Ames Housing Sale Price Analysis

Krist Zografi

Executive Summary

This project uses the Ames Housing dataset, which contains information about home sales in Ames, Iowa between 2006 and 2010. We going to use this dataset in order to understand some important data analysis like:
1.How different factors seem to influence home sales.
2.Explore the differences between subsets.
3.Explore Correlation.
4.Create a new column based on the values of 2 or more columns in a dataset.

The Data

Using the data from the Ames Housing dataset, we will start by exploring the contents, rows, columns,type of each column ect.

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd

file_path = "https://github.com/learn-co-curriculum/da-phase1-project-enterprise/
df = pd.read_csv(file_path, index_col=0)
```

Lets see the first five rows of this dataframe

In [2]:
```
df.head()
```

Out[2]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
Id									
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub

5 rows × 80 columns

In [3]:
```
df.shape
```
Out[3]:
```
(1460, 80)
```

As we can see our dataframe is build from 1460 rows and 80 columns

We will use the below code to get a description of the dataframe we are working with.

In [4]:
```
df.describe()
```
Out[4]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRem
count	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.0
mean	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.8
std	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.6
min	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.0
25%	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.0
50%	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.0
75%	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.0
max	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.0

8 rows × 37 columns

An explanation of what represents each row:
count - The number of not-empty values.
mean - The average (mean) value.
std - The standard deviation.
min - the minimum value.
25% - The 25% percentile. *50% - The 50% percentile*. 75% - The 75% percentile*.
max - the maximum value.

Using the code below we can see the names from each column

In [5]:
```
list(df.columns.values)
```

Out[5]:
```
['MSSubClass',
 'MSZoning',
 'LotFrontage',
 'LotArea',
 'Street',
 'Alley',
 'LotShape',
 'LandContour',
 'Utilities',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
 'BldgType',
 'HouseStyle',
 'OverallQual',
 'OverallCond',
 'YearBuilt',
 'YearRemodAdd',
 'RoofStyle',
 'RoofMatl',
 'Exterior1st',
 'Exterior2nd',
 'MasVnrType',
 'MasVnrArea',
 'ExterQual',
 'ExterCond',
 'Foundation',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinSF1',
 'BsmtFinType2',
 'BsmtFinSF2',
 'BsmtUnfSF',
 'TotalBsmtSF',
 'Heating',
 'HeatingQC',
 'CentralAir',
 'Electrical',
 '1stFlrSF',
