Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 3

Jeet Bhalodi (23031701006)

1) First, you need to read the titanic dataset from local disk and display first five records

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

In [209…
```
df = pd.read_csv('titanic.csv')
```

In [6]:
```
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

2) Identify Nominal, Ordinal, Binary and Numeric attributes from data sets and display all values.

In [24]:
```
numeric=["PassengerId","Age","SibSp","Parch","Fare"]
binary = ["Survived","Sex"]
ordinal=["Pclass"]
nominal = ["Name","Ticket","Cabin","Embarked"]
```

In [42]:
```
df['PassengerId'].unique()
```

Out[42]:
```
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], dtype=int64)
```

In [62]:
```
df['Survived'].unique()
```

Out[62]:
```
array([0, 1], dtype=int64)
```

In [20]:
```
df['Pclass'].unique()
```

Out[20]:
```
array([3, 1, 2], dtype=int64)
```

In [58]:
```
df['Name'].nunique()
```

Out[58]:
```
891
```

In [66]:
```
df['Sex'].unique()
```

Out[66]:
```
array(['male', 'female'], dtype=object)
```

In [48]:
```
df['Age'].unique()
```

Out[48]:
```
array([22.  , 38.  , 26.  , 35.  ,   nan, 54.  ,  2.  , 27.  , 14.  ,
        4.  , 58.  , 20.  , 39.  , 55.  , 31.  , 34.  , 15.  , 28.  ,
        8.  , 19.  , 40.  , 66.  , 42.  , 21.  , 18.  ,  3.  ,  7.  ,
       49.  , 29.  , 65.  , 28.5 ,  5.  , 11.  , 45.  , 17.  , 32.  ,
       16.  , 25.  ,  0.83, 30.  , 33.  , 23.  , 24.  , 46.  , 59.  ,
       71.  , 37.  , 47.  , 14.5 , 70.5 , 32.5 , 12.  ,  9.  , 36.5 ,
       51.  , 55.5 , 40.5 , 44.  ,  1.  , 61.  , 56.  , 50.  , 36.  ,
       45.5 , 20.5 , 62.  , 41.  , 52.  , 63.  , 23.5 ,  0.92, 43.  ,
       60.  , 10.  , 64.  , 13.  , 48.  ,  0.75, 53.  , 57.  , 80.  ,
       70.  , 24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 34.5 , 74.  ])
```

In [28]:
```
df['SibSp'].unique()
```

Out[28]:
```
array([1, 0, 3, 4, 2, 5, 8], dtype=int64)
```

In [30]:
```
df['Parch'].unique()
```

Out[30]:
```
array([0, 1, 2, 5, 3, 4, 6], dtype=int64)
```

In [50]:
```
df['Ticket'].unique()
```

Out[50]:
```
array(['A/5 21171', 'PC 17599', 'STON/O2. 3101282', '113803', '373450',
       '330877', '17463', '349909', '347742', '237736', 'PP 9549',
       '113783', 'A/5. 2151', '347082', '350406', '248706', '382652',
       '244373', '345763', '2649', '239865', '248698', '330923', '113788',
       '347077', '2631', '19950', '330959', '349216', 'PC 17601',
       'PC 17569', '335677', 'C.A. 24579', 'PC 17604', '113789', '2677',
       'A./5. 2152', '345764', '2651', '7546', '11668', '349253',
       'SC/Paris 2123', '330958', 'S.C./A.4. 23567', '370371', '14311',
       '2662', '349237', '3101295', 'A/4. 39886', 'PC 17572', '2926',
       '113509', '19947', 'C.A. 31026', '2697', 'C.A. 34651', 'CA 2144',
       '2669', '113572', '36973', '347088', 'PC 17605', '2661',
       'C.A. 29395', 'S.P. 3464', '3101281', '315151', 'C.A. 33111',
       'S.O.C. 14879', '2680', '1601', '348123', '349208', '374746',
       '248738', '364516', '345767', '345779', '330932', '113059',
       'SO/C 14885', '3101278', 'W./C. 6608', 'SOTON/OQ 392086', '343275',
       '343276', '347466', 'W.E.P. 5734', 'C.A. 2315', '364500', '374910',
       'PC 17754', 'PC 17759', '231919', '244367', '349245', '349215',
       '35281', '7540', '3101276', '349207', '343120', '312991', '349249',
       '371110', '110465', '2665', '324669', '4136', '2627',
       'STON/O 2. 3101294', '370369', 'PC 17558', 'A4. 54510', '27267',
       '370372', 'C 17369', '2668', '347061', '349241',
       'SOTON/O.Q. 3101307', 'A/5. 3337', '228414', 'C.A. 29178',
       'SC/PARIS 2133', '11752', '7534', 'PC 17593', '2678', '347081',
       'STON/O2. 3101279', '365222', '231945', 'C.A. 33112', '350043',
       '230080', '244310', 'S.O.P. 1166', '113776', 'A.5. 11206',
       'A/5. 851', 'Fa 265302', 'PC 17597', '35851', 'SOTON/OQ 392090',
       '315037', 'CA. 2343', '371362', 'C.A. 33595', '347068', '315093',
       '363291', '113505', 'PC 17318', '111240', 'STON/O 2. 3101280',
       '17764', '350404', '4133', 'PC 17595', '250653', 'LINE',
       'SC/PARIS 2131', '230136', '315153', '113767', '370365', '111428',
       '364849', '349247', '234604', '28424', '350046', 'PC 17610',
       '368703', '4579', '370370', '248747', '345770', '3101264', '2628',
       'A/5 3540', '347054', '2699', '367231', '112277',
       'SOTON/O.Q. 3101311', 'F.C.C. 13528', 'A/5 21174', '250646',
       '367229', '35273', 'STON/O2. 3101283', '243847', '11813',
       'W/C 14208', 'SOTON/OQ 392089', '220367', '21440', '349234',
       '19943', 'PP 4348', 'SW/PP 751', 'A/5 21173', '236171', '347067',
       '237442', 'C.A. 29566', 'W./C. 6609', '26707', 'C.A. 31921',
       '28665', 'SCO/W 1585', '367230', 'W./C. 14263',
       'STON/O 2. 3101275', '2694', '19928', '347071', '250649', '11751',
       '244252', '362316', '113514', 'A/5. 3336', '370129', '2650',
       'PC 17585', '110152', 'PC 17755', '230433', '384461', '110413',
       '112059', '382649', 'C.A. 17248', '347083', 'PC 17582', 'PC 17760',
       '113798', '250644', 'PC 17596', '370375', '13502', '347073',
       '239853', 'C.A. 2673', '336439', '347464', '345778', 'A/5. 10482',
       '113056', '349239', '345774', '349206', '237798', '370373',
       '19877', '11967', 'SC/Paris 2163', '349236', '349233', 'PC 17612',
       '2693', '113781', '19988', '9234', '367226', '226593', 'A/5 2466',
       '17421', 'PC 17758', 'P/PP 3381', 'PC 17485', '11767', 'PC 17608',
       '250651', '349243', 'F.C.C. 13529', '347470', '29011', '36928',
       '16966', 'A/5 21172', '349219', '234818', '345364', '28551',
       '111361', '113043', 'PC 17611', '349225', '7598', '113784',
       '248740', '244361', '229236', '248733', '31418', '386525',
       'C.A. 37671', '315088', '7267', '113510', '2695', '2647', '345783',
       '237671', '330931', '330980', 'SC/PARIS 2167', '2691',
       'SOTON/O.Q. 3101310', 'C 7076', '110813', '2626', '14313',
```

'PC 17477', '11765', '3101267', '323951', 'C 7077', '113503',
'2648', '347069', 'PC 17757', '2653', 'STON/O 2. 3101293',
'349227', '27849', '367655', 'SC 1748', '113760', '350034',
'3101277', '350052', '350407', '28403', '244278', '240929',
'STON/O 2. 3101289', '341826', '4137', '315096', '28664', '347064',
'29106', '312992', '349222', '394140', 'STON/O 2. 3101269',
'343095', '28220', '250652', '28228', '345773', '349254',
'A/5. 13032', '315082', '347080', 'A/4. 34244', '2003', '250655',
'364851', 'SOTON/O.Q. 392078', '110564', '376564', 'SC/AH 3085',
'STON/O 2. 3101274', '13507', 'C.A. 18723', '345769', '347076',
'230434', '65306', '33638', '113794', '2666', '113786', '65303',
'113051', '17453', 'A/5 2817', '349240', '13509', '17464',
'F.C.C. 13531', '371060', '19952', '364506', '111320', '234360',
'A/S 2816', 'SOTON/O.Q. 3101306', '113792', '36209', '323592',
'315089', 'SC/AH Basle 541', '7553', '31027', '3460', '350060',
'3101298', '239854', 'A/5 3594', '4134', '11771', 'A.5. 18509',
'65304', 'SOTON/OQ 3101317', '113787', 'PC 17609', 'A/4 45380',
'36947', 'C.A. 6212', '350035', '315086', '364846', '330909',
'4135', '26360', '111427', 'C 4001', '382651', 'SOTON/OQ 3101316',
'PC 17473', 'PC 17603', '349209', '36967', 'C.A. 34260', '226875',
'349242', '12749', '349252', '2624', '2700', '367232',
'W./C. 14258', 'PC 17483', '3101296', '29104', '2641', '2690',
'315084', '113050', 'PC 17761', '364498', '13568', 'WE/P 5735',
'2908', '693', 'SC/PARIS 2146', '244358', '330979', '2620',
'347085', '113807', '11755', '345572', '372622', '349251',
'218629', 'SOTON/OQ 392082', 'SOTON/O.Q. 392087', 'A/4 48871',
'349205', '2686', '350417', 'S.W./PP 752', '11769', 'PC 17474',
'14312', 'A/4. 20589', '358585', '243880', '2689',
'STON/O 2. 3101286', '237789', '13049', '3411', '237565', '13567',
'14973', 'A./5. 3235', 'STON/O 2. 3101273', 'A/5 3902', '364848',
'SC/AH 29037', '248727', '2664', '349214', '113796', '364511',
'111426', '349910', '349246', '113804', 'SOTON/O.Q. 3101305',
'370377', '364512', '220845', '31028', '2659', '11753', '350029',
'54636', '36963', '219533', '349224', '334912', '27042', '347743',
'13214', '112052', '237668', 'STON/O 2. 3101292', '350050',
'349231', '13213', 'S.O./P.P. 751', 'CA. 2314', '349221', '8475',
'330919', '365226', '349223', '29751', '2623', '5727', '349210',
'STON/O 2. 3101285', '234686', '312993', 'A/5 3536', '19996',
'29750', 'F.C. 12750', 'C.A. 24580', '244270', '239856', '349912',
'342826', '4138', '330935', '6563', '349228', '350036', '24160',
'17474', '349256', '2672', '113800', '248731', '363592', '35852',
'348121', 'PC 17475', '36864', '350025', '223596', 'PC 17476',
'PC 17482', '113028', '7545', '250647', '348124', '34218', '36568',
'347062', '350048', '12233', '250643', '113806', '315094', '36866',
'236853', 'STON/O2. 3101271', '239855', '28425', '233639',
'349201', '349218', '16988', '376566', 'STON/O 2. 3101288',
'250648', '113773', '335097', '29103', '392096', '345780',
'349204', '350042', '29108', '363294', 'SOTON/O2 3101272', '2663',
'347074', '112379', '364850', '8471', '345781', '350047',
'S.O./P.P. 3', '2674', '29105', '347078', '383121', '36865',
'2687', '113501', 'W./C. 6607', 'SOTON/O.Q. 3101312', '374887',
'3101265', '12460', 'PC 17600', '349203', '28213', '17465',
'349244', '2685', '2625', '347089', '347063', '112050', '347087',
'248723', '3474', '28206', '364499', '112058', 'STON/O2. 3101290',
'S.C./PARIS 2079', 'C 7075', '315098', '19972', '368323', '367228',
'2671', '347468', '2223', 'PC 17756', '315097', '392092', '11774',

