño y luego computar las ventas semanales totales. Guarde estos datos en *ventasXSize*.
- Utilice *ventasXSize* para realizar un gráfico de barras.

In [24]:
```
## TODO 4.1
ventasXSize = ___

## gráfica
# ventasXSize.plot(kind = "bar")
# plt.show()
```

## 4.2 Ventas a lo largo del tiempo

- Compute el número de paltas vendidas por fecha. Guarde los datos en *ventasXFecha*.
- Realice un gráfico de *línea* a partir de *ventasXFecha*.

In [25]:
```
## TODO 4.2

ventasXFecha = ___

##gráfica
pass
```

## 4.3 Ventas vs Precio

Realice un gráfico del tipo *scatter* en donde el eje x tenga los datos de la columna *nb_sold* y el eje y tenga los datos de la columna *avg_price*. Agregue un título al gráfico.

## 4.3 Convencional vs Orgánico

Realice un histograma donde contabilicen las ventas semanales de paltas orgánicas y otro en donde contabilicen las ventas semanales de paltas convenionales. Coloque ambos histogramas en

el mismo gráfico.

# 5. Analizando jugadores FIFA 2021 (obligatorio)

Para este ejercicio se utilizará el set de datos del juego FIFA 2021. El mismo contiene información acerca de los jugadores de fútbol de diferentes ligas y selecciones a nivel mundial.

El archivo con el cual trabajará se llama *fifa2021.csv*.

Antes de empezar, realizaremos una limpieza de los datos.

In [5]:
```
fifa21 = pd.read_csv("datasets/fifa2021.csv")
fifa21.head()
```

Out[5]:

| | sofifa_id | player_url | short_name | long_name | age | dob | heig |
|---|---|---|---|---|---|---|---|
| **0** | 158023 | https://sofifa.com/player/158023/lionel-messi/... | L. Messi | Lionel Andrés Messi Cuccittini | 33 | 6/24/1987 | |
| **1** | 20801 | https://sofifa.com/player/20801/c-ronaldo-dos-... | Cristiano Ronaldo | Cristiano Ronaldo dos Santos Aveiro | 35 | 2/5/1985 | |
| **2** | 200389 | https://sofifa.com/player/200389/jan-oblak/210002 | J. Oblak | Jan Oblak | 27 | 1/7/1993 | |
| **3** | 188545 | https://sofifa.com/player/188545/robert-lewand... | R. Lewandowski | Robert Lewandowski | 31 | 8/21/1988 | |
| **4** | 190871 | https://sofifa.com/player/190871/neymar-da-sil... | Neymar Jr | Neymar da Silva Santos Júnior | 28 | 2/5/1992 | |

5 rows × 82 columns

In [6]:
```
## Cantidad de datos
print(fifa21.shape)
```

```
(18944, 82)
```

In [28]:
```
## Eliminamos algunas columnas con drop()
fifa21 = fifa21.drop(["sofifa_id","player_url","long_name","dob","team_jersey_number",
fifa21.shape
```

Out[28]:
```
(18944, 75)
```

In [7]:
```
## Chequeamos si tenemos columnas con valores NaN (al menos uno)
print(fifa21.columns[fifa21.isna().any()])
```

```
Index(['club_name', 'league_name', 'league_rank', 'release_clause_eur',
       'player_tags', 'team_position', 'team_jersey_number', 'loaned_from',
       'joined', 'contract_valid_until', 'nation_position',
       'nation_jersey_number', 'pace', 'shooting', 'passing', 'dribbling',
       'defending', 'physic', 'gk_diving', 'gk_handling', 'gk_kicking',
       'gk_reflexes', 'gk_speed', 'gk_positioning', 'player_traits',
```

```
       'defending_marking'],
      dtype='object')
```

Vemos que varias columnas tienen al menos un valor NaN. Por simplicidad vamos a quitar dichas columnas.

In [8]:
```
fifa21 = fifa21.drop(['league_rank', 'release_clause_eur',
       'player_tags', 'team_position', 'loaned_from', 'joined',
       'contract_valid_until', 'nation_position', 'nation_jersey_number',
       'pace', 'shooting', 'passing', 'dribbling', 'defending', 'physic',
       'gk_diving', 'gk_handling', 'gk_kicking', 'gk_reflexes', 'gk_speed',
       'gk_positioning', 'player_traits', 'defending_marking','real_face'], axis = 1)
```

In [9]:
```
fifa21.shape
fifa21
```

Out[9]:

| | sofifa_id | player_url | short_name | long_name | age | d |
|---|---|---|---|---|---|---|
| **0** | 158023 | https://sofifa.com/player/158023/lionel-messi/... | L. Messi | Lionel Andrés Messi Cuccittini | 33 | 6/24/1 |
| **1** | 20801 | https://sofifa.com/player/20801/c-ronaldo-dos-... | Cristiano Ronaldo | Cristiano Ronaldo dos Santos Aveiro | 35 | 2/5/1 |
| **2** | 200389 | https://sofifa.com/player/200389/jan-oblak/210002 | J. Oblak | Jan Oblak | 27 | 1/7/1 |
| **3** | 188545 | https://sofifa.com/player/188545/robert-lewand... | R. Lewandowski | Robert Lewandowski | 31 | 8/21/1 |
| **4** | 190871 | https://sofifa.com/player/190871/neymar-da-sil... | Neymar Jr | Neymar da Silva Santos Júnior | 28 | 2/5/1 |
| **...** | ... | ... | ... | ... | ... | |
| **18939** | 256679 | https://sofifa.com/player/256679/kevin-angulo/... | K. Angulo | Kevin Angulo | 24 | 4/13/1 |
| **18940** | 257710 | https://sofifa.com/player/257710/mengxuan-zhan... | Zhang Mengxuan | Mengxuan Zhang | 21 | 4/26/1 |
| **18941** | 250989 | https://sofifa.com/player/250989/zhenghao-wang... | Wang Zhenghao | 王政豪 | 20 | 6/28/2 |
| **18942** | 257697 | https://sofifa.com/player/257697/zitong-chen/2... | Chen Zitong | Zitong Chen | 23 | 2/20/1 |
| **18943** | 257936 | https://sofifa.com/player/257936/yue-song/210002 | Song Yue | Yue Song | 28 | 11/20/1 |

18944 rows × 58 columns

Ahora vamos a chequear si tenemos algún valor Nan dentro del DataFrame.

In [10]:
```
fifa21.isna().any().any()
```

```
True
```

Out[10]:

Vemos que *fifa21.isna().any().any()* nos arroja *True*, indicando que tenemos valores NaN.

Cuando hacemos

```
fifa21.isna().any().any()
```

Estamos evaluando si algún valor dentro del objeto *Series* de pandas devuelto al hacer *fifa21.isna().any()* tiene algún valor True. Como al menos un valor dentro del objeto *Series* posee un valor True, *fifa21.isna().any().any()* nos arroja True.

In [11]:
```
## Nos quedamos con aquellos datos que tienen nombres de club que no sean NA
fifa21 = fifa21[fifa21["club_name"].notna()]

## Nos quedamos con aquellos datos que tienen nombres de liga que no sean NA
fifa21 = fifa21[fifa21["league_name"].notna()]
```

In [12]:
```
## El resto lo llenamos de cero
fifa21 = fifa21.fillna(0)
```

¿Estamos seguros que hemos sacado todos los valores NaN?

In [35]:
```
fifa21.isnull().values.any()
```

Out[35]:
```
False
```

Por lo que podemos ver, ¡sí!

### Chequeando valores *Null*

Podríamos tener valores *Null* dentro del dataframe. Pandas ofrece un método para chequear la presencia de este tipo de valores, el mismo es *.isnull()*.

Veamos si tenemos datos null.

In [13]:
```
fifa21.isnull().values.any()
```

Out[13]:
```
False
```

Por lo que podemos ver, no tenemos valores *null* dentro de nuestro set de datos.

Ahora podemos empezar a trabajar con el dataframe.

Empecemos...

In [37]:
```
print(f"Cantidad de filas/observaciones {fifa21.shape[0]}")
print(f"Cantidad de variables/columnas {fifa21.shape[1]}")
```

```
Cantidad de filas/observaciones 18719
Cantidad de variables/columnas 51
```

### ¿Cuantos y qué paises tenemos en el set de datos?

Podríamos usar *pandas.unique(values)* para determinar los jugadores por países de la siguiente manera.