 '2ndFlrSF',
 'LowQualFinSF',
 'GrLivArea',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',
 'BedroomAbvGr',
 'KitchenAbvGr',
 'KitchenQual',
 'TotRmsAbvGrd',
```

```
'Functional',
'Fireplaces',
'FireplaceQu',
'GarageType',
'GarageYrBlt',
'GarageFinish',
'GarageCars',
'GarageArea',
'GarageQual',
'GarageCond',
'PavedDrive',
'WoodDeckSF',
'OpenPorchSF',
'EnclosedPorch',
'3SsnPorch',
'ScreenPorch',
'PoolArea',
'PoolQC',
'Fence',
'MiscFeature',
'MiscVal',
'MoSold',
'YrSold',
'SaleType',
'SaleCondition',
'SalePrice']
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1460 entries, 1 to 1460
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MSSubClass     1460 non-null   int64  
 1   MSZoning       1460 non-null   object 
 2   LotFrontage    1201 non-null   float64
 3   LotArea        1460 non-null   int64  
 4   Street         1460 non-null   object 
 5   Alley          91 non-null     object 
 6   LotShape       1460 non-null   object 
 7   LandContour    1460 non-null   object 
 8   Utilities      1460 non-null   object 
 9   LotConfig      1460 non-null   object 
 10  LandSlope      1460 non-null   object 
 11  Neighborhood   1460 non-null   object 
 12  Condition1     1460 non-null   object 
 13  Condition2     1460 non-null   object 
 14  BldgType       1460 non-null   object 
 15  HouseStyle     1460 non-null   object 
 16  OverallQual    1460 non-null   int64  
 17  OverallCond    1460 non-null   int64  
 18  YearBuilt      1460 non-null   int64  
 19  YearRemodAdd   1460 non-null   int64  
 20  RoofStyle      1460 non-null   object 
 21  RoofMatl       1460 non-null   object 
 22  Exterior1st    1460 non-null   object 
 23  Exterior2nd    1460 non-null   object 
 24  MasVnrType     1452 non-null   object 
 25  MasVnrArea     1452 non-null   float64
 26  ExterQual      1460 non-null   object 
 27  ExterCond      1460 non-null   object 
 28  Foundation     1460 non-null   object 
 29  BsmtQual       1423 non-null   object 
 30  BsmtCond       1423 non-null   object 
 31  BsmtExposure   1422 non-null   object 
 32  BsmtFinType1   1423 non-null   object 
 33  BsmtFinSF1     1460 non-null   int64  
 34  BsmtFinType2   1422 non-null   object 
 35  BsmtFinSF2     1460 non-null   int64  
 36  BsmtUnfSF      1460 non-null   int64  
 37  TotalBsmtSF    1460 non-null   int64  
 38  Heating        1460 non-null   object 
 39  HeatingQC      1460 non-null   object 
 40  CentralAir     1460 non-null   object 
 41  Electrical     1459 non-null   object 
 42  1stFlrSF       1460 non-null   int64  
 43  2ndFlrSF       1460 non-null   int64  
 44  LowQualFinSF   1460 non-null   int64  
 45  GrLivArea      1460 non-null   int64  
 46  BsmtFullBath   1460 non-null   int64  
 47  BsmtHalfBath   1460 non-null   int64  
 48  FullBath       1460 non-null   int64  
```

```
 49  HalfBath       1460 non-null   int64
 50  BedroomAbvGr   1460 non-null   int64
 51  KitchenAbvGr   1460 non-null   int64
 52  KitchenQual    1460 non-null   object
 53  TotRmsAbvGrd   1460 non-null   int64
 54  Functional     1460 non-null   object
 55  Fireplaces     1460 non-null   int64
 56  FireplaceQu    770 non-null    object
 57  GarageType     1379 non-null   object
 58  GarageYrBlt    1379 non-null   float64
 59  GarageFinish   1379 non-null   object
 60  GarageCars     1460 non-null   int64
 61  GarageArea     1460 non-null   int64
 62  GarageQual     1379 non-null   object
 63  GarageCond     1379 non-null   object
 64  PavedDrive     1460 non-null   object
 65  WoodDeckSF     1460 non-null   int64
 66  OpenPorchSF    1460 non-null   int64
 67  EnclosedPorch  1460 non-null   int64
 68  3SsnPorch      1460 non-null   int64
 69  ScreenPorch    1460 non-null   int64
 70  PoolArea       1460 non-null   int64
 71  PoolQC         7 non-null      object
 72  Fence          281 non-null    object
 73  MiscFeature    54 non-null     object
 74  MiscVal        1460 non-null   int64
 75  MoSold         1460 non-null   int64
 76  YrSold         1460 non-null   int64
 77  SaleType       1460 non-null   object
 78  SaleCondition  1460 non-null   object
 79  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(34), object(43)
memory usage: 923.9+ KB
```

As we can see from the above table, we have the names of the 80 columns and their type. Also we can notice that some of them are missing datas, for example LotFrontage which has 1201 from 1460, PoolQc 7 from 1460, Fence 281 from 1460 and so on.

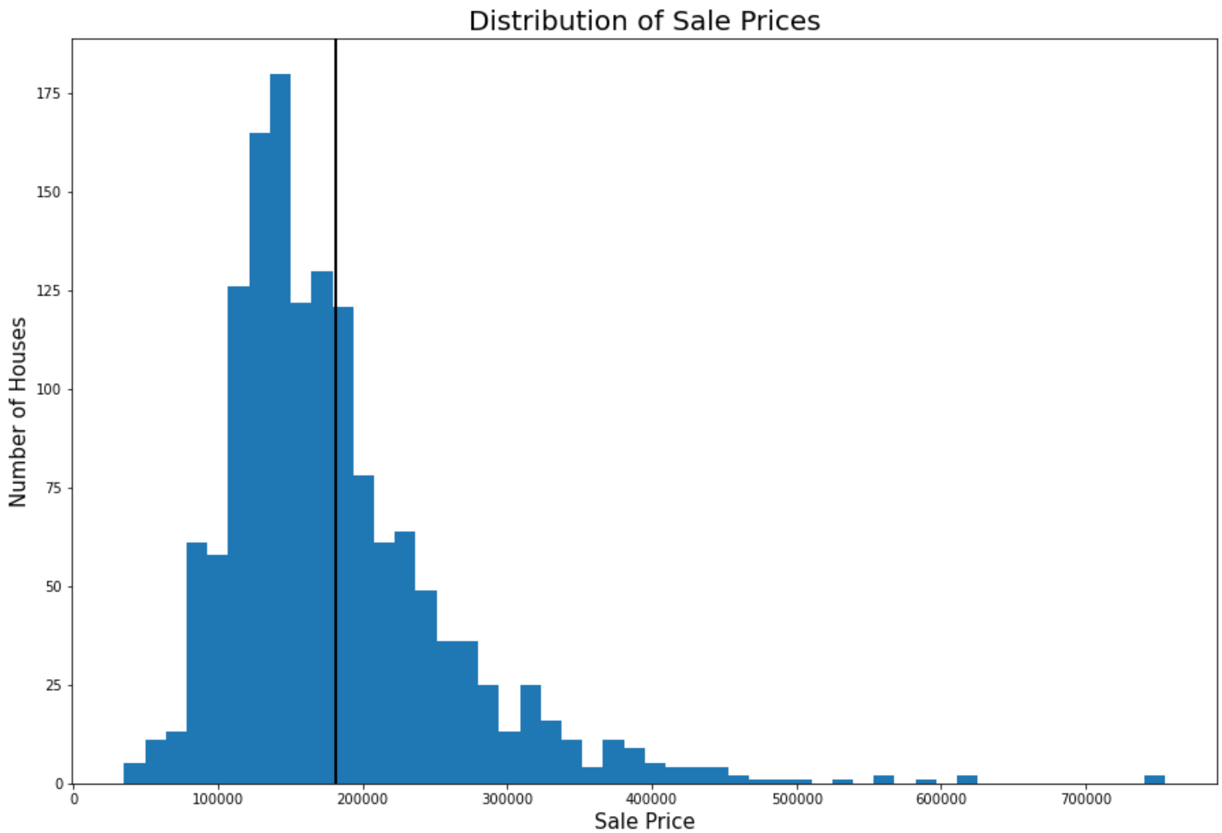
Distribution of `SalePrice`

Through the below code we will create a graphic vizualization to understand better the distribution of Sales Prices. Also we will add the mean.

In [7]:
```
fig, ax = plt.subplots(figsize = (15, 10))
ax.hist( df['SalePrice'], bins = 50)
ax.axvline(df['SalePrice'].mean(), color='k', linewidth=2)