```
       'SOTON/O2 3101287', '2683', '315090', 'C.A. 5547', '349213',
       '347060', 'PC 17592', '392091', '113055', '2629', '350026',
       '28134', '17466', '233866', '236852', 'SC/PARIS 2149', 'PC 17590',
       '345777', '349248', '695', '345765', '2667', '349212', '349217',
       '349257', '7552', 'C.A./SOTON 34068', 'SOTON/OQ 392076', '211536',
       '112053', '111369', '370376'], dtype=object)
```

In [52]:
```
df['Fare'].unique()
```

Out[52]:
```
array([  7.25  ,  71.2833,   7.925 ,  53.1   ,   8.05  ,   8.4583,
        51.8625,  21.075 ,  11.1333,  30.0708,  16.7   ,  26.55  ,
        31.275 ,   7.8542,  16.    ,  29.125 ,  13.    ,  18.    ,
         7.225 ,  26.    ,   8.0292,  35.5   ,  31.3875, 263.    ,
         7.8792,   7.8958,  27.7208, 146.5208,   7.75  ,  10.5   ,
        82.1708,  52.    ,   7.2292,  11.2417,   9.475 ,  21.    ,
        41.5792,  15.5   ,  21.6792,  17.8   ,  39.6875,   7.8   ,
        76.7292,  61.9792,  27.75  ,  46.9   ,  80.    ,  83.475 ,
        27.9   ,  15.2458,   8.1583,   8.6625,  73.5   ,  14.4542,
        56.4958,   7.65  ,  29.    ,  12.475 ,   9.    ,   9.5   ,
         7.7875,  47.1   ,  15.85  ,  34.375 ,  61.175 ,  20.575 ,
        34.6542,  63.3583,  23.    ,  77.2875,   8.6542,   7.775 ,
        24.15  ,   9.825 ,  14.4583, 247.5208,   7.1417,  22.3583,
         6.975 ,   7.05  ,  14.5   ,  15.0458,  26.2833,   9.2167,
        79.2   ,   6.75  ,  11.5   ,  36.75  ,   7.7958,  12.525 ,
        66.6   ,   7.3125,  61.3792,   7.7333,  69.55  ,  16.1   ,
        15.75  ,  20.525 ,  55.    ,  25.925 ,  33.5   ,  30.6958,
        25.4667,  28.7125,   0.    ,  15.05  ,  39.    ,  22.025 ,
        50.    ,   8.4042,   6.4958,  10.4625,  18.7875,  31.    ,
       113.275 ,  27.    ,  76.2917,  90.    ,   9.35  ,  13.5   ,
         7.55  ,  26.25  ,  12.275 ,   7.125 ,  52.5542,  20.2125,
        86.5   , 512.3292,  79.65  , 153.4625, 135.6333,  19.5   ,
        29.7   ,  77.9583,  20.25  ,  78.85  ,  91.0792,  12.875 ,
         8.85  , 151.55  ,  30.5   ,  23.25  ,  12.35  , 110.8833,
       108.9   ,  24.    ,  56.9292,  83.1583, 262.375 ,  14.    ,
       164.8667, 134.5   ,   6.2375,  57.9792,  28.5   , 133.65  ,
        15.9   ,   9.225 ,  35.    ,  75.25  ,  69.3   ,  55.4417,
       211.5   ,   4.0125, 227.525 ,  15.7417,   7.7292,  12.    ,
       120.    ,  12.65  ,  18.75  ,   6.8583,  32.5   ,   7.875 ,
        14.4   ,  55.9   ,   8.1125,  81.8583,  19.2583,  19.9667,
        89.1042,  38.5   ,   7.725 ,  13.7917,   9.8375,   7.0458,
         7.5208,  12.2875,   9.5875,  49.5042,  78.2667,  15.1   ,
         7.6292,  22.525 ,  26.2875,  59.4   ,   7.4958,  34.0208,
        93.5   , 221.7792, 106.425 ,  49.5   ,  71.    ,  13.8625,
         7.8292,  39.6   ,  17.4   ,  51.4792,  26.3875,  30.    ,
        40.125 ,   8.7125,  15.    ,  33.    ,  42.4   ,  15.55  ,
        65.    ,  32.3208,   7.0542,   8.4333,  25.5875,   9.8417,
         8.1375,  10.1708, 211.3375,  57.    ,  13.4167,   7.7417,
         9.4833,   7.7375,   8.3625,  23.45  ,  25.9292,   8.6833,
         8.5167,   7.8875,  37.0042,   6.45  ,   6.95  ,   8.3   ,
         6.4375,  39.4   ,  14.1083,  13.8583,  50.4958,   5.    ,
         9.8458,  10.5167])
```

In [54]:
```
df['Cabin'].unique()
```

Out[54]:
```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49', 'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'B96 B98', 'E10', 'E44',
       'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46', 'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

In [56]:
```
df['Embarked'].unique()
```

Out[56]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

3) Identify symmetric and asymmetric binary attributes from data sets and display all values.

In [ ]:
```
symmetric = ['Sex']
asymmetric = ['Survived']
```

4) For each quantitative attribute, calculate its average, standard deviation, minimum, mode, range and maximum values.

In [102…
```
col = ["PassengerId","Survived","Pclass","Age","SibSp","Parch","Fare",]

for i in col:
    print(i,":-")
    print("    Mean = ",df[i].mean())
    print("    Std Dev. = ",df[i].std())
    print("    MIN = ",df[i].min())
    print("    MAX = ",df[i].max())
    print("    Range = ",df[i].max()-df[i].min())
    print("    Mode = ",df[i].mode().iloc[0])
```

```
PassengerId :-
    Mean =  446.0
    Std Dev. =  257.3538420152301
    MIN =  1
    MAX =  891
    Range =  890
    Mode =  1
Survived :-
    Mean =  0.3838383838383838
    Std Dev. =  0.4865924542648585
    MIN =  0
    MAX =  1
    Range =  1
    Mode =  0
Pclass :-
    Mean =  2.308641975308642
    Std Dev. =  0.8360712409770513
    MIN =  1
    MAX =  3
    Range =  2
    Mode =  3
Age :-
    Mean =  29.69911764705882
    Std Dev. =  14.526497332334044
    MIN =  0.42
    MAX =  80.0
    Range =  79.58
    Mode =  24.0
SibSp :-
    Mean =  0.5230078563411896
    Std Dev. =  1.1027434322934275
    MIN =  0
    MAX =  8
    Range =  8
    Mode =  0
Parch :-
    Mean =  0.38159371492704824
    Std Dev. =  0.8060572211299559
    MIN =  0
    MAX =  6
    Range =  6
    Mode =  0
Fare :-
    Mean =  32.204207968574636
    Std Dev. =  49.693428597180905
    MIN =  0.0
    MAX =  512.3292
    Range =  512.3292
    Mode =  8.05
```

6) For the qualitative attribute (class), count the frequency for each of its distinct values.

In [104…

```
q_atr = df['Pclass'].value_counts()
q_atr
```

Out[104…
```
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

7) It is also possible to display the summary for all the attributes simultaneously in a table using the describe() function. If an attribute is quantitative, it will display its mean, standard deviation and various quantiles (including minimum, median, and maximum) values. If an attribute is qualitative, it will display its number of unique values and the top (most frequent) values.

In [108…
```
df.describe(include='all')
```

Out[108…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	P
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.00
unique	NaN	NaN	NaN	891	2	NaN	NaN	
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	
freq	NaN	NaN	NaN	1	577	NaN	NaN	
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.38
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.80
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.00
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.00
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.00
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.00
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.00

8) For multivariate statistics, you can compute the covariance and correlation between pairs of attributes.

In [197…
```
# numeric_df = df.select_dtypes(include=['float64', 'int64'])
# covariance =
# correlation = numeric_df.corr()
```

In [211…
```
df.cov(numeric_only=True)
```

Out[211…

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
PassengerId	66231.000000	-0.626966	-7.561798	138.696504	-16.325843	-0.342697	161.8
Survived	-0.626966	0.236772	-0.137703	-0.551296	-0.018954	0.032017	6.2
Pclass	-7.561798	-0.137703	0.699015	-4.496004	0.076599	0.012429	-22.8
Age	138.696504	-0.551296	-4.496004	211.019125	-4.163334	-2.344191	73.8
SibSp	-16.325843	-0.018954	0.076599	-4.163334	1.216043	0.368739	8.7
Parch	-0.342697	0.032017	0.012429	-2.344191	0.368739	0.649728	8.6
Fare	161.883369	6.221787	-22.830196	73.849030	8.748734	8.661052	2469.4

In [213…

```
df.corr(numeric_only=True)
```

Out[213…

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

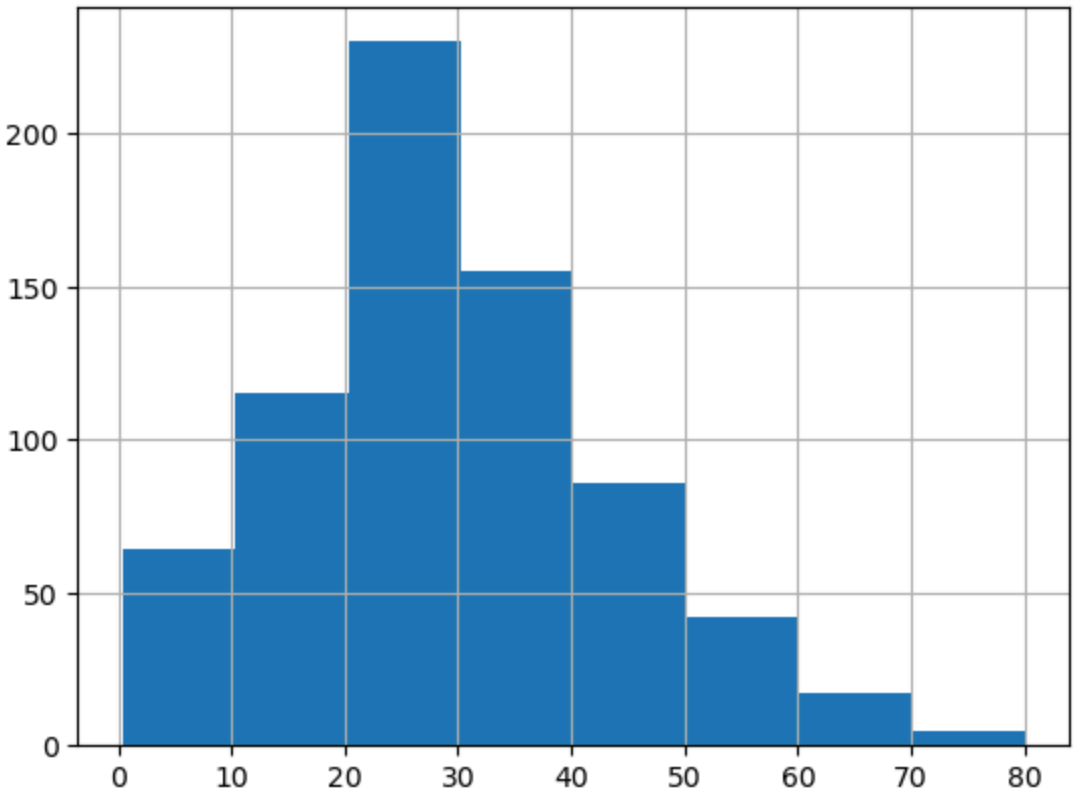
9) Display the histogram for Age attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

In [146…

```
import matplotlib.pyplot as plt
```

In [223…

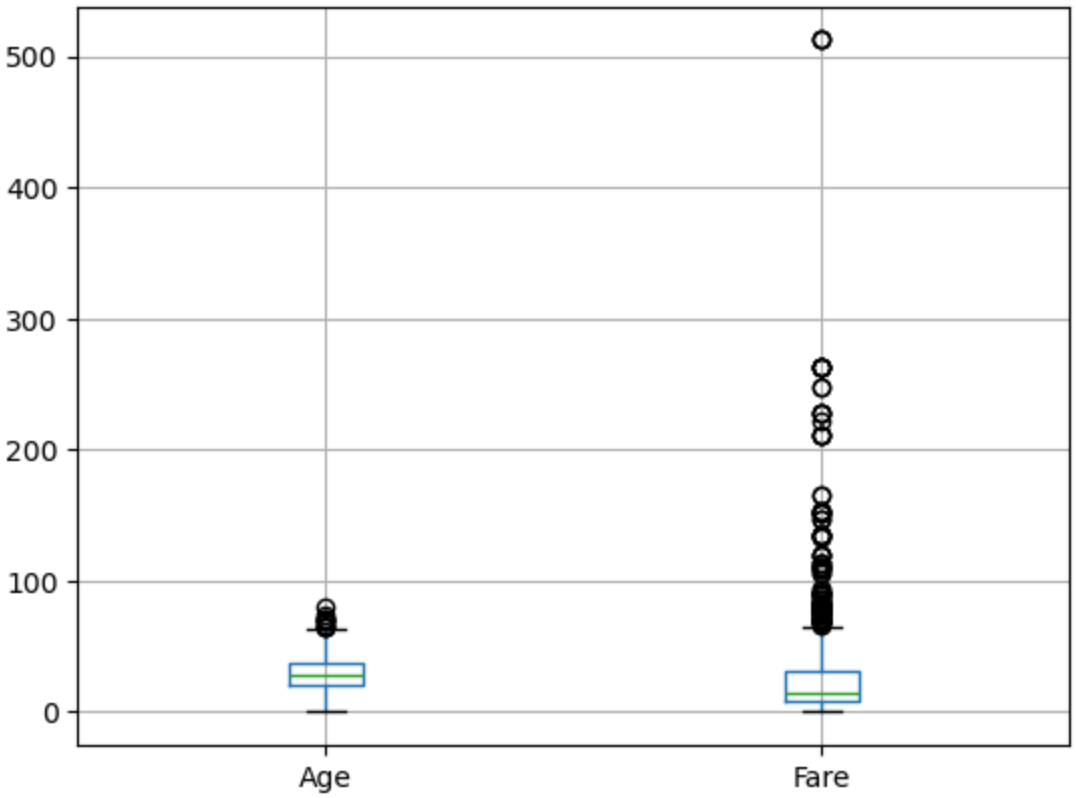
```
df['Age'].hist(bins=8)
plt.show()
```



10) A boxplot can also be used to show the distribution of values for each attribute.

In [227…

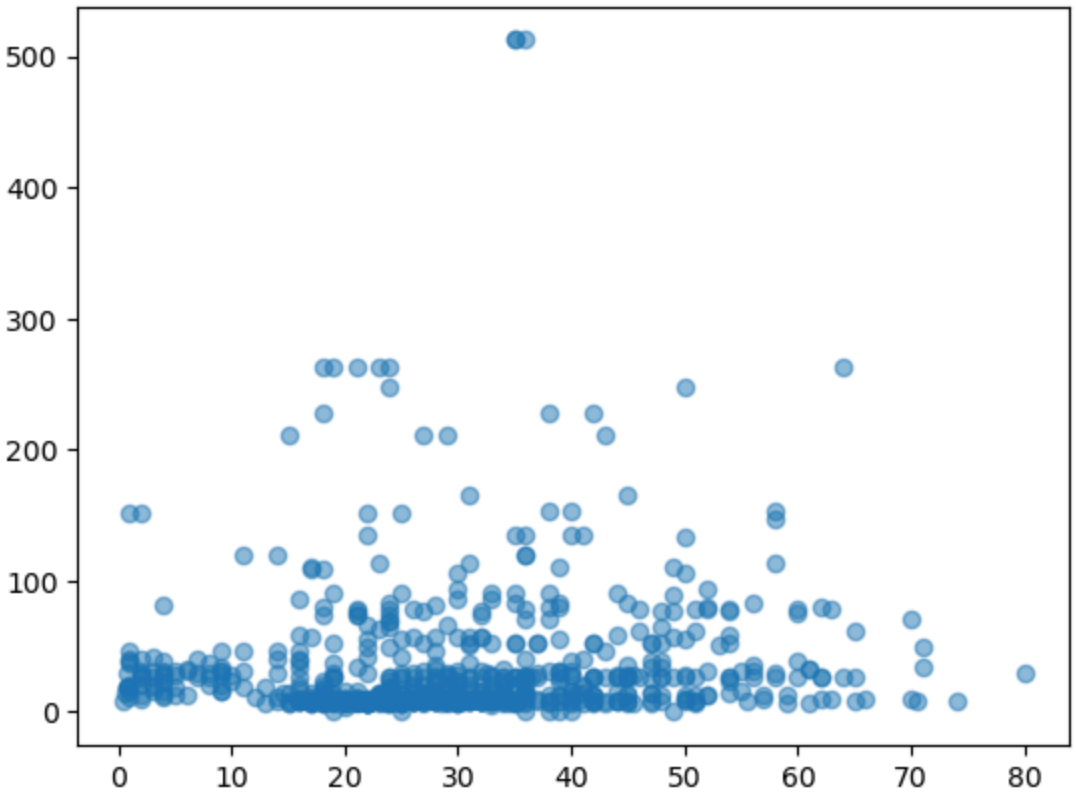
```
df[['Age', 'Fare']].boxplot()
plt.show()
```



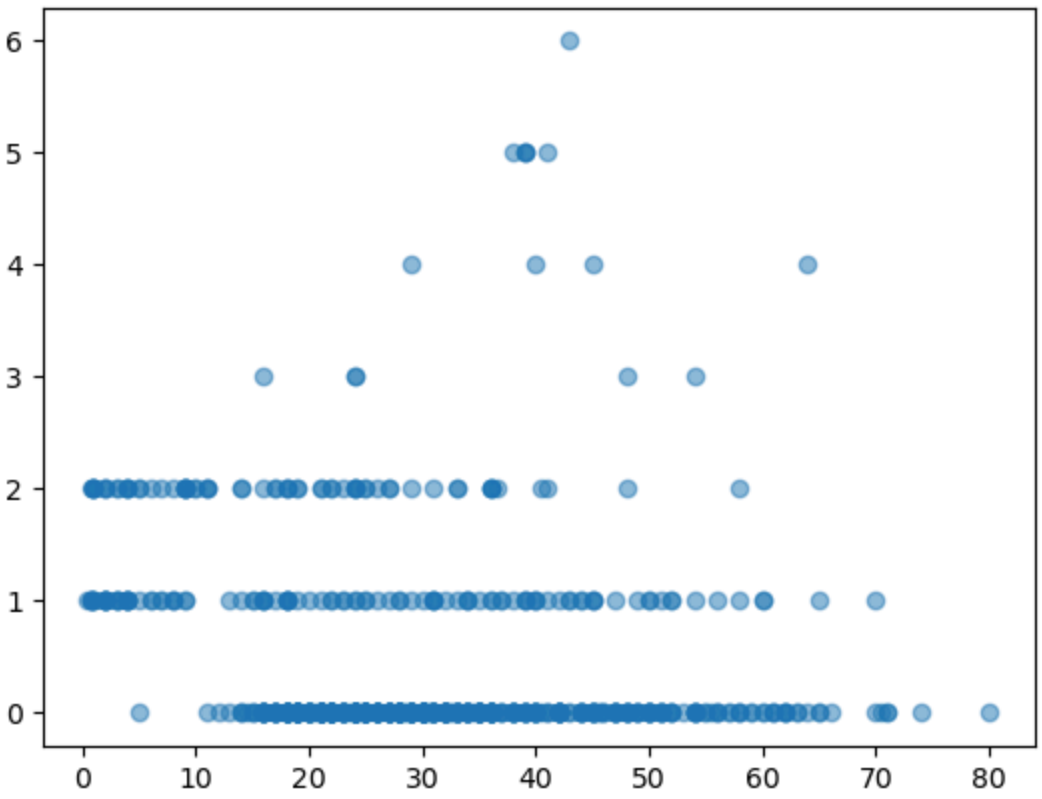
11) Display scatter plot for any 5 pair of attributes , we can use a scatter plot to visualize their joint distribution.

In [247…

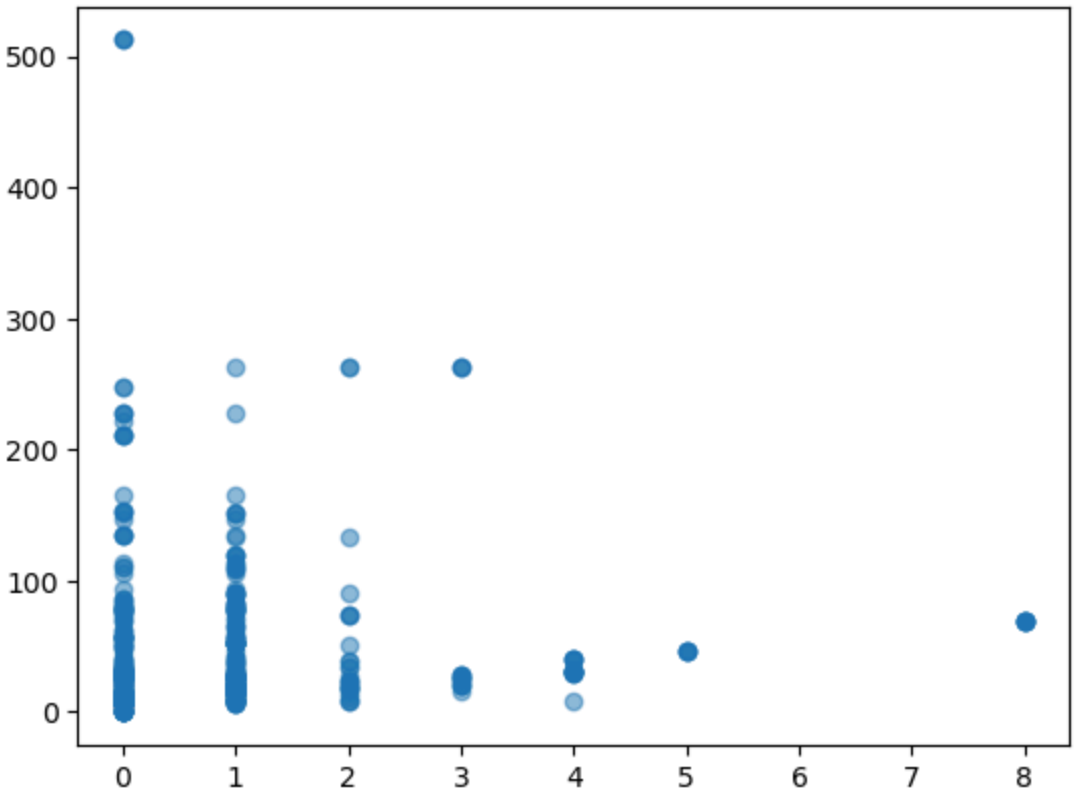
```
plt.scatter(df['Age'],df['Fare'],alpha=0.5)
plt.show()
```



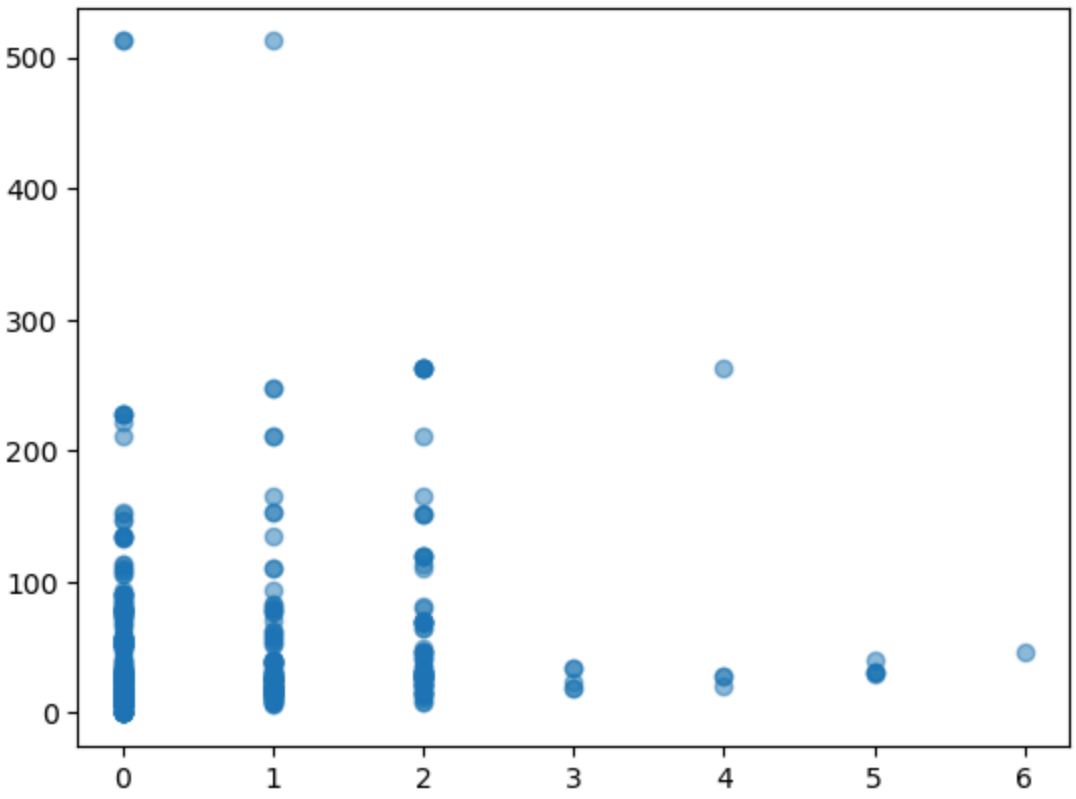
In [249…
```