In [14]:
```
print(f"Cantidad de países en el set {len(fifa21['nationality'].unique())}")
fifa21['nationality'].unique()
```

Out[14]:
```
Cantidad de países en el set 161
array(['Argentina', 'Portugal', 'Slovenia', 'Poland', 'Brazil', 'Belgium',
       'France', 'Germany', 'Netherlands', 'Senegal', 'Egypt', 'Spain',
       'England', 'Scotland', 'Italy', 'Uruguay', 'Croatia', 'Gabon',
       'Costa Rica', 'Korea Republic', 'Switzerland', 'Serbia',
       'Slovakia', 'Morocco', 'Bosnia Herzegovina', 'Hungary', 'Denmark',
       'Algeria', 'Norway', 'Cameroon', 'Nigeria', 'Ghana', 'Mexico',
       'Austria', 'Albania', 'Greece', 'Sweden', 'Wales', 'Chile',
       'Finland', 'Ivory Coast', 'Colombia', 'Togo', 'Czech Republic',
       'Russia', 'Canada', 'United States', 'Guinea', 'Montenegro',
       'Venezuela', 'Ukraine', 'Republic of Ireland', 'Israel', 'Jamaica',
       'Turkey', 'Australia', 'Northern Ireland', 'China PR', 'Armenia',
       'DR Congo', 'Ecuador', 'Kosovo', 'North Macedonia',
       'Central African Republic', 'Iceland', 'Peru', 'Mali',
       'Burkina Faso', 'Paraguay', 'Romania', 'New Zealand', 'Japan',
       'Cape Verde', 'Tunisia', 'Angola', 'Dominican Republic', 'Syria',
       'Iran', 'Zambia', 'Panama', 'Georgia', 'Equatorial Guinea',
       'Tanzania', 'Kenya', 'Honduras', 'Congo', 'Guinea Bissau',
       'Zimbabwe', 'South Africa', 'Iraq', 'Madagascar', 'Moldova',
       'Curacao', 'Cuba', 'Mozambique', 'Gambia', 'Cyprus', 'Estonia',
       'Benin', 'Lithuania', 'Saudi Arabia', 'Liberia', 'Bulgaria',
       'Libya', 'Philippines', 'Sierra Leone', 'Uzbekistan',
       'Saint Kitts and Nevis', 'United Arab Emirates', 'Comoros', 'Chad',
       'Namibia', 'Thailand', 'Bermuda', 'Luxembourg',
       'Trinidad & Tobago', 'Antigua & Barbuda', 'Burundi', 'Kazakhstan',
       'New Caledonia', 'Puerto Rico', 'Bolivia', 'Eritrea', 'Latvia',
       'Montserrat', 'São Tomé & Príncipe', 'Malawi', 'Mauritania',
       'El Salvador', 'Haiti', 'Uganda', 'Chinese Taipei', 'Aruba',
       'Faroe Islands', 'Guyana', 'Azerbaijan', 'Afghanistan', 'Sudan',
       'Guam', 'Belarus', 'Rwanda', 'Grenada', 'Palestine', 'Lebanon',
       'Papua New Guinea', 'Jordan', 'Saint Lucia', 'Liechtenstein',
       'Ethiopia', 'Belize', 'Niger', 'Malta', 'Andorra', 'Barbados',
       'South Sudan', 'Macau', 'Korea DPR', 'Hong Kong', 'Malaysia',
       'Indonesia', 'Nicaragua'], dtype=object)
```

## Ejercicios para punto 5

A partir de acá usted debe resolver lo que se pide.

5.1 Jugadores de Uruguay y Argentina

¿Qué y cuántos jugadores de Uruguay tenemos en el set de datos? ¿Cuales y cuantos de Argen tina?

Podemos responder a la pregunta aplicando lo que hemos visto. A continuación se muestra un ejemplo para obtener los jugadores de Uruguay. Repita para Argentina (y otros países si quisiera).

In [39]:
```
jugadoresUruguay = fifa21[fifa21["nationality"] == "Uruguay"].drop_duplicates(subset =
print(f"Cantidad de jugadores de Uruguay en el set de datos {jugadoresUruguay.shape[0]
print()
print("Lista")
print(jugadoresUruguay)
```

```
Cantidad de jugadores de Uruguay en el set de datos 311

Lista
10163         A. Ale
18260      A. Alfaro
```

```
7255      A. Argachá
17428     A. Barboza
9672     A. Canobbio
            ...
18088    T. Galletto
6424       T. Vecino
6194      W. Camacho
2818      W. Gargano
7581     Y. Calleros
Name: short_name, Length: 311, dtype: object
```

In [105]:

```
jugadoresArg = fifa21[fifa21["nationality"] == "Argentina"].drop_duplicates(subset = '
print(f"Cantidad de jugadores de Argentina en el set de datos {jugadoresArg.shape[0]}'
print()
print("Lista")
print(jugadoresArg)
```

```
Cantidad de jugadores de Argentina en el set de datos 898

Lista
2236       A. Aguerre
16751      A. Aguirre
12067         A. Aleo
7362      A. Almendra
9696       A. Antilef
             ...
1772     W. Caballero
2997       W. Montoya
15455        W. Ortíz
3392          Y. Asad
15529       Y. Juárez
Name: short_name, Length: 898, dtype: object
```

In [104]:

```
## TODO 5.1
jugadoresUruguayPorLiga= fifa21[fifa21["nationality"] == "Uruguay"].drop_duplicates(su
jugadoresUruguayPorLiga[["short_name","league_name"]].sort_values("league_name")#Jugad
```

Out[104]:

| | short_name | league_name |
|---|---|---|
| **7192** | M. Cauteruccio | Argentina Primera División |
| **9898** | F. Martínez | Argentina Primera División |
| **3038** | D. Zabala | Argentina Primera División |
| **9880** | M. Merentiel | Argentina Primera División |
| **3424** | S. Silva | Argentina Primera División |
| **...** | ... | ... |
| **9775** | E. Martínez | Uruguayan Primera División |
| **9778** | S. Pérez | Uruguayan Primera División |
| **9177** | I. Pallas | Uruguayan Primera División |
| **18689** | C. Silva | Uruguayan Primera División |
| **16068** | I. González | Venezuelan Primera División |

321 rows × 2 columns

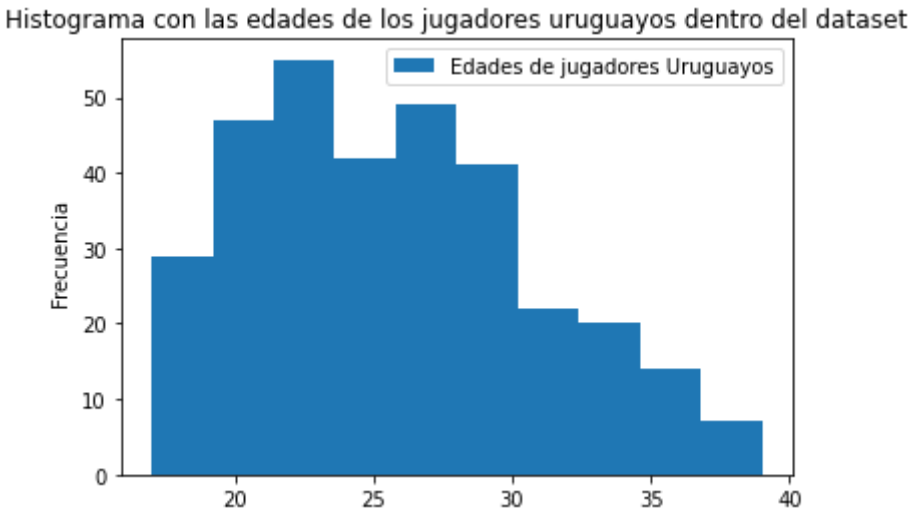
5.2 Un poco de análisis sobre jugadores uruguayos

1. ¿Cuales son las ligas donde juegan los jugadores de nacionalidad uruguaya según el set de datos?
2. ¿Cuál es la liga con mayor cantidad de jugadores uruguayos? ¿Qué valor tiene? ¿Cuál es la segunda liga con mayor presencia de jugadores uruguayos?
3. ¿Cuantos jugadores uruguayos juegan en la liga *Spain Primera Division*? ¿Quienes son?
4. Tomando solo la liga *Spain Primera Division*, ¿cuál es la proporción de jugadores uruguayos? Es decir, cuantos jugadores de nacionalidad uruguaya juegan en la *Spain Primera Division* respecto de todos los que juegan en la *Spain Primera Division*. Reporte un valor porcentual %
5. ¿Cuales son los porcentajes de jugadores uruguayos de *pie derecho* y *pie izquierdo*?
6. Realice un histograma con las edades de los jugadores uruguayos dentro del dataset.
7. ¿Cuál es el jugador más caro? ¿Cuál el más barato?