ax.set_title('Distribution of Sale Prices', fontsize = 20)
ax.set_ylabel('Number of Houses', fontsize = 15)
ax.set_xlabel('Sale Price', fontsize = 15)
```

Out[7]:
```
Text(0.5, 0, 'Sale Price')
```



In [8]:
```
df.SalePrice.describe()
```

Out[8]:
```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

From the above distribution I can see that there are several outliners which starts after the price of 500'000, but the most distant one is above the value of 700'000. The graphic shows mean as a vertical line in the graph but for specific value we are going to use the above table. From the graph we can say that this is a negative right-skewed distribution.

Differences between Subsets

On this part of the project we will split the data in two or more subsets, and plot the SalePrice distribiution for each subset. I found it quite handy to use the OverallCond for this pourpose. I started by finding which elements it involved.

In [9]:
```
df['OverallCond'].value_counts()
```

Out[9]:
```
5    821
6    252
7    205
8     72
4     57
3     25
9     22
2      5
1      1
Name: OverallCond, dtype: int64
```

After that, I created a category mapping so I could use it to divide the different values into three main categories: 'Below', 'Average', 'Above'.

In [10]:
```
category_mapping = {
    1: "Below",
    2: "Below",
    3: "Below",
    4: "Below",
    5: "Average",
    6: "Above",
    7: "Above",
    8: "Above",
    9: "Above",
}
```

Using the three main categories, I created a new column, so we can use it later for the grouping of the SalePrice

In [11]:
```
df['New_Category'] = df['OverallCond'].map(category_mapping)
df['New_Category']
```

Out[11]:
```
Id
1       Average
2         Above
3       Average
4       Average
5       Average
         ...   
1456    Average
1457      Above
1458      Above
1459      Above
1460      Above
Name: New_Category, Length: 1460, dtype: object
```

Using the new column ('New_Category'), I grouped the SalePrice values into the three categories.

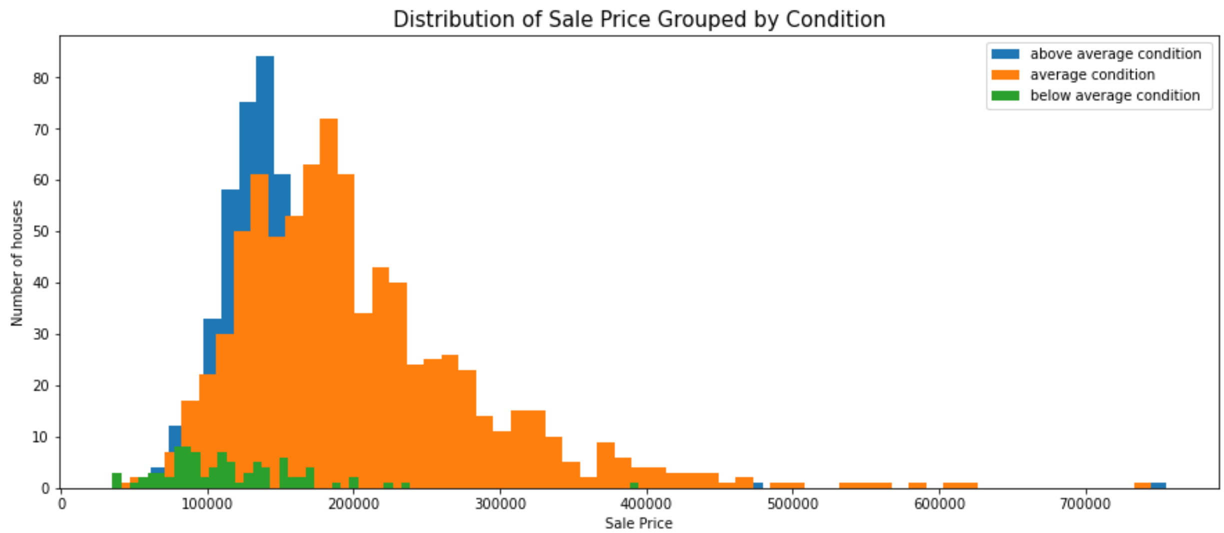
In [12]:
```
grouped = df.groupby(df['New_Category'])['SalePrice']
list(grouped)
```

Out[12]:
```
[('Above',
  Id
  2       181500
  8       200000
  10      118000
  13      144000
  16      132000
           ...
  1450     92000
  1457    210000
  1458    266500
  1459    142125
  1460    147500
  Name: SalePrice, Length: 551, dtype: int64),
 ('Average',
  Id
  1       208500
  3       223500
  4       140000
  5       250000
  6       143000
           ...
  1452    287090
  1453    145000
  1454     84500
  1455    185000
  1456    175000
  Name: SalePrice, Length: 821, dtype: int64),
 ('Below',
  Id
  31       40000
  70      225000
  89       85000
  92       98600
  105     169500
           ...
  1346    108500
  1363    104900
  1381     58500
  1399    138000
  1405    105000
  Name: SalePrice, Length: 88, dtype: int64)]
```

The list above shows the elements that are involved in each category.
All is left to do, is create a histogram showing the relationship each group of values has with each other.

In [13]:
```
grouped.hist(bins = 60, figsize = (15, 6), grid = False)
plt.title("Distribution of Sale Price Grouped by Condition", fontsize = 15)
plt.xlabel('Sale Price', fontsize = 10)
plt.ylabel('Number of houses', fontsize = 10)
plt.legend(['above average condition ', 'average condition', 'below average condi
```

Out[13]: <matplotlib.legend.Legend at 0x7fb62ec7c9a0>



By what i can see, all the groups have a right skew distribution. Also no matter the condition of the house the average price people spend to buy a new house is between $100k and 250$k.

I apologize for the graphic but i couldnt find a way to make it better.

Correlation between `OverallQual` and `SalePrice`

On this part of the project we are going to investigate the corrolation between two columns. Because we had determined that we would use "SalePrice" for this, the only thing left for us was to determin which one was a good column which has a notable correlation with our main column. For this we are going to use the above code to determinat the second column.

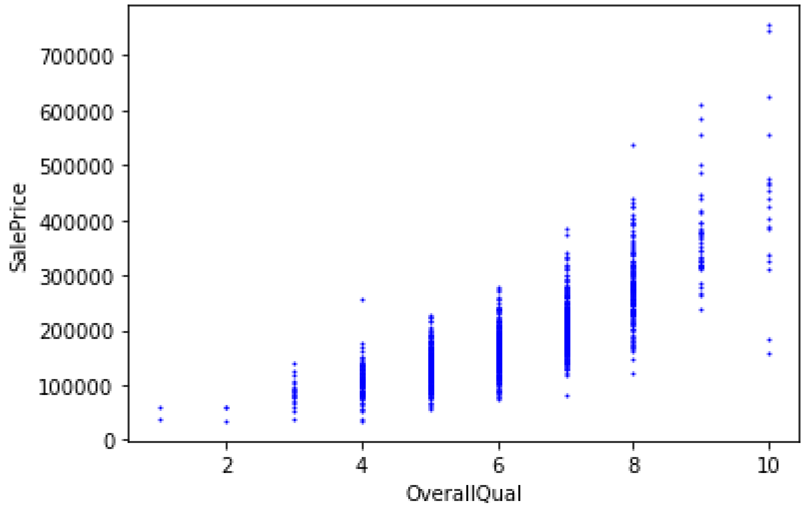
In [14]:
```python
df.corr()['SalePrice']
```

Out[14]:
```
MSSubClass      -0.084284
LotFrontage      0.351799
LotArea          0.263843
OverallQual      0.790982
OverallCond     -0.077856
YearBuilt        0.522897
YearRemodAdd     0.507101
MasVnrArea       0.477493
BsmtFinSF1       0.386420
BsmtFinSF2      -0.011378
BsmtUnfSF        0.214479
TotalBsmtSF      0.613581
1stFlrSF         0.605852
2ndFlrSF         0.319334
LowQualFinSF    -0.025606
GrLivArea        0.708624
BsmtFullBath     0.227122
BsmtHalfBath    -0.016844
FullBath         0.560664
HalfBath         0.284108
BedroomAbvGr     0.168213
KitchenAbvGr    -0.135907
TotRmsAbvGrd     0.533723
Fireplaces       0.466929
GarageYrBlt      0.486362
GarageCars       0.640409
GarageArea       0.623431
WoodDeckSF       0.324413
OpenPorchSF      0.315856
EnclosedPorch   -0.128578
3SsnPorch        0.044584
ScreenPorch      0.111447
PoolArea         0.092404
MiscVal         -0.021190
MoSold           0.046432
YrSold          -0.028923
SalePrice        1.000000
Name: SalePrice, dtype: float64
```

By what we know every value that is above 0.5 has a notable corrolation, and every value that is between 0.7 and 0.9 has a high correlation. The column I will choose to explore the correlation with the Sale Price, in this excercise is **OverallQual**.

In [15]:
```
df.plot.scatter( x = 'OverallQual', y = 'SalePrice', s = .9, c = 'blue' )
```

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



We can see that this is a strong positive corrolation between the two columns. The strength of the corrolation is stronger in the middle of the graphs and weaker on the sides.

Age, Total Baths, PorchDesk area

As you can notice by the title we are going to create 3 new column that will include:
1.Age, will represent the age of the house
2.Total Baths, will represent the total amount of baths included in the house.
3.PorchDesk area, will represent the total space area in square feet of the porch and deck in the house.