plt.scatter(df['Age'],df['Parch'],alpha=0.5)
plt.show()
```



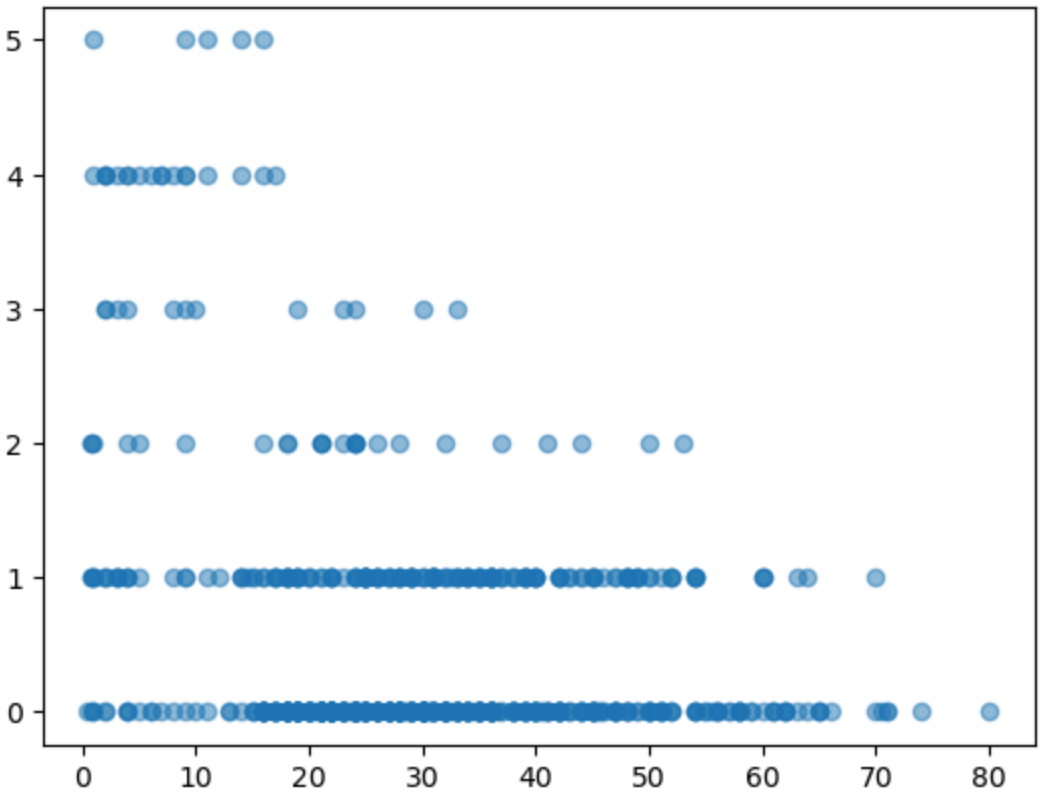
In [253…
```
plt.scatter(df['SibSp'],df['Fare'],alpha=0.5)
plt.show()
```



In [255…
```
plt.scatter(df['Parch'],df['Fare'],alpha=0.5)
plt.show()
```



In [257…
```
plt.scatter(df['Age'],df['SibSp'],alpha=0.5)
plt.show()
```



In [ ]: