Comprensión de los Datos

In [3]:
```
#importa librerías
import pandas as pd
```

Descripción de Variables

Pclass Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd): Categórica Nominal survival Survival (0 = No; 1 = Yes)
name Name
sex Sex
age Age
sibsp Number of Siblings/Spouses Aboard
parch Number of Parents/Children Aboard
ticket Ticket Number
fare Passenger Fare (British pound)
cabin Cabin
embarked Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
boat Lifeboat
body Body Identification Number
home.dest Home/Destination

Ejemplo: Crear un objeto DataFrame con base en un archivo .csv

In [4]:
```
#lee archivo csv
df = pd.read_csv('titanic.csv')
```

In [5]:
```
#Usa función shape para revisar el total de renglones y columnas
df.shape
```

Out[5]: (891, 12)

In [6]:
```
#Revisa los primeros 5 renglones del dataset usando la función head()
df.head(5)
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [7]:
```
#Revisa los últimos 5 renglones del dataset usando la función tail()
df.tail(5)
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75

In [8]:
```
#Revisa la información mas completa del conjunto de datos usando la función info()
#Muestra el total de datos, las columnas y su tipo correspondiente, dice si contien
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [9]:
```
#revisa cuántos valores únicos tiene cada atributo del archivo usando la función nu
df.nunique()
```

Out[9]:
```
PassengerId    891
Survived         2
Pclass           3
Name           891
Sex              2
Age             88
SibSp            7
Parch            7
Ticket         681
Fare           248
Cabin          147
Embarked         3
dtype: int64
```

Exploración de Datos

In [64]:
```
#utiliza la función describe() para obtener estadística básica. se puede incluir -0
df.describe(),df.describe(include="object")
```

Out[64]:
```
(       PassengerId    Survived      Pclass         Age       SibSp  \
 count   891.000000  891.000000  891.000000  714.000000  891.000000
 mean    446.000000    0.383838    2.308642   29.699118    0.523008
 std     257.353842    0.486592    0.836071   14.526497    1.102743
 min       1.000000    0.000000    1.000000    0.420000    0.000000
 25%     223.500000    0.000000    2.000000   20.125000    0.000000
 50%     446.000000    0.000000    3.000000   28.000000    0.000000
 75%     668.500000    1.000000    3.000000   38.000000    1.000000
 max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare  familySize
 count  891.000000  891.000000  891.000000
 mean     0.381594   32.204208    1.904602
 std      0.806057   49.693429    1.613459
 min      0.000000    0.000000    1.000000
 25%      0.000000    7.910400    1.000000
 50%      0.000000   14.454200    1.000000
 75%      0.000000   31.000000    2.000000
 max      6.000000  512.329200   11.000000  ,
                          Name   Sex  Ticket Cabin Embarked
 count                     891   891     891   204      889
 unique                    891     2     681   147        3
 top     Braund, Mr. Owen Harris  male  347082    G6        S
 freq                        1   577       7     4      644)
```

In [17]:
```
#Revisa Valores nulos con funcion isnull().sum()
df.isnull().sum()
```

Out[17]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [19]:
```
#Revisar valores únicos por columna usando función unique(): nombre-columna.unique(
df.PassengerId.unique(), df.Survived.unique()
```

Out[19]:
```
(array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104,
       105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
       118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
       131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
       144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
       157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169,
       170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182,
       183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
       196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
       209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
       222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234,
       235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247,
       248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260,
       261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273,
       274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286,
       287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299,
       300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312,
       313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325,
       326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338,
       339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351,
       352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364,
       365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377,
       378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390,
       391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403,
       404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416,
       417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429,
       430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442,
       443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455,
       456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468,
       469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481,
       482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494,
       495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507,
       508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520,
       521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533,
       534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546,
       547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559,
       560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572,
       573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585,
       586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598,
       599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611,
       612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624,
       625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637,
       638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650,
       651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663,
       664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676,
       677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,
       690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702,
       703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715,
       716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728,
```

```
        729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741,
        742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754,
        755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767,
        768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780,
        781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793,
        794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806,
        807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819,
        820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832,
        833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845,
        846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858,
        859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871,
        872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884,
        885, 886, 887, 888, 889, 890, 891]),
 array([0, 1]))
```

Variables Cuantitativas

Medidas de tendencia central

In [ ]:
```
#Edad
#Se puede obtener la media, mediana y moda para
mean_age = titanic['Age'].mean()
median_age =titanic['Age'].median()
mode_age = titanic['Age'].mode()
print("Mean_age:",mean_age)
print("Median_age:",median_age)
print("Mode_age:",mode_age)
```