Age for each house

In [16]:
```
# Age of the house, we can calculate the age for the house by substracting the ye
#build from the year it was sold .First we need to see each column, for the index
#I used the table we got from df.info()
df.iloc[:, [18, 76]][:5]
```

Out[16]:

	YearBuilt	YrSold
Id		
1	2003	2008
2	1976	2007
3	2001	2008
4	1915	2006
5	2000	2008

In [17]:
```
#Now we need to create the new column 'Age' which will get the substraction of th
df['Age'] = df.apply(lambda x: x['YrSold'] - x['YearBuilt'] , axis = 1)
df['Age']
```

Out[17]:
```
Id
1        5
2       31
3        7
4       91
5        8
        ..
1456     8
1457    32
1458    69
1459    60
1460    43
Name: Age, Length: 1460, dtype: int64
```

As we can see above for each building we have the age of each house in years

Total Baths in the house

In [18]:
```
# First we need to see the columns with indicate the baths in different areas of
df.iloc[:, 46:49][:5]
```

Out[18]:

	BsmtFullBath	BsmtHalfBath	FullBath
Id			
1	1	0	2
2	0	1	2
3	1	0	2
4	1	0	1
5	1	0	2

In [19]:
```
#The code below will give us the total number of baths in a house
df['TotalBaths'] = df.apply(lambda x: x['BsmtFullBath'] + x['BsmtHalfBath'] + x['
df['TotalBaths'][:5]
```

Out[19]:
```
Id
1    4
2    3
3    4
4    2
5    4
Name: TotalBaths, dtype: int64
```

If we compare the first row of the table where we have all the columns of the baths with the new column, we can check that the total number matches.

Porch and Deck Area

In [20]:
```
# As we did for the previous excercises we will start by showing a table of all t
df.iloc[:, 65:69][:5]
```

Out[20]:

	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch
Id				
1	0	61	0	0
2	298	0	0	0
3	0	42	0	0
4	0	35	272	0
5	192	84	0	0

In [21]:
```
# And now we find the total area by added each column to the new one which we wil
df['PorchDeckSF'] = df.apply(lambda x: x['WoodDeckSF'] + x['OpenPorchSF'] + x['Er
df['PorchDeckSF'][:5]
```

Out[21]:
```
Id
1     61
2    298
3     42
4    307
5    276
Name: PorchDeckSF, dtype: int64
```

Scatterplot between "Age" vs "SalePrice"

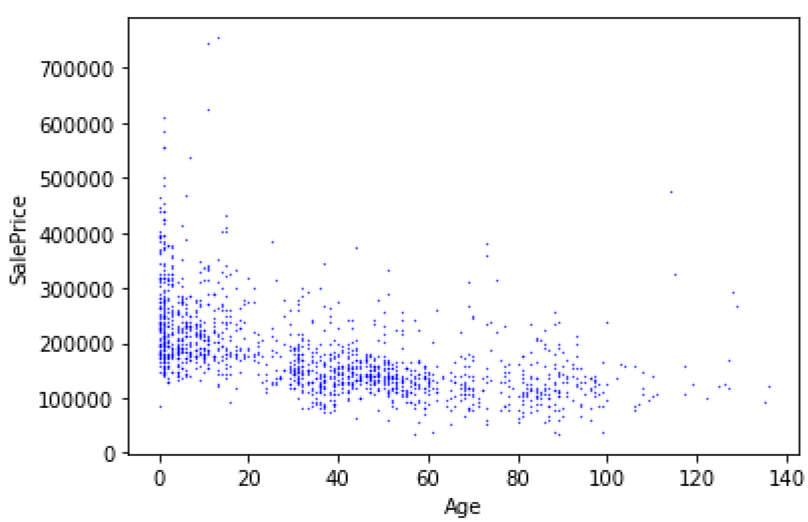
In [22]:
```
#Before we start lets take a look at the two columns we are going to use
df.iloc[:5, [79, 80]]
```

Out[22]:

Id	SalePrice	New_Category
1	208500	Average
2	181500	Above
3	223500	Average
4	140000	Average
5	250000	Average

In [23]:
```
#Thi code lets us create a plot
df.plot.scatter( x = 'Age', y = 'SalePrice', s = .09, c = 'blue' )
```

Out[23]:
```
<AxesSubplot:xlabel='Age', ylabel='SalePrice'>
```



By what we can see, the sales price gets lower with the increase of the age. Its kind of hard to distinguish but this looks like a strong negative corrolation. Also we can say that the age of the house matters , the newer the house the higher the price.