In [154]:

```
import matplotlib.pyplot as plt
#5.2 - 1
jugadoresUruguayPorLiga= fifa21[fifa21["nationality"] == "Uruguay"].drop_duplicates(su
jugadoresUruguayPorLiga[["short_name","league_name"]].sort_values("league_name") #Cont

#5.2 - 2
cantidadJugadoresPorLiga=jugadoresUruguayPorLiga[["short_name","league_name"]].sort_va
juagdoreUYporLiga=cantidadJugadoresPorLiga["league_name"].value_counts().sort_values(a
juagdoreUYporLiga[0:1]#Primer liga con mayor presencia de uruguayos
juagdoreUYporLiga[1:2]#Segunda liga con mayor presencia de uruguayos

#5.2 - 3
uruguayosLigaEsp = juagdoreUYporLiga["Spain Primera Division"] #Jugadores uruguayos er
print(f"La cantidad de jugadores uruguayos en la Spain Primera Division son:{uruguayos

#5.2 - 4
totalLigaEspañola=fifa21[fifa21["league_name"] == "Spain Primera Division"].shape

proporcionUYLigaEsp=round((uruguayosLigaEsp/totalLigaEspañola[0])*100,1) #Porcentaje d
print(f"La proporción de jugadores uruguayos en la Primera divión española es: {propor

#5.2 - 5
jugadoresUYDiestro= fifa21[(fifa21["nationality"] == "Uruguay") & (fifa21["preferred_f
jugadoresUYZiniestro= fifa21[(fifa21["nationality"] == "Uruguay") & (fifa21["preferrec

#5.2 - 6
fig, ax = plt.subplots()
jugadoresUYedad= fifa21[fifa21["nationality"] == "Uruguay"]["age"] #Filtramos los jugc
ax.hist(jugadoresUYedad, label = "Edades de jugadores Uruguayos") #generamos el histog
ax.set_ylabel(" Frecuencia") #Leyenda eje Y
ax.set_title("Histograma con las edades de los jugadores uruguayos dentro del dataset'
plt.legend()
plt.show()

#5.2.7
jugadorUYMasCaro= fifa21[fifa21["nationality"] == "Uruguay"]["value_eur"].max() #Cream
fifa21[(fifa21["nationality"] == "Uruguay") & (fifa21["value_eur"] == jugadorUYMasCarc

jugadorUYMasBarato= fifa21[fifa21["nationality"] == "Uruguay"]["value_eur"].min() #Cre
fifa21[(fifa21["nationality"] == "Uruguay") & (fifa21["value_eur"] == jugadorUYMasBara
```

```
La cantidad de jugadores uruguayos en la Spain Primera Division son:15
La proporción de jugadores uruguayos en la Primera divión española es: 2.3 %
```




Out[154]:

| | sofifa_id | player_url | short_name | long_name | age | dob | he |
|---|---|---|---|---|---|---|---|
| **17473** | 258113 | https://sofifa.com/player/258113/alvaro-garcia... | A. García | Álvaro Marcelo García Zaroba | 36 | 1/13/1984 | |

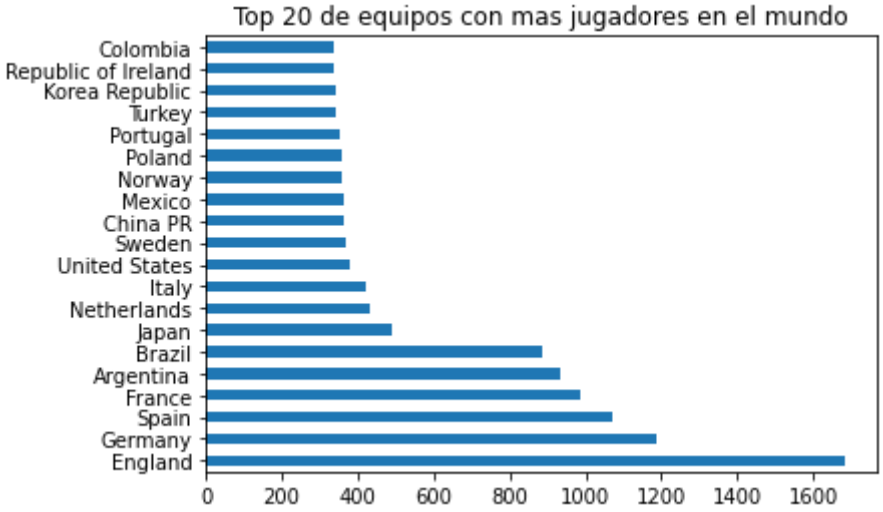
1 rows × 58 columns

5.3 Top 20 países con mayores jugadores

Realice un gráfico de barras horizontal (*kind = 'barh'*) donde se muestre la cantidad de jugadores de los 20 países con mayores cantidades de jugadores.

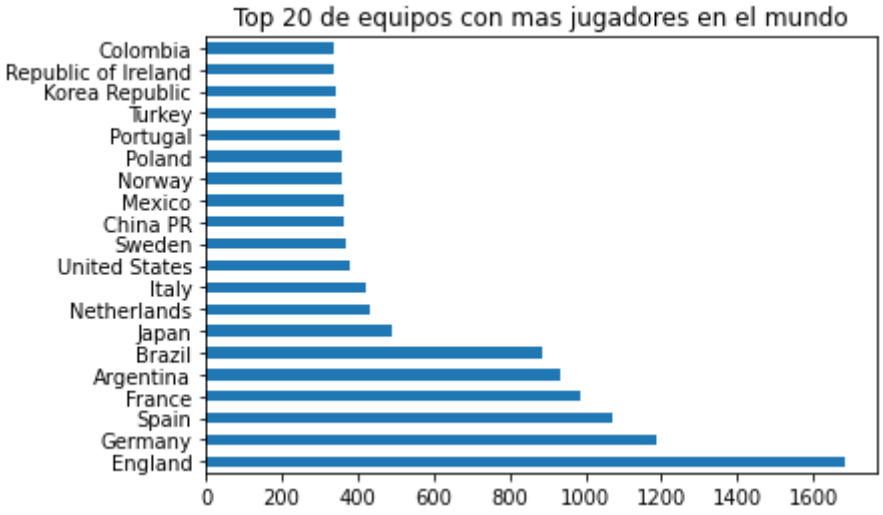
In [155]:
```
## TODO 5.3 Forma 1
top20 = fifa21["nationality"].value_counts() # realizo un conteo de cada jugador por p
muestra = top20[:20].copy() #Me quedo con los primeros 20 a partir de usar slicing
muestra.plot(kind='barh',  title="Top 20 de equipos con mas jugadores en el mundo") #
```

Out[155]:
```
<AxesSubplot:title={'center':'Top 20 de equipos con mas jugadores en el mundo'}>
```




In [157]:
```
# 5.3 Forma 2
top20 = fifa21["nationality"].value_counts()
top20.head(20).plot(kind='barh',  title="Top 20 de equipos con mas jugadores en el mur
```

Out[157]:
```
<AxesSubplot:title={'center':'Top 20 de equipos con mas jugadores en el mundo'}>
```




5.4 Puntaje total (***overall***) vs potencial (***potential***)

El set de datos cuenta con las columnas llamadas *overall* y *potential*. El *overall* es un puntaje que va de 0 a 100 y representa la media ponderada de todas las habilidades (ej. *skill_dribbling*, *movement_sprint_speed*, etc) para un cierto jugador. Por otro lado, el *potential* o *potencial* es una estimación de si el jugador puede dar más en un futuro.

Por ejemplo,

In [43]:
```
filtromessi = fifa21["short_name"] == "L. Messi"
filtrombappe = fifa21["short_name"] == "K. Mbappé"
fifa21[filtromessi | filtrombappe][["short_name","overall","potential"]].set_index("sh
```

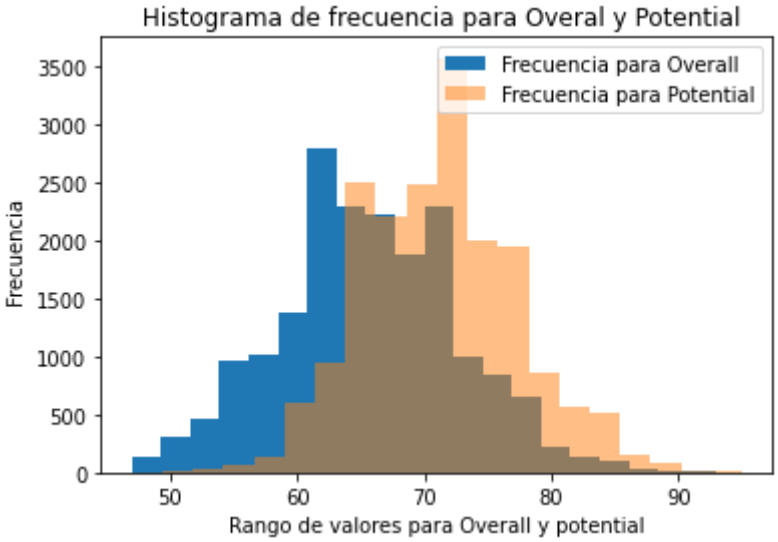
Out[43]:

| short_name | overall | potential |
|---|---|---|
| **L. Messi** | 93 | 93 |
| **K. Mbappé** | 90 | 95 |

Podemos ver del dataframe anterior que Messi esta dando todo lo que puede dar, mientras que Mbappe aún podría dar un poco más según las estimaciones del FIFA.

Se le pide que grafique un histograma considerando el dataframe completo donde se pueda ver la frecuencia para *overall* y para *potential*.

In [56]:
```
#5.4
fig, ax = plt.subplots()
ax.hist(fifa21["overall"], label = "Frecuencia para Overall", bins = 20) #Histograma d
ax.hist(fifa21["potential"], label = "Frecuencia para Potential", alpha = 0.5, bins =
ax.set_xlabel("Rango de valores para Overall y potential")
ax.set_ylabel(" Frecuencia")
ax.set_title(" Histograma de frecuencia para Overal y Potential")
plt.legend()
plt.show()
```




5.5 Analizando datos por posiciones

Si se observa la columna *player_positions* del set de datos podemos ver en qué posiciones juega cierto jugador.

En base al juego, las posiciones pueden resumirse en:

- 'ST','CF','RW','LW' = Atacante
- 'LM','RM','CM','CDM','CAM', "RWB" = Mediocampista
- 'LB','RB','CB', 'LWB' = Defensor
- 'GK' = Arquero

Ejemplo

In [45]:
```
fifa21[["player_positions","short_name"]].set_index("short_name").head()
```

Out[45]:

| short_name | player_positions |
|---|---|
| **L. Messi** | RW, ST, CF |
| **Cristiano Ronaldo** | ST, LW |
| **J. Oblak** | GK |
| **R. Lewandowski** | ST |
| **Neymar Jr** | LW, CAM |

Resuelva

1. Recorra la columna *player_positions* del dataframe y genere una nueva columna llamada *onePosition* donde cada posición tenga los valores Atacante, Mediocampista, Defensor o Arquero, según corresponda. Puede utilizar un *for* para recorrer los datos de la columna. Hay jugadores que juegan en más de una posición, por ejemplo Cristiano Ronaldo, en esos casos reemplace con la primer ocurrencia, en el ejemplo de CR7 su posición sería reemplazada por *Atacante* ya que *ST* es la primer ocurrencia. Una vez agregada la columna *onePosition*, elimine la columna *player_positions*

2. Genere un gráfico de barras donde se muestre la cantidad de jugadores por cada posición del total del dataframe.
3. Repita el punto 3 pero solamente para jugadores de nacionalidad uruguaya.
4. ¿Cual es el promedio de costos por posición de los jugadores de nacionalidad uruguaya?

In [46]:
```
## TODO 5.5
```

## 5.6 TOP 10

1. Genere un nuevo subset con los datos filtrado por la columna *position* para *Atacante*. Sólo tome os primeros 10 valores. El nuevo subset debe tener las columnas *[short_name, nationality, club_name, height_cm, position, value_eur, overall, potential]*. Ordenarlo por overall y potential (en ese orden) de manera ascendente (ambos). Realizar un set_index() por *short_name*.
2. Repita el punto 1 para formar subsets para las posiciones *Mediocampista*, *Defensor* y *Arquero.

In [47]:
```
## TODO 5.6
```