```
Mean_age: 29.69911764705882
Median_age: 28.0
Mode_age: 0    24.0
dtype: float64
```

Conclusiones: La edad promedio fue 29 La edad al centro es 28 La edad más repetida fue de 24

Variables Categóricas

In [20]:
```
#Para conteo  de cada valor en una columna, en orden descendente usar función value
# nombreDataframe.columna.value_counts()
# nombreDataframe['columna'].value_counts()
df.Survived.value_counts()
```

Out[20]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

In [34]:
```
#Revisa conteo de varias columnas
df[['PassengerId', 'Survived']].value_counts()
```

Out[34]:
```
PassengerId  Survived
1            0           1
2            1           1
3            1           1
4            1           1
5            0           1
                        ..
887          0           1
888          1           1
889          0           1
890          1           1
891          0           1
Name: count, Length: 891, dtype: int64
```

In [36]:
```
# Crear variable familySize que incluya la suma de las columnas SibSp y Parch
df['familySize'] = df['SibSp'] + df['Parch'] + 1

# Mostrar el total por cada tamaño de familia
df['familySize'].value_counts()
```

Out[36]:
```
familySize
1     537
2     161
3     102
4      29
6      22
5      15
7      12
11      7
8       6
Name: count, dtype: int64
```

Consulta

In [38]:
```
# df.iloc[i]: Accede a la fila en la posición i.
# Acceder a la primera fila
df.iloc[0]
```

Out[38]:
```
PassengerId                                                     2
Survived                                                        1
Pclass                                                          1
Name           Cumings, Mrs. John Bradley (Florence Briggs Th...
Sex                                                       female
Age                                                         38.0
SibSp                                                          1
Parch                                                          0
Ticket                                                  PC 17599
Fare                                                     71.2833
Cabin                                                        C85
Embarked                                                       C
familySize                                                     2
Name: 1, dtype: object
```

In [43]:
```
# Acceder a las dos primeras filas
df.iloc[[0,1]]
```

Out[43]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	

In [47]:
```
#Seleccionar columnas, indicando entre corchetes [nombreColumna, nombreColumna]
df[['PassengerId', 'Survived']]
```

Out[47]:

	PassengerId	Survived
0	1	0
1	2	1
2	3	1
3	4	1
4	5	0
...	...	...
886	887	0
887	888	1
888	889	0
889	890	1
890	891	0

891 rows × 2 columns

In [48]:
```
#Selección de filas [indicar dataframe[columna] operador valor]
df[df['Pclass'] == 1].head()
```

Out[48]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000

In [57]:
```
#ordenar usando funcion sort_values(by=atributo, ascending=True/false)
df.sort_values(by='Age', ascending=True).head(), df.sort_values(by='Fare', ascendin
```

Out[57]:
```
(     PassengerId  Survived  Pclass                                Name     Sex  \
803           804         1       3  Thomas, Master. Assad Alexander    male
755           756         1       2        Hamalainen, Master. Viljo    male
644           645         1       3           Baclini, Miss. Eugenie  female
469           470         1       3    Baclini, Miss. Helene Barbara  female
78             79         1       2    Caldwell, Master. Alden Gates    male

       Age  SibSp  Parch  Ticket     Fare Cabin Embarked  familySize
803  0.42      0      1    2625   8.5167   NaN        C           2
755  0.67      1      1  250649  14.5000   NaN        S           3
644  0.75      2      1    2666  19.2583   NaN        C           4
469  0.75      2      1    2666  19.2583   NaN        C           4
78   0.83      0      2  248738  29.0000   NaN        S           3  ,
     PassengerId  Survived  Pclass                                Name  \
679          680         1       1  Cardeza, Mr. Thomas Drake Martinez
258          259         1       1                     Ward, Miss. Anna
737          738         1       1              Lesurer, Mr. Gustave J
88            89         1       1          Fortune, Miss. Mabel Helen
438          439         0       1                    Fortune, Mr. Mark

        Sex   Age  SibSp  Parch    Ticket      Fare        Cabin Embarked  \
679    male  36.0      0      1  PC 17755  512.3292  B51 B53 B55        C
258  female  35.0      0      0  PC 17755  512.3292          NaN        C
737    male  35.0      0      0  PC 17755  512.3292         B101        C
88   female  23.0      3      2     19950  263.0000  C23 C25 C27        S
438    male  64.0      1      4     19950  263.0000  C23 C25 C27        S

     familySize
679           2
258           1
737           1
88            6
438           6  )
```

In [58]:
```
#Agrupar por un atributo y calcular función de agregación utilizando groupby(atribu
df.groupby('Pclass')['Fare'].mean()
```

Out[58]:
```
Pclass
1    84.154687
2    20.662183
3    13.675550
Name: Fare, dtype: float64
```

Crea un subconjunto de **titanic** para el costo mayor a 500

In [62]:
```
# usa el criterio para extraer solo los boletos caros con fare > 50
df_boletos_caros = df[df['Fare'] > 50]
df_boletos_caros.head()
```

Out[62]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000
31	32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female	NaN	1	0	PC 17569	146.5208

In [ ]: