Project: Housing and Property Sales

In this project, we will be doing a complete data analysis on housing and property sales to get useful information about it.

Dataset Features:

- Date of Sale (Datesold)
- Price
- Property type
- Number of bedrooms
- 4-digit postcode
- Year

Study outcome:

At the end of this project, you will learn how to

- Import Python libraries
- Read datasets in a CSV format
- Group the dataset by year, number of bedrooms, and property type.
- Find average house price
- Find how many houses and properties are sold.
- Calculate the average house price.
- Find the highest sales
- Create a histogram that shows the sales dataset, etc.

Import libraries

In [1]:
```
import pandas as pd # for data manipulation
import numpy as np # for numerical computation
import matplotlib.pyplot as plt # for data visualization
```

```
''' magic command that ensures matplotlib plots
are displayed directly inside the Jupyter Notebook cells '''
%matplotlib inline
```

Read the dataset

In [2]:
```
# define a variable called 'sales' to store the dataframe
sales = pd.read_csv('House Sales.csv')
```

In [3]:
```
# displays the dataframe to the console
sales
```

Out[3]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
0	07/02/2007 00:00	2607	525000	house	4	2007
1	27/02/2007 00:00	2906	290000	house	3	2007
2	07/03/2007 00:00	2905	328000	house	3	2007
3	09/03/2007 00:00	2905	380000	house	4	2007
4	21/03/2007 00:00	2906	310000	house	3	2007
...	...	...	...	...	...	...
28190	21/12/2018 00:00	2612	580000	unit	2	2018
28191	22/12/2018 00:00	2602	750000	house	3	2018
28192	24/12/2018 00:00	2914	640000	house	4	2018
28193	24/12/2018 00:00	2602	780000	house	3	2018
28194	24/12/2018 00:00	2603	1410000	house	4	2018

28195 rows × 6 columns

Question 1: Find how many houses and properties are sold?

In [4]:
```
# -----------------------
# each row corresponds to a sale, so we can use
# the count method in finding the number that was sold
# -----------------------

# count the total number of entries (rows) in the 'sales' object and store it in nu
num_sales = sales.count()

# print the result to see how many sales records exist
print(num_sales)
```

```
# Answer: 28195
```

```
Datesold         28195
Postcode         28195
Price            28195
Property Type    28195
Bedrooms         28195
Year             28195
dtype: int64
```

In [ ]:

Question 2: Calculate the average house price

In [5]:

```
# filter only houses
houses = sales[sales['Property Type'] == 'house']

# display the filtered DataFrame containing only houses
houses
```

Out[5]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
0	07/02/2007 00:00	2607	525000	house	4	2007
1	27/02/2007 00:00	2906	290000	house	3	2007
2	07/03/2007 00:00	2905	328000	house	3	2007
3	09/03/2007 00:00	2905	380000	house	4	2007
4	21/03/2007 00:00	2906	310000	house	3	2007
...	...	...	...	...	...	...
28185	21/12/2018 00:00	2602	910000	house	4	2018
28191	22/12/2018 00:00	2602	750000	house	3	2018
28192	24/12/2018 00:00	2914	640000	house	4	2018
28193	24/12/2018 00:00	2602	780000	house	3	2018
28194	24/12/2018 00:00	2603	1410000	house	4	2018

23530 rows × 6 columns

In [6]:

```
# calculate the average house price
average_price = houses['Price'].mean()

#displays the avearge price to the console
print("Average House Price:", average_price)
```

```
Average House Price: 645124.8735231619
```

In [ ]:

Question 3: Find the highest sales

In [7]:
```
# find the maximum value in the 'Price' column of the sales DataFrame
highest_sales = sales['Price'].max()

# displays the highest sales
print(f'Highest sales: {highest_sales}')
```

```
Highest sales: 8000000
```

In [8]:
```
# ---------------------
# finding more information about the house
# the property was sold on 2nd November 2015.
# the property is located in postcode 2611
# a 4-bedroom house—quite standard for a family home
# ---------------------

# returns the row with the highest price in the dataset
sales[sales['Price'].max() == sales['Price']]
```

Out[8]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
15186	02/11/2015 00:00	2611	8000000	house	4	2015

In [ ]:

Question 4: Group the dataset by years, number of bedrooms and property type

In [9]:
```
# Group the sales DataFrame by the 'Year' column
sales_y = sales.groupby('Year')

# Group the sales DataFrame by the 'Bedrooms' column
sales_b = sales.groupby('Bedrooms')

# Group the sales DataFrame by the 'Property Type' column
sales_p = sales.groupby('Property Type')
```

In [10]:
```
sales_y.groups
```

Out[10]:
```
{2007: [0, 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, 4
1, 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, 8
2, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, ...], 2008:
[147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 1
63, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 17
9, 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, 21
2, 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, 24
5, 246, ...], 2009: [786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 7
98, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 81
4, 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, 84
7, 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, 88
0, 881, 882, 883, 884, 885, ...], 2010: [2212, 2213, 2214, 2215, 2216, 2217, 2218,
2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 223
2, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2
246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259,
2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 227
3, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2
287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300,
2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, ...], 2011: [376
7, 3768, 3769, 3770, 3771, 3772, 3773, 3774, 3775, 3776, 3777, 3778, 3779, 3780, 3
781, 3782, 3783, 3784, 3785, 3786, 3787, 3788, 3789, 3790, 3791, 3792, 3793, 3794,
3795, 3796, 3797, 3798, 3799, 3800, 3801, 3802, 3803, 3804, 3805, 3806, 3807, 380
8, 3809, 3810, 3811, 3812, 3813, 3814, 3815, 3816, 3817, 3818, 3819, 3820, 3821, 3
822, 3823, 3824, 3825, 3826, 3827, 3828, 3829, 3830, 3831, 3832, 3833, 3834, 3835,
3836, 3837, 3838, 3839, 3840, 3841, 3842, 3843, 3844, 3845, 3846, 3847, 3848, 384
9, 3850, 3851, 3852, 3853, 3854, 3855, 3856, 3857, 3858, 3859, 3860, 3861, 3862, 3
863, 3864, 3865, 3866, ...], 2012: [5400, 5401, 5402, 5403, 5404, 5405, 5406, 540
7, 5408, 5409, 5410, 5411, 5412, 5413, 5414, 5415, 5416, 5417, 5418, 5419, 5420, 5
421, 5422, 5423, 5424, 5425, 5426, 5427, 5428, 5429, 5430, 5431, 5432, 5433, 5434,
5435, 5436, 5437, 5438, 5439, 5440, 5441, 5442, 5443, 5444, 5445, 5446, 5447, 544
8, 5449, 5450, 5451, 5452, 5453, 5454, 5455, 5456, 5457, 5458, 5459, 5460, 5461, 5
462, 5463, 5464, 5465, 5466, 5467, 5468, 5469, 5470, 5471, 5472, 5473, 5474, 5475,
5476, 5477, 5478, 5479, 5480, 5481, 5482, 5483, 5484, 5485, 5486, 5487, 5488, 548
9, 5490, 5491, 5492, 5493, 5494, 5495, 5496, 5497, 5498, 5499, ...], 2013: [7258,
7259, 7260, 7261, 7262, 7263, 7264, 7265, 7266, 7267, 7268, 7269, 7270, 7271, 727
2, 7273, 7274, 7275, 7276, 7277, 7278, 7279, 7280, 7281, 7282, 7283, 7284, 7285, 7
286, 7287, 7288, 7289, 7290, 7291, 7292, 7293, 7294, 7295, 7296, 7297, 7298, 7299,
7300, 7301, 7302, 7303, 7304, 7305, 7306, 7307, 7308, 7309, 7310, 7311, 7312, 731
3, 7314, 7315, 7316, 7317, 7318, 7319, 7320, 7321, 7322, 7323, 7324, 7325, 7326, 7
327, 7328, 7329, 7330, 7331, 7332, 7333, 7334, 7335, 7336, 7337, 7338, 7339, 7340,
7341, 7342, 7343, 7344, 7345, 7346, 7347, 7348, 7349, 7350, 7351, 7352, 7353, 735
4, 7355, 7356, 7357, ...], 2014: [9377, 9378, 9379, 9380, 9381, 9382, 9383, 9384,
9385, 9386, 9387, 9388, 9389, 9390, 9391, 9392, 9393, 9394, 9395, 9396, 9397, 939
8, 9399, 9400, 9401, 9402, 9403, 9404, 9405, 9406, 9407, 9408, 9409, 9410, 9411, 9
412, 9413, 9414, 9415, 9416, 9417, 9418, 9419, 9420, 9421, 9422, 9423, 9424, 9425,
9426, 9427, 9428, 9429, 9430, 9431, 9432, 9433, 9434, 9435, 9436, 9437, 9438, 943
9, 9440, 9441, 9442, 9443, 9444, 9445, 9446, 9447, 9448, 9449, 9450, 9451, 9452, 9
453, 9454, 9455, 9456, 9457, 9458, 9459, 9460, 9461, 9462, 9463, 9464, 9465, 9466,
9467, 9468, 9469, 9470, 9471, 9472, 9473, 9474, 9475, 9476, ...], 2015: [12240, 12
241, 12242, 12243, 12244, 12245, 12246, 12247, 12248, 12249, 12250, 12251, 12252,
12253, 12254, 12255, 12256, 12257, 12258, 12259, 12260, 12261, 12262, 12263, 1226
4, 12265, 12266, 12267, 12268, 12269, 12270, 12271, 12272, 12273, 12274, 12275, 12
276, 12277, 12278, 12279, 12280, 12281, 12282, 12283, 12284, 12285, 12286, 12287,
12288, 12289, 12290, 12291, 12292, 12293, 12294, 12295, 12296, 12297, 12298, 1229
9, 12300, 12301, 12302, 12303, 12304, 12305, 12306, 12307, 12308, 12309, 12310, 12
311, 12312, 12313, 12314, 12315, 12316, 12317, 12318, 12319, 12320, 12321, 12322,
12323, 12324, 12325, 12326, 12327, 12328, 12329, 12330, 12331, 12332, 12333, 1233
4, 12335, 12336, 12337, 12338, 12339, ...], 2016: [15888, 15889, 15890, 15891, 158
92, 15893, 15894, 15895, 15896, 15897, 15898, 15899, 15900, 15901, 15902, 15903, 1
5904, 15905, 15906, 15907, 15908, 15909, 15910, 15911, 15912, 15913, 15914, 15915,
15916, 15917, 15918, 15919, 15920, 15921, 15922, 15923, 15924, 15925, 15926, 1592
7, 15928, 15929, 15930, 15931, 15932, 15933, 15934, 15935, 15936, 15937, 15938, 15
939, 15940, 15941, 15942, 15943, 15944, 15945, 15946, 15947, 15948, 15949, 15950,
15951, 15952, 15953, 15954, 15955, 15956, 15957, 15958, 15959, 15960, 15961, 1596
2, 15963, 15964, 15965, 15966, 15967, 15968, 15969, 15970, 15971, 15972, 15973, 15
974, 15975, 15976, 15977, 15978, 15979, 15980, 15981, 15982, 15983, 15984, 15985,
15986, 15987, ...], 2017: [19796, 19797, 19798, 19799, 19800, 19801, 19802, 19803,
19804, 19805, 19806, 19807, 19808, 19809, 19810, 19811, 19812, 19813, 19814, 1981
5, 19816, 19817, 19818, 19819, 19820, 19821, 19822, 19823, 19824, 19825, 19826, 19
827, 19828, 19829, 19830, 19831, 19832, 19833, 19834, 19835, 19836, 19837, 19838,
19839, 19840, 19841, 19842, 19843, 19844, 19845, 19846, 19847, 19848, 19849, 1985
0, 19851, 19852, 19853, 19854, 19855, 19856, 19857, 19858, 19859, 19860, 19861, 19
862, 19863, 19864, 19865, 19866, 19867, 19868, 19869, 19870, 19871, 19872, 19873,
19874, 19875, 19876, 19877, 19878, 19879, 19880, 19881, 19882, 19883, 19884, 1988
5, 19886, 19887, 19888, 19889, 19890, 19891, 19892, 19893, 19894, 19895, ...], 201
8: [24337, 24338, 24339, 24340, 24341, 24342, 24343, 24344, 24345, 24346, 24347, 2
4348, 24349, 24350, 24351, 24352, 24353, 24354, 24355, 24356, 24357, 24358, 24359,
24360, 24361, 24362, 24363, 24364, 24365, 24366, 24367, 24368, 24369, 24370, 2437
1, 24372, 24373, 24374, 24375, 24376, 24377, 24378, 24379, 24380, 24381, 24382, 24
383, 24384, 24385, 24386, 24387, 24388, 24389, 24390, 24391, 24392, 24393, 24394,
24395, 24396, 24397, 24398, 24399, 24400, 24401, 24402, 24403, 24404, 24405, 2440
6, 24407, 24408, 24409, 24410, 24411, 24412, 24413, 24414, 24415, 24416, 24417, 24
418, 24419, 24420, 24421, 24422, 24423, 24424, 24425, 24426, 24427, 24428, 24429,
24430, 24431, 24432, 24433, 24434, 24435, 24436, ...]}
```

In [11]:

```
sales_b.groups
```

```
Out[11]: {0: [493, 6581, 7382, 7399, 7627, 7898, 8188, 8316, 8513, 9414, 11994, 12312, 1234
4, 13168, 13627, 13718, 15080, 16458, 16465, 18659, 18992, 19104, 19892, 21384, 22
574, 22974, 26796, 26868], 1: [21, 85, 93, 120, 141, 155, 156, 209, 254, 370, 392,
443, 460, 496, 509, 597, 624, 669, 671, 731, 787, 872, 902, 986, 989, 991, 1014, 1
099, 1110, 1148, 1149, 1229, 1230, 1251, 1280, 1321, 1327, 1342, 1409, 1411, 1445,
1468, 1493, 1512, 1513, 1533, 1551, 1587, 1623, 1639, 1716, 1818, 1832, 1845, 190
1, 1919, 1933, 1947, 2000, 2036, 2060, 2061, 2087, 2128, 2157, 2169, 2174, 2178, 2
199, 2222, 2232, 2260, 2276, 2324, 2429, 2491, 2500, 2539, 2544, 2579, 2604, 2622,
2648, 2730, 2777, 2848, 2869, 2870, 2921, 2959, 2968, 3009, 3049, 3073, 3078, 309
1, 3118, 3140, 3145, 3146, ...], 2: [13, 18, 29, 51, 54, 60, 62, 74, 76, 77, 79, 8
4, 104, 106, 137, 140, 147, 151, 178, 186, 192, 215, 218, 222, 232, 259, 265, 268,
274, 286, 320, 325, 327, 366, 373, 379, 390, 416, 426, 428, 434, 435, 437, 447, 44
8, 452, 461, 469, 472, 481, 497, 504, 510, 518, 528, 540, 549, 555, 568, 596, 615,
618, 644, 670, 674, 702, 709, 723, 724, 761, 773, 774, 785, 786, 790, 792, 800, 81
0, 821, 854, 864, 865, 871, 877, 878, 884, 886, 887, 891, 892, 909, 910, 915, 916,
917, 926, 927, 939, 940, 954, ...], 3: [1, 2, 4, 6, 8, 9, 10, 11, 17, 22, 23, 24,
25, 26, 27, 31, 33, 35, 36, 38, 39, 40, 41, 42, 43, 46, 53, 55, 56, 58, 61, 65, 6
7, 68, 78, 80, 81, 86, 87, 91, 92, 98, 102, 103, 108, 109, 110, 112, 113, 117, 11
8, 119, 121, 122, 123, 125, 126, 128, 129, 130, 131, 134, 138, 142, 144, 145, 146,
148, 160, 162, 163, 164, 165, 166, 167, 168, 173, 183, 185, 189, 190, 194, 195, 19
6, 197, 199, 204, 207, 210, 213, 219, 220, 223, 224, 225, 228, 229, 231, 233, 236,
...], 4: [0, 3, 5, 7, 12, 14, 15, 20, 28, 30, 32, 34, 44, 45, 48, 49, 50, 52, 57,
59, 63, 64, 66, 70, 71, 72, 73, 75, 82, 83, 89, 90, 94, 96, 99, 100, 105, 107, 11
1, 114, 115, 116, 127, 132, 133, 135, 136, 139, 143, 149, 150, 152, 153, 154, 157,
158, 159, 161, 169, 170, 171, 172, 175, 177, 180, 182, 184, 187, 188, 191, 193, 19
8, 201, 202, 206, 208, 211, 212, 214, 216, 217, 227, 230, 234, 235, 240, 241, 244,
245, 248, 249, 251, 252, 255, 257, 262, 263, 264, 267, 270, ...], 5: [16, 19, 37,
47, 69, 88, 95, 97, 101, 124, 174, 176, 179, 181, 200, 203, 205, 221, 226, 250, 26
6, 334, 352, 356, 385, 391, 402, 427, 522, 551, 553, 611, 614, 616, 621, 659, 678,
712, 745, 752, 809, 852, 875, 899, 953, 960, 977, 982, 983, 985, 1009, 1018, 1060,
1062, 1065, 1096, 1098, 1116, 1144, 1154, 1183, 1193, 1194, 1213, 1277, 1286, 128
7, 1310, 1354, 1367, 1379, 1402, 1452, 1460, 1511, 1524, 1547, 1596, 1606, 1622, 1
650, 1715, 1742, 1747, 1753, 1797, 1814, 1821, 1824, 1844, 1913, 1918, 1924, 1950,
1953, 1978, 2008, 2018, 2042, 2047, ...]}
```

```
In [12]: sales_p.groups
```

```
Out[12]: {'house': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 19, 20, 21, 2
2, 23, 24, 25, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 40, 41, 42, 43, 44, 45, 46,
47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 63, 64, 65, 66, 67, 69, 7
0, 71, 72, 73, 75, 78, 80, 81, 82, 83, 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, 1
13, ...], 'unit': [13, 18, 26, 29, 38, 39, 60, 62, 68, 74, 76, 77, 79, 84, 140, 14
1, 146, 155, 186, 192, 232, 254, 259, 268, 286, 287, 366, 370, 379, 392, 428, 435,
437, 443, 447, 448, 452, 460, 461, 472, 493, 496, 497, 504, 509, 510, 528, 549, 55
5, 565, 596, 597, 624, 644, 669, 670, 671, 724, 725, 731, 761, 773, 784, 785, 786,
787, 790, 792, 797, 810, 835, 848, 854, 864, 865, 871, 872, 877, 886, 887, 891, 89
2, 902, 908, 909, 910, 915, 916, 917, 926, 927, 934, 939, 940, 954, 955, 966, 974,
979, 986, ...]}
```

Question 5: Find the cheapest house sale in 2010

In [13]:
```
# retrieve all rows from the sales DataFrame where 'Property Type' is 'house'
House = sales_p.get_group('house')

# display the filtered DataFrame containing only house records
House
```

Out[13]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
0	07/02/2007 00:00	2607	525000	house	4	2007
1	27/02/2007 00:00	2906	290000	house	3	2007
2	07/03/2007 00:00	2905	328000	house	3	2007
3	09/03/2007 00:00	2905	380000	house	4	2007
4	21/03/2007 00:00	2906	310000	house	3	2007
...	...	...	...	...	...	...
28185	21/12/2018 00:00	2602	910000	house	4	2018
28191	22/12/2018 00:00	2602	750000	house	3	2018
28192	24/12/2018 00:00	2914	640000	house	4	2018
28193	24/12/2018 00:00	2602	780000	house	3	2018
28194	24/12/2018 00:00	2603	1410000	house	4	2018

23530 rows × 6 columns

In [14]:
```
# group the houses by year
House_Year = House.groupby('Year')
```

In [15]:
```
# filters the dataframe for houses sold in 2010
H_2010 = House_Year.get_group(2010)
```

In [16]:
```
# displays the dataframe with houses sold in 2010
H_2010
```

Out[16]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
2212	04/01/2010 00:00	2615	435000	house	3	2010
2213	05/01/2010 00:00	2904	712000	house	4	2010
2214	06/01/2010 00:00	2617	435000	house	4	2010
2215	06/01/2010 00:00	2606	1350000	house	5	2010
2216	07/01/2010 00:00	2905	612500	house	4	2010
...	...	...	...	...	...	...
3760	23/12/2010 00:00	2902	687000	house	4	2010
3761	23/12/2010 00:00	2602	767000	house	4	2010
3762	24/12/2010 00:00	2615	447000	house	4	2010
3763	24/12/2010 00:00	2913	457500	house	3	2010
3764	24/12/2010 00:00	2602	595000	house	3	2010

1374 rows × 6 columns

In [17]:
```
# displays the price of the cheapest house in 2010
H_2010['Price'].min()
```

Out[17]:
```
110000
```

In [18]:
```
# displays the price of the cheapest house in 2010 which is '110000'
# the number of bedrooms, property type, the date sold etc.
H_2010[H_2010['Price'].min() == H_2010['Price']]
```

Out[18]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
3262	21/09/2010 00:00	2606	110000	house	4	2010

In [ ]:

Question 6: Find the most expensive house sale in 2017

In [19]:
```
# filters the dataframe for houses sold in 2017
H_2017 = House_Year.get_group(2017)
```

In [20]:
```
H_2017
```

Out[20]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
19796	01/01/2017 00:00	2602	1095000	house	4	2017
19797	03/01/2017 00:00	2615	426000	house	3	2017
19798	05/01/2017 00:00	2615	410000	house	2	2017
19799	05/01/2017 00:00	2906	645000	house	4	2017
19800	05/01/2017 00:00	2914	745000	house	5	2017
...	...	...	...	...	...	...
24318	22/12/2017 00:00	2904	1000000	house	5	2017
24319	22/12/2017 00:00	2600	1450000	house	5	2017
24332	23/12/2017 00:00	2912	565000	house	3	2017
24334	28/12/2017 00:00	2905	520000	house	3	2017
24336	29/12/2017 00:00	2905	590000	house	3	2017

3630 rows × 6 columns

In [21]:

```
# prints the most expensive house in 2017
H_2017['Price'].max()
```

Out[21]:

```
4700000
```

In [22]:

```
# filter the H_2017 DataFrame to return the row(s) where the 'Price'
# is equal to the minimum price in the 'Price' column
H_2017[H_2017['Price'].min() == H_2017['Price']]
```

Out[22]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
19985	02/02/2017 00:00	2607	190000	house	1	2017

In [ ]:

```

```

Question 7: Find the most expensive house with 5 bedrooms

In [23]:

```
# filters the dataframe for houses with 5 bedrooms
B_5 = sales_b.get_group(5)
B_5
```

Out[23]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
16	02/07/2007 00:00	2914	800000	house	5	2007
19	06/07/2007 00:00	2615	535000	house	5	2007
37	07/08/2007 00:00	2904	815000	house	5	2007
47	21/08/2007 00:00	2902	418000	house	5	2007
69	21/09/2007 00:00	2603	1460000	house	5	2007
...	...	...	...	...	...	...
28097	14/12/2018 00:00	2905	720000	house	5	2018
28099	14/12/2018 00:00	2602	975000	house	5	2018
28112	15/12/2018 00:00	2607	1115000	house	5	2018
28115	16/12/2018 00:00	2611	1000000	house	5	2018
28181	21/12/2018 00:00	2615	820000	house	5	2018

1855 rows × 6 columns

In [24]:
```
# outputs the most expensive house
# with 5 bedrooms
B_5['Price'].max()
```

Out[24]:
```
7300000
```

In [25]:
```
B_5[B_5['Price'].max() == B_5['Price']]
```

Out[25]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
2589	22/04/2010 00:00	2603	7300000	house	5	2010

In [ ]:

Question 8: Find the cheapest unit

In [26]:
```
# retrieve all rows from the sales DataFrame where 'Property Type' is 'unit'
Unit = sales_p.get_group('unit')
Unit
```

Out[26]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
13	27/06/2007 00:00	2606	300000	unit	2	2007
18	05/07/2007 00:00	2611	300000	unit	2	2007
26	19/07/2007 00:00	2607	480000	unit	3	2007
29	20/07/2007 00:00	2604	360000	unit	2	2007
38	07/08/2007 00:00	2617	385000	unit	3	2007
...	...	...	...	...	...	...
28186	21/12/2018 00:00	2615	323000	unit	2	2018
28187	21/12/2018 00:00	2604	475000	unit	2	2018
28188	21/12/2018 00:00	2914	495000	unit	3	2018
28189	21/12/2018 00:00	2602	535000	unit	3	2018
28190	21/12/2018 00:00	2612	580000	unit	2	2018

4665 rows × 6 columns

In [27]:
```
# this returns the cheapest price from the 'Price' column of the DataFrame named Un
Unit['Price'].min()
```

Out[27]:
```
85000
```

In [28]:
```
Unit[Unit['Price'].min() == Unit['Price']]
```

Out[28]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
12179	18/12/2014 00:00	2612	85000	unit	1	2014

In [ ]:

Question 9: Find the cheapest unit in 2008

In [29]:
```
# group the Unit DataFrame by the 'Year' column
unit_year = Unit.groupby('Year')

# display the groupby object,
#which shows that the data is grouped by year
unit_year
```

Out[29]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002309E5EBBB0>
```

In [30]:
```
# retrieve all rows from the Unit DataFrame for the year 2008
U_2008 = unit_year.get_group(2008)
```

```
# display the filtered DataFrame containing only units sold in 2008
U_2008
```

Out[30]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
155	21/01/2008 00:00	2600	315000	unit	1	2008
186	04/03/2008 00:00	2612	400000	unit	2	2008
192	05/04/2008 00:00	2612	438000	unit	2	2008
232	30/05/2008 00:00	2600	329000	unit	2	2008
254	23/06/2008 00:00	2606	289000	unit	1	2008
259	25/06/2008 00:00	2604	347300	unit	2	2008
268	02/07/2008 00:00	2612	350000	unit	2	2008
286	21/07/2008 00:00	2612	385000	unit	2	2008
287	22/07/2008 00:00	2617	415000	unit	3	2008
366	10/09/2008 00:00	2606	367000	unit	2	2008
370	12/09/2008 00:00	2602	270000	unit	1	2008
379	16/09/2008 00:00	2606	257000	unit	2	2008
392	19/09/2008 00:00	2604	420000	unit	1	2008
428	29/09/2008 00:00	2905	295000	unit	2	2008
435	30/09/2008 00:00	2602	312000	unit	2	2008
437	01/10/2008 00:00	2617	340000	unit	2	2008
443	03/10/2008 00:00	2601	325000	unit	1	2008
447	07/10/2008 00:00	2604	351000	unit	2	2008
448	07/10/2008 00:00	2612	415000	unit	2	2008
452	08/10/2008 00:00	2615	280500	unit	2	2008
460	10/10/2008 00:00	2612	330000	unit	1	2008
461	10/10/2008 00:00	2604	358000	unit	2	2008
472	14/10/2008 00:00	2612	395000	unit	2	2008
493	21/10/2008 00:00	2612	90000	unit	0	2008
496	22/10/2008 00:00	2602	231000	unit	1	2008
497	22/10/2008 00:00	2604	373000	unit	2	2008
504	23/10/2008 00:00	2606	313000	unit	2	2008
509	24/10/2008 00:00	2620	140500	unit	1	2008
510	24/10/2008 00:00	2612	432500	unit	2	2008
528	31/10/2008 00:00	2614	220000	unit	2	2008

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
549	07/11/2008 00:00	2602	280000	unit	2	2008
555	10/11/2008 00:00	2607	240000	unit	2	2008
565	14/11/2008 00:00	2604	405000	unit	3	2008
596	21/11/2008 00:00	2913	319950	unit	2	2008
597	21/11/2008 00:00	2612	400000	unit	1	2008
624	27/11/2008 00:00	2602	347000	unit	1	2008
644	01/12/2008 00:00	2912	280000	unit	2	2008
669	04/12/2008 00:00	2602	250000	unit	1	2008
670	04/12/2008 00:00	2602	300000	unit	2	2008
671	04/12/2008 00:00	2601	385000	unit	1	2008
724	15/12/2008 00:00	2606	295000	unit	2	2008
725	15/12/2008 00:00	2612	385000	unit	3	2008
731	16/12/2008 00:00	2601	279000	unit	1	2008
761	22/12/2008 00:00	2603	350000	unit	2	2008
773	23/12/2008 00:00	2606	315000	unit	2	2008
784	24/12/2008 00:00	2606	400000	unit	3	2008
785	24/12/2008 00:00	2612	440000	unit	2	2008

In [31]:
```
# displays the cheapest unit in 2008
U_2008['Price'].min()
```

Out[31]:
```
90000
```

In [32]:
```
# the cheapest unit in 2008 was 90000
# with no bedrooms
U_2008[U_2008['Price'].min() == U_2008['Price']]
```

Out[32]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
493	21/10/2008 00:00	2612	90000	unit	0	2008

In [ ]:
```

```

Question 10: Find the total amount of unit sales in 2016

In [33]:
```
# filters the dataframe for units sold in the year '2016'
U_2016 = unit_year.get_group(2016)
U_2016
```

Out[33]:

	Datesold	Postcode	Price	Property Type	Bedrooms	Year
15888	01/01/2016 00:00	2612	420000	unit	1	2016
15890	04/01/2016 00:00	2612	360000	unit	1	2016
15898	11/01/2016 00:00	2612	335000	unit	1	2016
15899	11/01/2016 00:00	2606	360000	unit	1	2016
15900	11/01/2016 00:00	2617	375000	unit	2	2016
...	...	...	...	...	...	...
19788	23/12/2016 00:00	2617	395000	unit	3	2016
19789	23/12/2016 00:00	2600	408000	unit	1	2016
19790	23/12/2016 00:00	2617	472000	unit	2	2016
19791	23/12/2016 00:00	2604	570000	unit	2	2016
19792	23/12/2016 00:00	2617	615000	unit	3	2016

695 rows × 6 columns

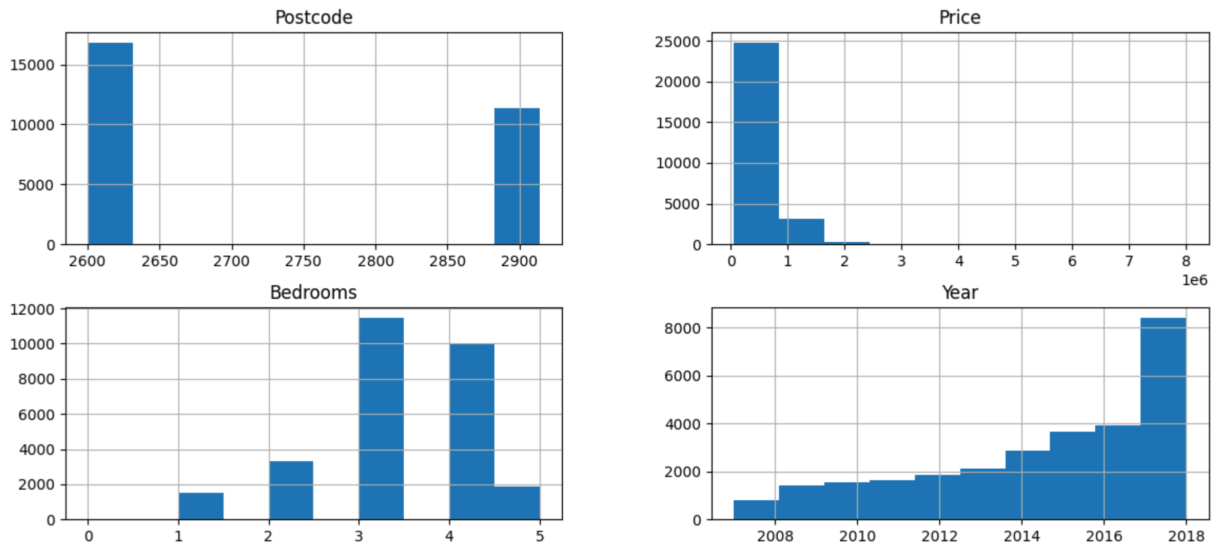
In [34]:
```
# prints the total amount of unit sales in 2016
U_2016['Price'].sum()
```

Out[34]:
```
296981819
```

In [ ]:
```
```

Question 11: Create a histogram that shows the Sales dataset

In [35]:
```
# create a histogram of the sales dataset
Histogram = sales.hist(figsize = (14, 6))
```


Postcode
Price
1e6
Bedrooms
Year