UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
Facultad de Ciencias Físico – Matemáticas
Ejercicios 1
Mineria de Datos

Equipo: Visualización de datos

Nombre	*Matricula*
Flor Karina Juárez Rodríguez	1802920
Gloria Nohemí Martínez Jiménez	1805800
Pilar Abigail Mendoza Alvarez	1815973
Margarita Ordaz Ruiz	1802473
Tania Sarahi Rossel Castillo	1810461

In [1]:
```
#Datos x
Altura = [ 162, 212, 220, 206, 152, 183, 167, 175, 156, 186, 183, 163, 163, 17
2, 194, 168, 161, 164,
          188, 187, 162, 192, 184, 206, 175, 154, 187, 212, 195, 205 ]

#Datos y
Peso = [ 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, 67.89, 68.14, 69.08, 72.8, 6
7.42, 68.49, 68.61, 74.03, 71.52, 69.18 ]
```

In [2]:
```
import pandas as pd

tabla = {'Altura': Altura, 'Peso': Peso}

Mostrart = pd.DataFrame(tabla)
```

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.

In [3]:
```
Mostrart
```

Out[3]:

	Altura	Peso
0	162	68.78
1	212	74.11
2	220	71.73
3	206	69.88
4	152	67.25
5	183	68.78
6	167	68.34
7	175	67.01
8	156	63.45
9	186	71.19
10	183	67.19
11	163	65.80
12	163	64.30
13	172	67.97
14	194	72.18
15	168	65.27
16	161	66.09
17	164	67.51
18	188	70.10
19	187	68.25
20	162	67.89
21	192	68.14
22	184	69.08
23	206	72.80
24	175	67.42
25	154	68.49
26	187	68.61
27	212	74.03
28	195	71.52
29	205	69.18

In [4]:
```
import matplotlib.pyplot as plt

plt.scatter(x=Altura , y=Peso,  marker='*', c='pink', s=60)
plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

In [5]:
```
import numpy as np

array_x = np.array(Altura)
array_y = np.array(Peso)
```

In [6]:
```
n = len(array_x)
sum_x = sum(array_x)
sum_y = sum(array_y)
sum_xy = sum(array_x*array_y)
sum_xx = sum(array_x*array_x)
sum_yy = sum(array_y*array_y)
```

In [7]:
```
# Se obtienen los valores
s_xy = sum_xy-(1/n)*sum_x*sum_y
s_xx = sum_xx-(1/n)*sum_x**2

beta_1 = s_xy / s_xx
beta_0 = (1/n)*sum_y - beta_1*(1/n)*sum_x

# Imprimir los resultados
print("La estimación de los parámetros para el modelo de regresión son: ")
print("beta1: "+str(beta_1))
print("beta0: "+str(beta_0))
```

```
La estimación de los parámetros para el modelo de regresión son:
beta1: 0.1086107819535774
beta0: 49.07163369547534
```

Estos resultados nos llevan a concluir el siguiente modelo de regresión para los datos dados:

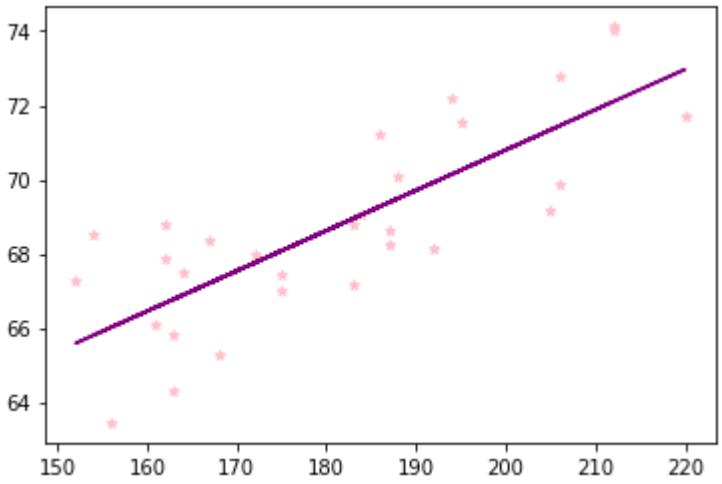
$$y = \hat{\beta}_0 + \hat{\beta}_1 * x = 49.07163 + 0.10861 * x$$

Ahora agregaremos esta línea dentro del gráfico

In [8]:
```
import matplotlib.pyplot as plt

plt.scatter(x=Altura , y=Peso,  marker='*', c='pink', s=25)
plt.plot(array_x, beta_0 + beta_1 * array_x, '-', c='purple')
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x21b2e431308>]
```



REGLAS DE ASOCIACIÓN

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.

In [9]:
```
pip install apyori
```

```
Requirement already satisfied: apyori in c:\users\jln\anaconda3\lib\site-pack
ages (1.1.2)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from apyori import apriori
```

In [11]:
```
datos={'ID':[1,2,3,4,5],'Transacciones':["A" "B" "C" "E","B" "E","C" "D" "E",
"A" "C" "D","A" "C" "E"]}
```

In [12]:
```
Tabla=pd.DataFrame(datos,columns=['ID','Transacciones'])
Tabla
```

Out[12]:

	ID	Transacciones
0	1	ABCE
1	2	BE
2	3	CDE
3	4	ACD
4	5	ACE

In [13]:
```
records = []
for i in range(0, 5):
    records.append([str(Tabla.Transacciones[i][j]) for j in range(0,len(Tabla.
Transacciones[i]))])
```

In [14]:
```
rules = apriori(records, min_support = 0.5, min_confidence = 0.5, min_lift = 0
, min_length =1)
```

In [15]:
```
results = list(rules)
```

In [16]:
```
antecedents = [tuple(item[0]) for result in results for item in result[2]]
consecuents = [tuple(item[1]) for result in results for item in result[2]]
supports = [result[1] for result in results for item in result[2]]
confidences = [item[2] for result in results for item in result[2]]
lifts = [item[3] for result in results for item in result[2]]
table = list(zip(antecedents, consecuents, supports, confidences, lifts))
```

In [17]:
```
TablaDeResultados=pd.DataFrame(table, columns = ['Antecedente', 'Consecuente',
'Soporte','Confianza', 'Lift'])
TablaDeResultados
```

Out[17]:

	Antecedente	Consecuente	Soporte	Confianza	Lift
0	()	(A,)	0.6	0.60	1.0000
1	()	(C,)	0.8	0.80	1.0000
2	()	(E,)	0.8	0.80	1.0000
3	()	(A, C)	0.6	0.60	1.0000
4	(A,)	(C,)	0.6	1.00	1.2500
5	(C,)	(A,)	0.6	0.75	1.2500
6	()	(E, C)	0.6	0.60	1.0000
7	(C,)	(E,)	0.6	0.75	0.9375
8	(E,)	(C,)	0.6	0.75	0.9375

K=1

A *Soporte=0.6 Confianza=0.6 Lift=1*

C *Soporte=0.8 Confianza=0.8 Lift=1*

E *Soporte=0.8 Confianza=0.8 Lift=1*

K=2

(A,C) *Soporte=0.6 Confianza=0.6 Lift=1*

A->C *Soporte=0.6 Confianza=1 Lift=1.25*

C->A *Soporte=0.6 Confianza=0.75 Lift=1.25*

(C,E) *Soporte=0.6 Confianza=0.6 Lift=1*

C->E *Soporte=0.6 Confianza=0.75 Lift=0.9375*

E->C *Soporte=0.6 Confianza=0.75 Lift=0.9375*

El nivel de soporte más alto con umbral de 0.5 es **K = 2**.

Con un umbral de 0.5 obtenemos las siguientes reglas de asociación:

- A -> C
- C -> A
- C -> E
- E -> C

Un **lift>1** indica que ese conjunto aparece una cantidad de veces superior a lo esperado bajo condiciones de independencia, por lo que se puede intuir que existe una relación que hace que los productos se encuentren en el conjunto más veces de lo normal.

Por lo que las mejores reglas de asociación son **A->C** y **C->A**