EJERCICIO 1 MINERIA DE DATOS

- Manuel de Jesús Vázquez Bocanegra 1823593
- Christian Servando Garza Gonzalez 1505813
- Jorge Noe Zuñiga Gomez 1565813
- Asiel Romero Galvan 1937895
- Manuel Joseph Romero Pascacio 1811177

Problema 1

EJERCICIO REGRESIÓN LINEAL.

Tomando los datos de la siguiente tabla sobre los pesos y alturas de una población de 30 personas, crea una gráfica en donde el valor x represente la altura y el valor y represente el peso. Después traza una línea que se apegue lo mas posible a los datos que graficaste.

Peso	Altura	Peso	Altura	Peso	Altura
68.78	162	67.19	183	67.89	162
74.11	212	65.80	163	68.14	192
71.73	220	64.30	163	69.08	184
69.88	206	67.97	172	72.80	206
67.25	152	72.18	194	67.42	175
68.78	183	65.27	168	68.49	154
68.34	167	66.09	161	68.61	187
67.01	175	67.51	164	74.03	212
63.45	156	70.10	188	71.52	195
71.19	186	68.25	187	69.18	205

Libreria de Regresion lineal

In [ ]:
```
from sklearn.linear_model import LinearRegression
```

In [46]:
```
import matplotlib.pyplot as plt
```

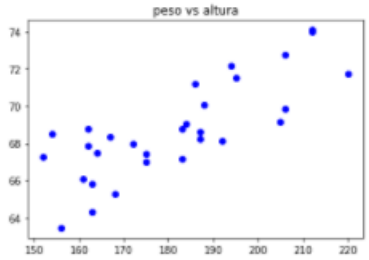
In [16]:
```
import numpy as np
```

In [45]:
```
y=[68.78,74.11,71.73,69.88,67.25,68.78,68.34,67.01,63.45,71.19,67.19,65.8,64.3,67.97,72.18,65.27,66.09,67.51,70.1,68.25,67.8
x=[162,212,220,206,152,183,167,175,156,186,183,163,163,172,194,168,161,164,188,187,162,192,184,206,175,154,187,212,195,205]
```

Pasamos de listas a arreglos

In [21]:
```
Xaltura= np.array(x)
Ypeso = np.array(y)
```

In [47]:
```
plt.plot(Xaltura,Ypeso,"o",label="Datos" ,color='blue')
plt.title('peso vs altura')
plt.show()
```



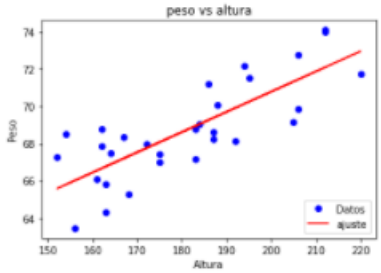
In [33]:
```
regresion_lineal = LinearRegression()
regresion_lineal.fit(Xaltura.reshape(-1,1),Ypeso)
#https://www.iartificial.net/regresion-lineal-con-ejemplos-en-python/
```

Out[33]:
```
LinearRegression()
```

In [49]:
```
print(f"los coeficientes son: m= {regresion_lineal.coef_} y b= {regresion_lineal.intercept_}")
print(f"Y la ecuacion de la regresión sería: y ={regresion_lineal.coef_} X + {regresion_lineal.intercept_}")
```

```
los coeficientes son: m= [0.10861078] y b= 49.071633695475924
Y la ecuacion de la regresión sería: y =[0.10861078] X + 49.071633695475924
```

In [50]:
```
plt.plot(Xaltura,Ypeso,"o",label="Datos" ,color='blue')
plt.title('peso vs altura')
plt.plot(Xaltura,regresion_lineal.coef_*Xaltura+regresion_lineal.intercept_,label="ajuste",color="Red")
plt.xlabel("Altura")
plt.ylabel("Peso")
plt.legend(loc= 4)
plt.show()
```



Problema 2

Observa la tabla que se describe a continuación. Utilizando el algoritmo a priori, y la técnica de asociación, realiza la tabla de relaciones y resuelve cuál es el nivel **K** de soporte más alto al que podemos llegar con estos datos teniendo un umbral de 0.5.

ID	Transacciones
1	A B C E
2	B E
3	C D E
4	A C D
5	A C E

In [15]:
```
def Load_data():

    transacciones = (["A", "B", "C", "E"],["B", "E"],["C", "D", "E"],["A", "C", "D"],["A", "C", "E"],)

    return transacciones
```

In [16]:
```
def Conjunto1 (data):
    C1 = []
    for transaction in data:
        for item in transaction:
            if not [item] in C1:
                C1.append([item])
    C1.sort()

    return [ set (x) for x in C1]
```

In [17]:
```
def ConjuntoK(Lk, k):
    list_cand = []
    len_Lk = len(Lk)

    for i in range(len_Lk):
        for j in range(i+1, len_Lk):
            L1 = list(Lk[i])[:k-2]
            L2 = list(Lk[j])[:k-2]
            L1.sort()
            L2.sort()
            if L1==L2:
                list_cand.append(Lk[i] | Lk[j])
    return list_cand
```

In [18]:
```
def scanD(Data, Ck, min_support):

    count = {}
    for transaction in data:
        tr=set(transaction)
        for candidate in Ck:
            if candidate.issubset(tr):
                can=frozenset(candidate)
                if can not in count:
                    count[can]=1
                else:
                    count[can]+=1
                    num_items= float(len(D))

    list_cand=[]
    support_data={}


    for key in count:
        support=count[key]/num_items

        if support>=min_support:
            list_cand.insert(0,key)
        support_data[key]=support

    return list_cand, support_data
```

Umbral de 0.5

In [19]:
```
min_support= 0.5
```

In [20]:
```
data= Load_data()
data
```

Out[20]:
```
(['A', 'B', 'C', 'E'],
 ['B', 'E'],
 ['C', 'D', 'E'],
 ['A', 'C', 'D'],
 ['A', 'C', 'E'])
```

In [21]:
```
C1 = Conjunto1(data)
C1
```

Out[21]:
```
[{'A'}, {'B'}, {'C'}, {'D'}, {'E'}]
```

In [22]:
```
D=list(map(set,data))
D
```

Out[22]:
```
[{'A', 'B', 'C', 'E'},
 {'B', 'E'},
 {'C', 'D', 'E'},
 {'A', 'C', 'D'},
 {'A', 'C', 'E'}]
```

In [23]:
```
L1, support_data1 = scanD(D, C1, min_support)
L1
```

Out[23]:
```
[frozenset({'E'}), frozenset({'C'}), frozenset({'A'})]
```

In [35]:
```
support_data1
```

Out[35]:
```
{frozenset({'A'}): 0.6,
 frozenset({'B'}): 0.4,
 frozenset({'C'}): 0.8,
 frozenset({'E'}): 0.8,
 frozenset({'D'}): 0.4}
```

En k=2 entonces Ck=C2

In [36]:
```
C2= ConjuntoK(L1, k=2)
C2
```

Out[36]:
```
[frozenset({'C', 'E'}), frozenset({'A', 'E'}), frozenset({'A', 'C'})]
```

In [37]:
```
L2, support_data2 = scanD(D, C2, min_support)
L2
```

Out[37]:
```
[frozenset({'A', 'C'}), frozenset({'C', 'E'})]
```

In [38]:
```
support_data2
```

Out[38]:
```
{frozenset({'C', 'E'}): 0.6,
 frozenset({'A', 'E'}): 0.4,
 frozenset({'A', 'C'}): 0.6}
```

Ahora k=3

In [39]:
```
C3= ConjuntoK(L2, k=3)
C3
```

Out[39]:
```
[frozenset({'A', 'C', 'E'})]
```

In [40]:
```
L3, support_data3 = scanD(D, C3, min_support)
L3
```

Out[40]:
```
[]
```

In [41]:
```
support_data3
```

Out[41]:
```
{frozenset({'A', 'C', 'E'}): 0.4}
```

Los conjuntos serian:

con k=1

In [42]:
```
support_data1
```

Out[42]:
```
{frozenset({'A'}): 0.6,
 frozenset({'B'}): 0.4,
 frozenset({'C'}): 0.8,
 frozenset({'E'}): 0.8,
 frozenset({'D'}): 0.4}
```

Con k=2

In [ ]:
```
support_data2
```

Con k=3

Con k=3

In [22]:
```
support_data3
```

Out[22]:
```
{}
```

Conclusion

Los niveles de k mas altos y que cumplen con el umbral de 0.5 son los siguientes:

- En k=1

  "A" = 0.6
  "C" = 0.8
  "E" = 0.8

- En k=2

  "C,E"= 0.6
  "A,C"= 0.6