In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from scipy import stats
from matplotlib.lines import Line2D
import seaborn as sns
from sklearn.linear_model import LinearRegression
from IPython.display import Image
import statsmodels.api as sm
```

In [2]:
```
df = pd.read_csv("desktop/house_truncated.csv")
```

In [3]:
```
df.corr()
```

Out[3]:

| | Id | SalePrice | LotFrontage | LotArea | GrLivArea | OverallCond | YearBuilt | Tota |
|---|---|---|---|---|---|---|---|---|
| **Id** | 1.000000 | -0.021917 | -0.010601 | -0.033226 | 0.008273 | 0.012609 | -0.012713 | - |
| **SalePrice** | -0.021917 | 1.000000 | 0.351799 | 0.263843 | 0.708624 | -0.077856 | 0.522897 | |
| **LotFrontage** | -0.010601 | 0.351799 | 1.000000 | 0.426095 | 0.402797 | -0.059213 | 0.123349 | |
| **LotArea** | -0.033226 | 0.263843 | 0.426095 | 1.000000 | 0.263116 | -0.005636 | 0.014228 | |
| **GrLivArea** | 0.008273 | 0.708624 | 0.402797 | 0.263116 | 1.000000 | -0.079686 | 0.199010 | |
| **OverallCond** | 0.012609 | -0.077856 | -0.059213 | -0.005636 | -0.079686 | 1.000000 | -0.375983 | - |
| **YearBuilt** | -0.012713 | 0.522897 | 0.123349 | 0.014228 | 0.199010 | -0.375983 | 1.000000 | |
| **TotalBsmtSF** | -0.015415 | 0.613581 | 0.392075 | 0.260833 | 0.454868 | -0.171098 | 0.391452 | |
| **1stFlrSF** | 0.010496 | 0.605852 | 0.457181 | 0.299475 | 0.566024 | -0.144203 | 0.281986 | |
| **2ndFlrSF** | 0.005590 | 0.319334 | 0.080177 | 0.050986 | 0.687501 | 0.028942 | 0.010308 | - |
| **BsmtFullBath** | 0.002289 | 0.227122 | 0.100949 | 0.158155 | 0.034836 | -0.054942 | 0.187599 | |
| **BsmtHalfBath** | -0.020155 | -0.016844 | -0.007234 | 0.048046 | -0.018918 | 0.117821 | -0.038162 | - |
| **FullBath** | 0.005587 | 0.560664 | 0.198769 | 0.126031 | 0.630012 | -0.194149 | 0.468271 | |
| **HalfBath** | 0.006784 | 0.284108 | 0.053532 | 0.014259 | 0.415772 | -0.060769 | 0.242656 | - |
| **KitchenAbvGr** | 0.002951 | -0.135907 | -0.006069 | -0.017784 | 0.100063 | -0.087001 | -0.174800 | - |
| **GarageCars** | 0.016570 | 0.640409 | 0.285691 | 0.154871 | 0.467247 | -0.185758 | 0.537850 | |
| **PoolArea** | 0.057044 | 0.092404 | 0.206167 | 0.077672 | 0.170205 | -0.001985 | 0.004950 | |

In [4]:
```
X = [['GrLivArea','YearBuilt','GarageCars','LotFrontage','TotalBsmtSF','Str
```

I chose my explanatory variables based on the strength of their correlation with SalePrice.

In [5]:
```
df.GrLivArea.unique()
```

Out[5]:
```
array([1710, 1262, 1786, 1717, 2198, 1362, 1694, 2090, 1774, 1077, 1040,
       2324,  912, 1494, 1253,  854, 1004, 1296, 1114, 1339, 2376, 1108,
       1795, 1060, 1600,  900, 1704,  520, 1317, 1228, 1234, 1700, 1561,
       2452, 1097, 1297, 1057, 1152, 1324, 1328,  884,  938, 1150, 1752,
       2149, 1656, 1452,  955, 1470, 1176,  816, 1842, 1360, 1425, 1739,
       1720, 2945,  780, 1158, 1111, 1370, 2034, 2473, 2207, 1479,  747,
       2287, 2223,  845, 1718, 1086, 1605,  988,  952, 1285, 1768, 1230,
       2142, 1337, 1563, 1065, 1474, 2417, 1560, 1224, 1526,  990, 1235,
        964, 2291, 1588,  960,  835, 1225, 1610, 1732, 1535, 1226, 1818,
       1992, 1047,  789, 1517, 1844, 1855, 1430, 2696, 2259, 2320, 1458,
       1092, 1125, 3222, 1456, 1123, 1080, 1199, 1586,  754,  958,  840,
       1348, 1053, 2157, 2054, 1327, 1721, 1682, 1214, 1959, 1852, 1764,
        864, 1734, 1385, 1501, 1728, 1709,  875, 2035, 1344,  969, 1993,
       1252, 1200, 1096, 1968, 1947, 2462, 1232, 2668, 1541,  882, 1616,
       1355, 1867, 2161, 1707, 1382, 1767, 1651, 2158, 2060, 1920, 2234,
        968, 1525, 1802, 1340, 2082, 3608, 1217, 1593, 2727, 1431, 1726,
       3112, 2229, 1713, 1121, 1279, 1310,  848, 1284, 1442, 1696, 1100,
       2062, 1212, 1392, 1236, 1436, 1954, 1248, 1498, 2267, 1552, 2392,
       1302, 2520,  987, 1555, 1194, 2794,  894, 1960, 1414, 1744, 1487,
       1566,  866, 1440, 2110, 1872, 1928, 1375, 1668, 2144, 1306, 1625,
       1640, 1314, 1604, 1792, 2574, 1316,  764, 1422, 1511, 2192,  778,
       1113, 1939, 1363, 2270, 1632, 1548, 2121, 2022, 1982, 1468, 1575,
       1250,  858, 1396, 1919, 1716, 2263, 1644, 1003, 1558, 1950, 1743,
       1336, 3493, 2000, 2243, 1406,  861, 1944,  972, 1118, 2036, 1641,
       1432, 2353, 2646, 1472, 2596, 2468, 2730, 1163, 2978,  803, 1719,
       1383, 2134, 1192, 1056, 1629, 1358, 1638, 1922, 1536, 1621, 1215,
       1908,  841, 1684, 1112, 1577, 1478, 1626, 2728, 1869, 1453,  720,
       1595, 1167, 1142, 1352, 1924, 1505, 1574, 1394, 1268, 1287, 1664,
        752, 1319,  904,  914, 2466, 1856, 1800, 1691, 1301, 1797,  784,
       1953, 1269, 1184, 2332, 1367, 1961,  788, 1034, 1144, 1812, 1550,
       1288,  672, 1572, 1620, 1639, 1680, 2172, 2078, 1276, 1028, 2097,
       1400, 2624, 1134, 1602, 2630, 1196, 1389,  907, 1208, 1412, 1198,
       1365,  630, 1661,  694, 2402, 1573, 1258, 1689, 1888, 1886, 1376,
       1183,  813, 1533, 1756, 1590, 1242, 1663, 1666, 1203, 1935, 1135,
       1660, 1277, 1634, 1502, 1969, 1072, 1976, 1652,  970, 1493, 2643,
       1131, 1850, 1826, 1216,  999, 1073, 1484, 2414, 1304, 1578,  886,
       3228, 1820,  899, 1218, 1801, 1322, 1911, 1378, 1041, 1368, 2020,
       2119, 2344, 1796, 2080, 1294, 1244, 4676, 2398, 1266,  928, 2713,
        605, 2515, 1509,  827,  334, 1347, 1724, 1159, 1601, 1838, 2285,
        767, 1496, 2183, 1635,  768,  825, 2094, 1069, 1126, 2046, 1048,
       1446, 1557,  996, 1674, 2295, 1647, 2504, 2132,  943, 1692, 1109,
       1477, 1320, 1429, 2042, 2775, 2028,  838,  860, 1473,  935, 1582,
       2296,  924, 1402, 1556, 1904, 1915, 1986, 2008, 3194, 1029, 2153,
       1032, 1120, 1054,  832, 1828, 2262, 2614,  980, 1512, 1790, 1116,
       1520, 1350, 1750, 1554, 1411, 3395,  800, 1387,  796, 1567, 1518,
       1929, 2704, 1766,  981, 1094, 1839, 1665, 1510, 1469, 2113, 1486,
       2448, 1181, 1936, 2380, 1679, 1437, 1180, 1476, 1369, 1136, 1441,
        792,  923, 1291, 1761, 1102, 1419, 4316, 2519, 1539, 1137,  616,
       1148, 1391, 1164, 2576, 1824,  729, 1178, 2554, 2418,  971, 1742,
       1698, 1776, 1146, 2031,  948, 1349, 1464, 2715, 2256, 2640, 1529,
       1140, 2098, 1026, 1471, 1386, 2531, 1547, 2365, 1506, 1714, 1836,
       3279, 1220, 1117, 1973, 1204, 1614, 1603, 1110, 1342, 2084,  901,
       2087, 1145, 1062, 2013, 1895, 1564,  773, 3140, 1688, 2822, 1128,
       1428, 1576, 2138, 1309, 1044, 1008, 1052,  936, 1733, 1489, 1434,
       2126, 1223, 1829, 1516, 1067, 1559, 1099, 1482, 1165, 1416, 1701,
```

```
1775, 2358, 1646, 1445, 1779, 1481, 2654, 1426, 1039, 1372, 1002,
1949,  910, 2610, 2224, 1155, 1090, 2230,  892, 1712, 1393, 2217,
1683, 1068,  951, 2240, 2364, 1670,  902, 1063, 1636, 2057, 2274,
1015, 2002,  480, 1229, 2127, 2200, 1617, 1686, 2374, 1978, 1788,
2236, 1466,  925, 1905, 1500, 2069, 1971, 1962, 2403, 1381,  965,
1958, 2872, 1894, 1308, 1098, 1095,  918, 2019,  869, 1241, 2612,
2290, 1940, 2030, 1851, 1050,  944,  691, 1504,  985, 1657, 1522,
1271, 1022, 1082, 1132, 2898, 1264, 3082, 1654,  954, 1803, 2329,
2524, 2868, 1771,  930, 1977, 1989, 1523, 1364, 2184, 1991, 1338,
2337, 1103, 1154, 2260, 1571, 1611, 2521,  893, 1240, 1740, 1459,
1251, 1247, 1088,  438,  950, 2622, 2021, 1690, 1658, 1964,  833,
1012,  698, 1005, 1530, 1981,  974, 2210,  986, 1020, 1868, 2828,
1006, 1298,  932, 1811, 1265, 1580, 1876, 1671, 2108, 3627, 1261,
3086, 2345, 1343, 1124, 2514, 4476, 1130, 1221, 1699, 1624, 1804,
1622, 1863, 1630, 1074, 2196, 1283, 1845, 1902, 1211, 1846, 2136,
1490, 1138, 1933, 1702, 1507, 2620, 1190, 1188, 1784, 1948, 1141,
1173, 2076, 1553, 2058, 1405,  874, 2167, 1987, 1166, 1675, 1889,
2018, 3447, 1524, 1357, 1395, 2447, 1659, 1970, 2372, 5642, 1246,
1983, 2526, 1708, 1122, 1274, 2810, 2599, 2112, 1787, 1923,  708,
 774, 2792, 1334,  693, 1861,  872, 2169, 1913, 2156, 2634, 3238,
1865, 1078, 1980, 2601, 1738, 1475, 1374, 2633,  790, 2117, 1762,
2784, 1746, 1584, 1912, 2482, 1687, 1513, 1608, 2093, 1840, 1848,
1569, 2450, 2201,  804, 1537, 1932, 1725, 2555, 2007,  913, 1346,
2073, 2340, 1256])
```

In [6]:
```
df.YearBuilt.unique()
```

Out[6]:
```
array([2003, 1976, 2001, 1915, 2000, 1993, 2004, 1973, 1931, 1939, 1965,
       2005, 1962, 2006, 1960, 1929, 1970, 1967, 1958, 1930, 2002, 1968,
       2007, 1951, 1957, 1927, 1920, 1966, 1959, 1994, 1954, 1953, 1955,
       1983, 1975, 1997, 1934, 1963, 1981, 1964, 1999, 1972, 1921, 1945,
       1982, 1998, 1956, 1948, 1910, 1995, 1991, 2009, 1950, 1961, 1977,
       1985, 1979, 1885, 1919, 1990, 1969, 1935, 1988, 1971, 1952, 1936,
       1923, 1924, 1984, 1926, 1940, 1941, 1987, 1986, 2008, 1908, 1892,
       1916, 1932, 1918, 1912, 1947, 1925, 1900, 1980, 1989, 1992, 1949,
       1880, 1928, 1978, 1922, 1996, 2010, 1946, 1913, 1937, 1942, 1938,
       1974, 1893, 1914, 1906, 1890, 1898, 1904, 1882, 1875, 1911, 1917,
       1872, 1905])
```

In [7]:
```
df.GarageCars.unique()
```

Out[7]:
```
array([2, 3, 1, 0, 4])
```

In [8]:
```
df.LotFrontage.unique()
```

Out[8]:
```
array([ 65.,  80.,  68.,  60.,  84.,  85.,  75.,  nan,  51.,  50.,  70.,
        91.,  72.,  66., 101.,  57.,  44., 110.,  98.,  47., 108., 112.,
        74., 115.,  61.,  48.,  33.,  52., 100.,  24.,  89.,  63.,  76.,
        81.,  95.,  69.,  21.,  32.,  78., 121., 122.,  40., 105.,  73.,
        77.,  64.,  94.,  34.,  90.,  55.,  88.,  82.,  71., 120., 107.,
        92., 134.,  62.,  86., 141.,  97.,  54.,  41.,  79., 174.,  99.,
        67.,  83.,  43., 103.,  93.,  30., 129., 140.,  35.,  37., 118.,
        87., 116., 150., 111.,  49.,  96.,  59.,  36.,  56., 102.,  58.,
        38., 109., 130.,  53., 137.,  45., 106., 104.,  42.,  39., 144.,
       114., 128., 149., 313., 168., 182., 138., 160., 152., 124., 153.,
        46.])
```

In [9]:
```
df.TotalBsmtSF.unique()
```

Out[9]:
```
array([ 856, 1262,  920,  756, 1145,  796, 1686, 1107,  952,  991, 1040,
       1175,  912, 1494, 1253,  832, 1004,    0, 1114, 1029, 1158,  637,
       1777, 1060, 1566,  900, 1704, 1484,  520,  649, 1228, 1234, 1398,
       1561, 1117, 1097, 1297, 1057, 1088, 1350,  840,  938, 1150, 1752,
       1434, 1656,  736,  955,  794,  816, 1842,  384, 1425,  970,  860,
       1410,  780,  530, 1370,  576, 1143, 1947, 1453,  747, 1304, 2223,
        845, 1086,  462,  672, 1768,  440,  896, 1237, 1563, 1065, 1288,
        684,  612, 1013,  990, 1235,  876, 1214,  824,  680, 1588,  960,
        458,  950, 1610,  741, 1226, 1053,  641,  789,  793, 1844,  994,
       1264, 1809, 1028,  729, 1092, 1125, 1673,  728,  732, 1080, 1199,
       1362, 1078,  660, 1008,  924,  992, 1063, 1267, 1461, 1907,  928,
        864, 1734,  910, 1490, 1728,  715,  884,  969, 1710,  825, 1602,
       1200,  572,  774, 1392, 1232, 1572, 1541,  882, 1149,  644, 1617,
       1582,  720, 1064, 1606, 1202, 1151, 1052, 2216,  968,  504, 1188,
       1593,  853,  725, 1431,  855, 1726, 1360,  755, 1713, 1121, 1196,
        617,  848, 1424, 1140, 1100, 1157, 1212,  689, 1070, 1436,  686,
        798, 1248, 1498, 1010,  713, 2392,  630, 1203,  483, 1373, 1194,
       1462,  894, 1414,  996, 1694,  735,  540,  626,  948, 1845, 1020,
       1367, 1444, 1573, 1302, 1314,  975, 1604,  963, 1482,  506,  926,
       1422,  802,  740, 1095, 1385, 1152, 1240, 1560, 2121, 1160,  807,
       1468, 1575,  625,  858,  698, 1079,  768,  795, 1416, 1003,  702,
       1165, 1470, 2000,  700,  319,  861, 1896,  697,  972, 2136,  716,
       1347, 1372, 1249, 1136, 1502, 1162,  710, 1719, 1383,  844,  596,
       1056, 3206, 1358,  943, 1499, 1922, 1536, 1208, 1215,  967,  721,
       1684,  536,  958, 1478,  764, 1848, 1869,  616,  624,  940, 1142,
       1062,  888,  883, 1394, 1099, 1268,  953,  744,  608,  847,  683,
        870, 1580, 1856,  982, 1026, 1293,  939,  784, 1256,  658, 1041,
       1682,  804,  788, 1144,  961, 1260, 1310, 1141,  806, 1281, 1034,
       1276, 1340, 1344,  988,  651, 1518,  907,  901,  765,  799,  648,
       3094, 1440, 1258,  915, 1517,  930,  813, 1533,  872, 1242, 1364,
        588,  709,  560, 1375, 1277, 1626, 1488,  808,  547, 1976, 2153,
       1705, 1833, 1792, 1216,  999, 1113, 1073,  954,  264, 1269,  190,
       3200,  866, 1501,  777, 1218, 1368, 1084, 2006, 1244, 3138, 1379,
       1257, 1452,  528, 2035,  611,  707,  880, 1051, 1581, 1838, 1650,
        723,  654, 1204, 1069, 1709,  998,  993, 1374, 1389, 1163, 1122,
       1496,  846,  372, 1164, 1050, 2042, 1868, 1437,  742,  770, 1722,
       1814, 1430, 1058,  908,  600,  965, 1032, 1299, 1120,  936,  783,
       1822, 1522,  980, 1116,  978, 1156,  636, 1554, 1386,  811, 1520,
       1952, 1766,  981, 1094, 2109,  525,  776, 1486, 1629, 1138, 2077,
       1406, 1021, 1408,  738, 1477, 2046,  923, 1291, 1195, 1190,  874,
        551, 1419, 2444, 1210,  927, 1112, 1391, 1800,  360, 1473, 1643,
       1324,  270,  859,  718, 1176, 1311,  971, 1742,  941, 1698, 1584,
       1595,  868, 1153,  893, 1349, 1337, 1720, 1479, 1030, 1318, 1252,
        983, 1860,  836, 1935, 1614,  761, 1413,  956,  712,  650,  773,
       1926,  731, 1417, 1024,  849, 1442, 1649, 1568,  778, 1489, 2078,
       1454, 1516, 1067, 1559, 1127, 1390, 1273,  918, 1763, 1090, 1054,
       1039, 1148, 1002, 1638,  105,  676, 1184, 1109,  892, 2217, 1505,
       1059,  951, 2330, 1670, 1623, 1017, 1105, 1001,  546,  480, 1134,
       1104, 1272, 1316, 1126, 1181, 1753,  964, 1466,  925, 1905, 1500,
        585, 1632,  819, 1616, 1161,  828,  945,  979,  561,  696, 1330,
        817, 1098, 1428,  673, 1241,  944, 1225, 1266, 1128,  485, 1930,
       1396,  916,  822,  750, 1700, 1007, 1187,  691, 1574, 1680, 1346,
        985, 1657,  602, 1022, 1082,  810, 1504, 1220, 1132, 1565, 1338,
       1654, 1620, 1055,  800, 1306, 1475, 2524, 1992, 1193,  973,  854,
        662, 1103, 1154,  942, 1048,  727,  690, 1096, 1459, 1251, 1247,
```

```
1074, 1271,  290,  655, 1463, 1836,  803,  833,  408,  533, 1012,
1552, 1005, 1530,  974, 1567, 1006, 1042, 1298,  704,  932, 1219,
1296, 1198,  959, 1261, 1598, 1683,  818, 1600, 2396, 1624,  831,
1224,  663,  879,  815, 1630, 2158,  931, 1660,  559, 1300, 1702,
1075, 1361, 1106, 1476, 1689, 2076,  792, 2110, 1405, 1192,  746,
1986,  841, 2002, 1332,  935, 1019,  661, 1309, 1328, 1085, 6110,
1246,  771,  976, 1652, 1278, 1902, 1274, 1393, 1622, 1352,  420,
1795,  544, 1510,  911,  693, 1284, 1732, 2033,  570, 1980,  814,
 873,  757, 1108, 2633, 1571,  984, 1205,  714, 1746, 1525,  482,
1356,  862,  839, 1286, 1485, 1594,  622,  791,  708, 1223,  913,
 656, 1319, 1932,  539, 1221, 1542])
```

In [10]:
```
df.KitchenQual.unique()
```

Out[10]:
```
array(['Gd', 'TA', 'Ex', 'Fa'], dtype=object)
```

In [11]:
```
df.Street.unique()
```

Out[11]:
```
array(['Pave', 'Grvl'], dtype=object)
```

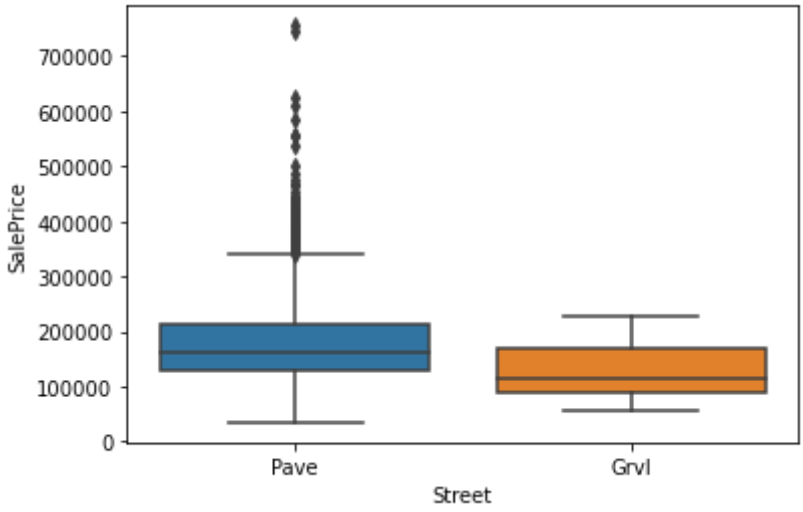
In [12]:
```
df.OverallCond.unique()
```

Out[12]:
```
array([5, 8, 6, 7, 4, 2, 3, 9, 1])
```

In [13]:
```
final = pd.get_dummies(df, drop_first = True)
```

In [14]:
```
sns.boxplot(x=df.Street ,y=df.SalePrice , data=final)
```

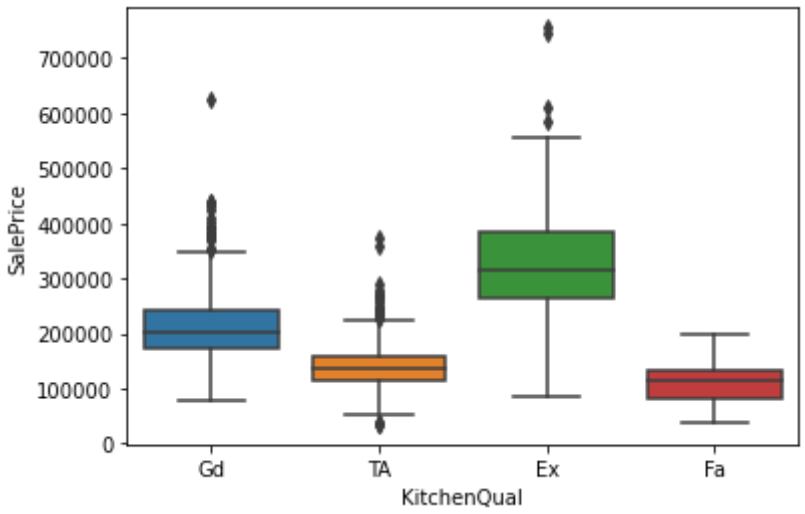
Out[14]:
```
<AxesSubplot:xlabel='Street', ylabel='SalePrice'>
```




Looking at the boxplots, we could argue tat the Price of a house is higher when the Street is paved.

In [15]:
```
sns.boxplot(x=df.KitchenQual ,y=df.SalePrice , data=final)
```

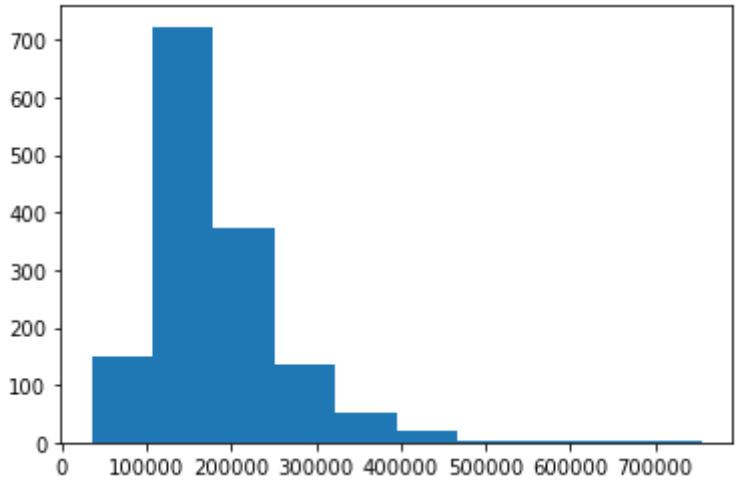
Out[15]:
```
<AxesSubplot:xlabel='KitchenQual', ylabel='SalePrice'>
```




Looking at the boxplots, we could argue that the Price of a house is higher for houses with Good/Excellent Kitchen Quality.

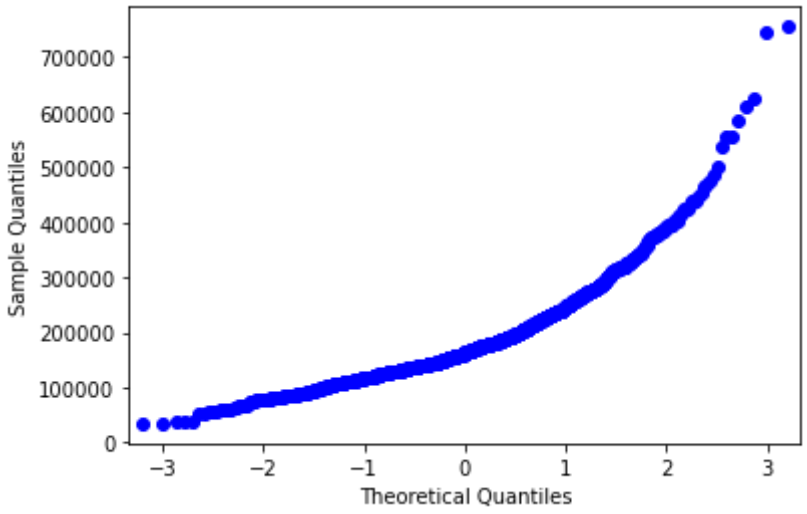
In [16]:
```
plt.hist(df.SalePrice)
```

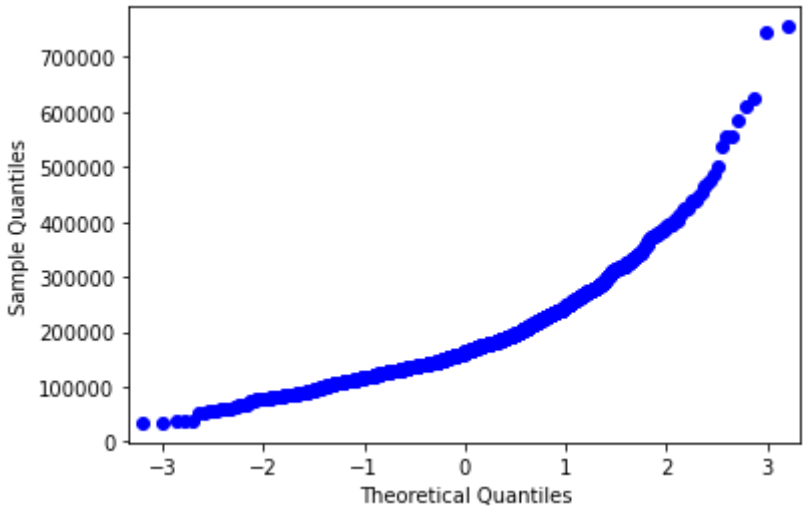
Out[16]:
```
(array([148., 723., 373., 135.,  51.,  19.,   4.,   3.,   2.,   2.]),
 array([ 34900., 106910., 178920., 250930., 322940., 394950., 466960.,
        538970., 610980., 682990., 755000.]),
 <BarContainer object of 10 artists>)
```



In [17]:
```
sm.qqplot(df.SalePrice)
```

Out[17]:

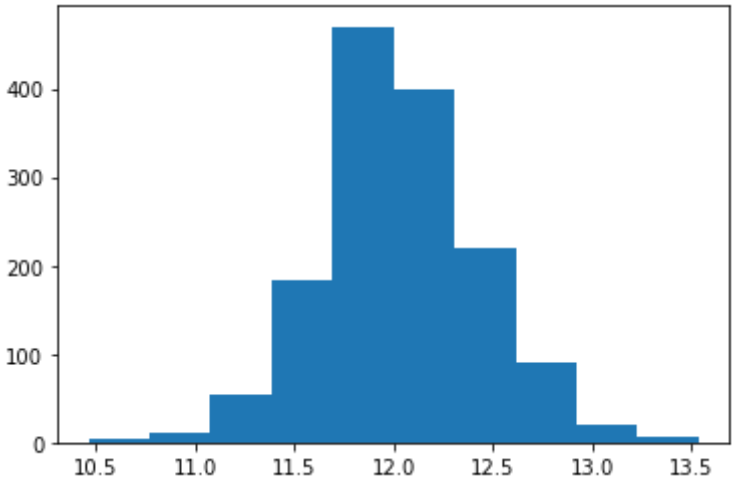



Looking at the plots, SalePrice has an F-distribution. I believe that a log tranformation for SalePrice would help us achieve a normal dsitribution.

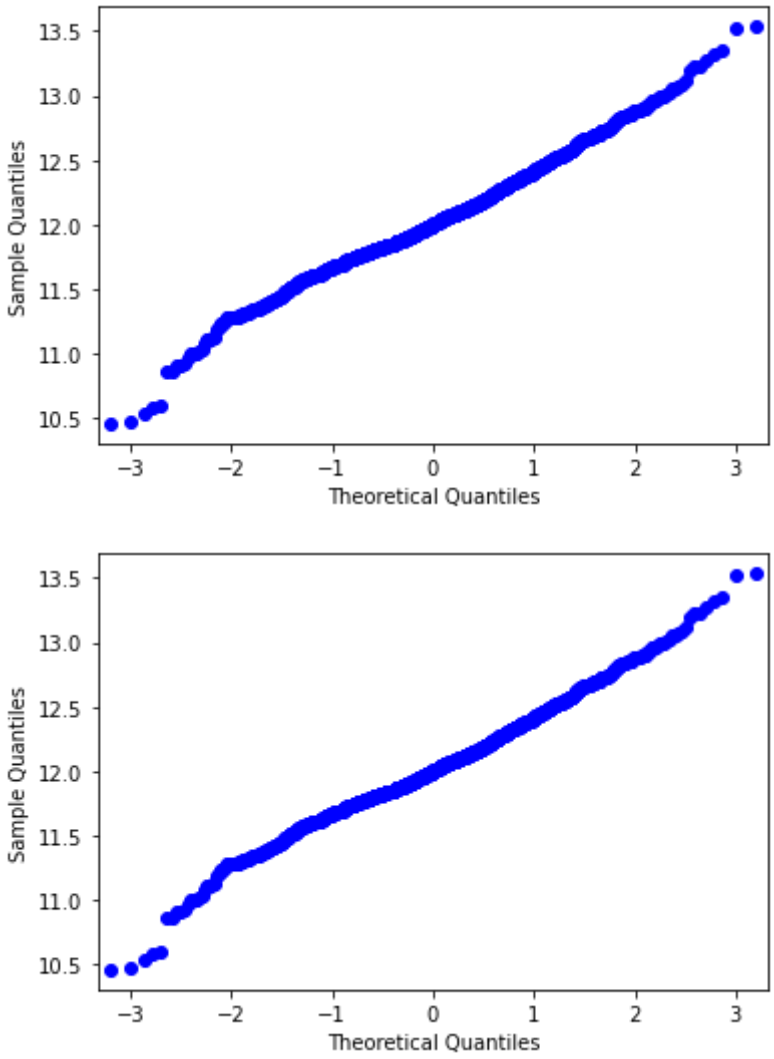
In [18]:
```
plt.hist(np.log(df.SalePrice))
```

Out[18]:
```
(array([  5.,  12.,  54., 184., 470., 400., 220.,  90.,  19.,   6.]),
 array([10.46024211, 10.7676652 , 11.07508829, 11.38251138, 11.68993448,
        11.99735757, 12.30478066, 12.61220375, 12.91962684, 13.22704994,
        13.53447303]),
 <BarContainer object of 10 artists>)
```



In [19]:
```
sm.qqplot(np.log(df.SalePrice))
```

Out[19]:




In [20]:
```
final = final.dropna
```

In [23]:
```
X = final()[["GrLivArea","LotArea","YearBuilt","LotFrontage","TotalBsmtSF",'
           "OverallCond"]]
X = sm.add_constant(X)
Y = np.log(final().SalePrice)
model_1 = sm.OLS(Y,X).fit()
model_1.summary()
```

Out[23]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | SalePrice | **R-squared:** | 0.821 |
| **Model:** | OLS | **Adj. R-squared:** | 0.819 |
| **Method:** | Least Squares | **F-statistic:** | 494.3 |
| **Date:** | Thu, 04 Nov 2021 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 17:27:39 | **Log-Likelihood:** | 381.22 |
| **No. Observations:** | 1201 | **AIC:** | -738.4 |
| **Df Residuals:** | 1189 | **BIC:** | -677.4 |
| **Df Model:** | 11 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 1.9697 | 0.490 | 4.019 | 0.000 | 1.008 | 2.931 |
| **GrLivArea** | 0.0003 | 1.27e-05 | 23.696 | 0.000 | 0.000 | 0.000 |
| **LotArea** | 3.381e-06 | 7.35e-07 | 4.602 | 0.000 | 1.94e-06 | 4.82e-06 |
| **YearBuilt** | 0.0045 | 0.000 | 18.694 | 0.000 | 0.004 | 0.005 |
| **LotFrontage** | -5.98e-05 | 0.000 | -0.238 | 0.812 | -0.001 | 0.000 |
| **TotalBsmtSF** | 0.0001 | 1.51e-05 | 9.001 | 0.000 | 0.000 | 0.000 |
| **GarageCars** | 0.1070 | 0.009 | 11.911 | 0.000 | 0.089 | 0.125 |
| **Street_Pave** | 0.1942 | 0.080 | 2.417 | 0.016 | 0.037 | 0.352 |
| **KitchenQual_Fa** | -0.2622 | 0.040 | -6.504 | 0.000 | -0.341 | -0.183 |
| **KitchenQual_TA** | -0.2323 | 0.024 | -9.715 | 0.000 | -0.279 | -0.185 |
| **KitchenQual_Gd** | -0.1391 | 0.021 | -6.545 | 0.000 | -0.181 | -0.097 |
| **OverallCond** | 0.0709 | 0.005 | 13.313 | 0.000 | 0.060 | 0.081 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 1004.053 | **Durbin-Watson:** | 2.031 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 77705.131 |
| **Skew:** | -3.343 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 41.834 | **Cond. No.** | 1.24e+06 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.24e+06. This might indicate that there are strong multicollinearity or other numerical problems.

All our coefficients, except LotFrontage, are statistically significant at a 5% level of significance. Our $R^2$ is quite good, 0.821, which means that our explanatory variables explain 82.1% of the data. All the explanatory variables are positevly correlated with SalePrice except for the kitchen quality dummies.

In [24]:
```
X["LotFrontage:LotArea"] = final()["LotFrontage"]*final()["LotArea"]
X["GrLivArea**2"] = X["GrLivArea"]**2
X = sm.add_constant(X)
Y = np.log(final()['SalePrice'])
model_3 = sm.OLS(Y,X).fit()
model_3.summary()
```

Out[24]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | SalePrice | **R-squared:** | 0.866 |
| **Model:** | OLS | **Adj. R-squared:** | 0.864 |
| **Method:** | Least Squares | **F-statistic:** | 588.6 |
| **Date:** | Thu, 04 Nov 2021 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 17:28:20 | **Log-Likelihood:** | 555.20 |
| **No. Observations:** | 1201 | **AIC:** | -1082. |
| **Df Residuals:** | 1187 | **BIC:** | -1011. |
| **Df Model:** | 13 | | |
| **Covariance Type:** | nonrobust | | |

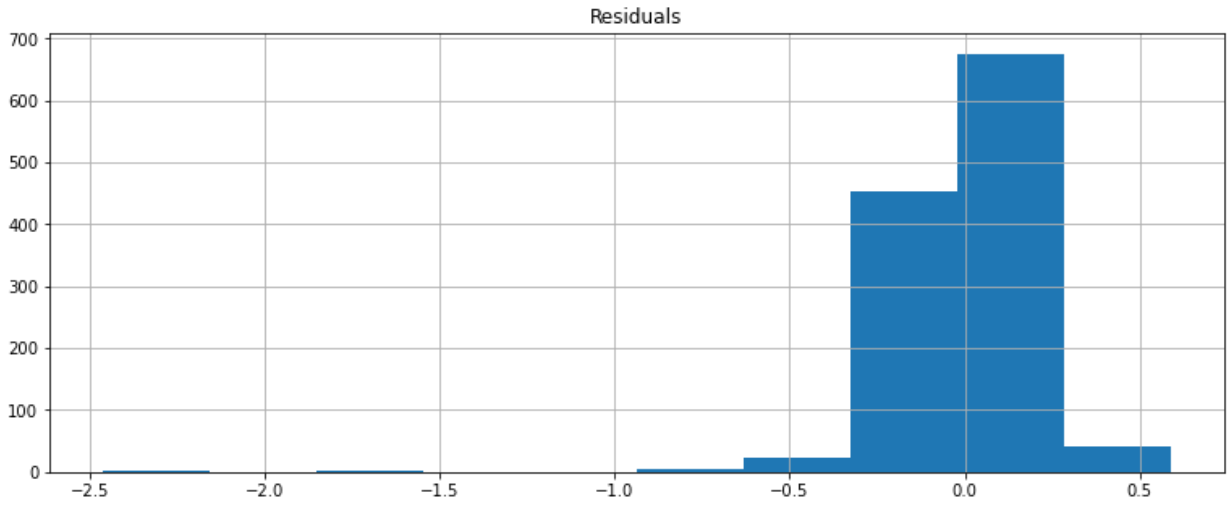
| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 1.4090 | 0.426 | 3.309 | 0.001 | 0.574 | 2.244 |
| **GrLivArea** | 0.0007 | 3.2e-05 | 21.067 | 0.000 | 0.001 | 0.001 |
| **LotArea** | 1.592e-05 | 1.46e-06 | 10.942 | 0.000 | 1.31e-05 | 1.88e-05 |
| **YearBuilt** | 0.0045 | 0.000 | 21.789 | 0.000 | 0.004 | 0.005 |
| **LotFrontage** | 0.0013 | 0.000 | 5.392 | 0.000 | 0.001 | 0.002 |
| **TotalBsmtSF** | 0.0002 | 1.32e-05 | 13.476 | 0.000 | 0.000 | 0.000 |
| **GarageCars** | 0.0728 | 0.008 | 9.149 | 0.000 | 0.057 | 0.088 |
| **Street_Pave** | 0.1984 | 0.070 | 2.839 | 0.005 | 0.061 | 0.335 |
| **KitchenQual_Fa** | -0.2540 | 0.035 | -7.274 | 0.000 | -0.322 | -0.185 |
| **KitchenQual_TA** | -0.2357 | 0.021 | -11.367 | 0.000 | -0.276 | -0.195 |
| **KitchenQual_Gd** | -0.1601 | 0.019 | -8.637 | 0.000 | -0.197 | -0.124 |
| **OverallCond** | 0.0757 | 0.005 | 16.347 | 0.000 | 0.067 | 0.085 |
| **LotFrontage:LotArea** | -9.802e-08 | 1.07e-08 | -9.167 | 0.000 | -1.19e-07 | -7.7e-08 |
| **GrLivArea**2** | -9.867e-08 | 7.91e-09 | -12.480 | 0.000 | -1.14e-07 | -8.32e-08 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 247.707 | **Durbin-Watson:** | 2.025 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 1356.751 |
| **Skew:** | -0.838 | **Prob(JB):** | 2.43e-295 |
| **Kurtosis:** | 7.930 | **Cond. No.** | 3.33e+08 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.33e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

```
I chose LotArea and LotFrontage as interaction variables becaus ethey seem
to be relatively highly correlated.
I chose GrlivArea with a 2nd power because it is the variable with highest
correlation to SalePrice, therefore, I checked the marginal effect of
GrLivArea on SalePrice.
```

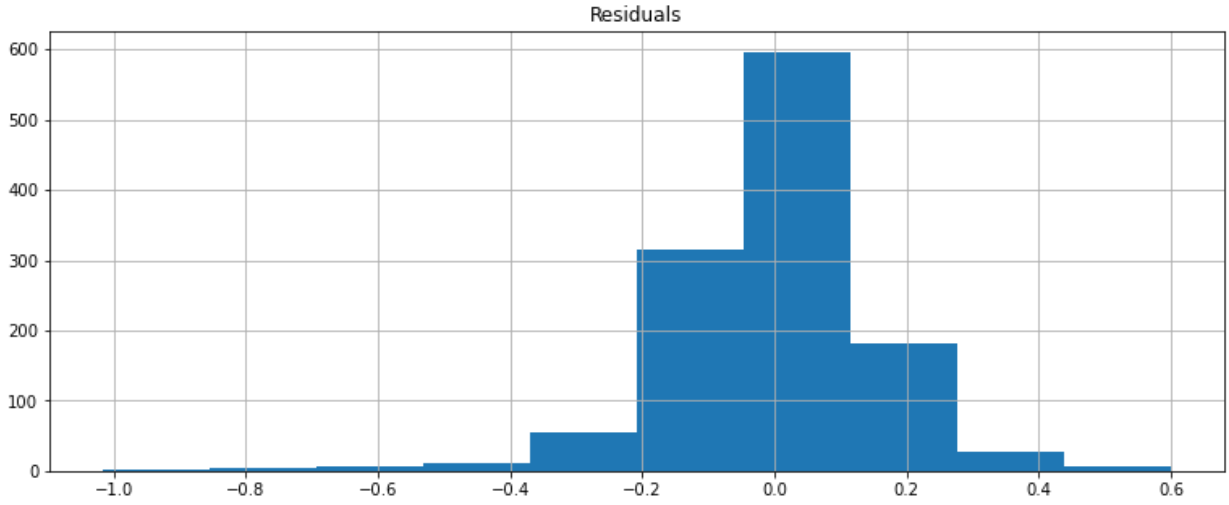
In [28]:
```
plt.figure(figsize=(13,5))
plt.title("Residuals")
plt.hist(model_1.resid)
plt.grid()
```




In [29]:
```
stats.jarque_bera(model_1.resid)
```

Out[29]:
```
Jarque_beraResult(statistic=77705.1310530387, pvalue=0.0)
```

In [30]:
```
plt.figure(figsize=(13,5))
plt.title("Residuals")
plt.hist(model_3.resid)
plt.grid()
```




In [31]:
```
stats.jarque_bera(model_3.resid)
```

Out[31]:
```
Jarque_beraResult(statistic=1356.7507304024825, pvalue=0.0)
```

The residuals in both models are NOT normally distributed.

In [ ]: