In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
import seaborn as sns
# можно и так: import seaborn as sb
from scipy.stats import norm
from scipy import stats
from pandas import DataFrame
%matplotlib inline
from google.colab import drive

from plotly.offline import init_notebook_mode, iplot
import plotly.figure_factory as ff
import cufflinks
cufflinks.go_offline()
cufflinks.set_config_file(world_readable=True, theme='pearl')
import plotly.graph_objs as go
```

In [ ]:

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:

```
pd.set_option('display.max_columns', 100)
df = pd.read_csv('/content/drive/MyDrive/data_python/AGU_rules/house_train.csv')
df.drop('Id', axis=1, inplace=True)
df.head()
```

Out[ ]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neig
0	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	
1	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	
2	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	
3	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	
4	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	

In [ ]:

```
df.info(memory_usage='deep')
```

In [ ]:

```
df.describe().T
```

In [ ]:

```
df['SalePrice'].describe()
```

In [ ]:

```
print(df.isna().sum())
```

```
MSSubClass          0
MSZoning            0
LotFrontage       259
```

```
LotArea            0
Street             0
                 ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 80, dtype: int64
```

In [ ]:

```
na_number=(df.isna().sum())
print(na_number)
```

```
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
Street             0
                 ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 80, dtype: int64
```

In [ ]:

```
print(df.columns)
```

In [ ]:

```
print(df.duplicated().sum())
```

```
0
```

In [ ]:

```
duplicated_number=df.duplicated().sum()
print(duplicated_number)
```

```
0
```

In [ ]:

```
duplicated_number=df.duplicated().sum()
print(duplicated_number)
```

```
0
```

In [ ]:

```
na_count = df.isnull().sum().sort_values(ascending=False)
# Вычисляем, сколько пропущенных значений в параметрах
na_rate = na_count / len(df)
# Вычисляем частоту или вероятность, с которой пропущенное значение встречается в каждом параметре. Если вероятность большая (>0.5), столбцы-параметры можно смело удалять).
# формируем массив для печати
na_data = pd.concat([na_count, na_rate],axis=1,keys=['count','ratio'])
print(na_data)
```

```
               count     ratio
PoolQC          1453  0.995205
MiscFeature     1406  0.963014
Alley           1369  0.937671
Fence           1179  0.807534
FireplaceQu      690  0.472603
...              ...       ...
CentralAir         0  0.000000
SaleCondition      0  0.000000
```

```
Heating             0  0.000000
TotalBsmtSF         0  0.000000
MSSubClass          0  0.000000

[80 rows x 2 columns]
```

In [ ]:

```
df.shape
```

Out[ ]:

```
(1460, 80)
```

In [ ]:

```
df_new=df.drop(['PoolQC', 'MiscFeature', 'Alley'], axis=1)
dq=df_new.isna().sum()
print(dq)
dq.shape
```

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ...
SaleType           0
SaleCondition      0
SalePrice          0
HouseAge           0
AgeGrp             1
Length: 80, dtype: int64
```

Out[ ]:

```
(80,)
```

In [ ]:

```
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
  print(dq)
```

```
MSSubClass           0
MSZoning             0
LotFrontage        259
LotArea              0
Street               0
LotShape             0
LandContour          0
Utilities            0
LotConfig            0
LandSlope            0
Neighborhood         0
Condition1           0
Condition2           0
BldgType             0
HouseStyle           0
OverallQual          0
OverallCond          0
YearBuilt            0
YearRemodAdd         0
RoofStyle            0
RoofMatl             0
Exterior1st          0
Exterior2nd          0
MasVnrType           8
MasVnrArea           8
ExterQual            0
ExterCond            0
```

```
ExterCond          0
Foundation         0
BsmtQual          37
BsmtCond          37
BsmtExposure      38
BsmtFinType1      37
BsmtFinSF1         0
BsmtFinType2      38
BsmtFinSF2         0
BsmtUnfSF          0
TotalBsmtSF        0
Heating            0
HeatingQC          0
CentralAir         0
Electrical         1
1stFlrSF           0
2ndFlrSF           0
LowQualFinSF       0
GrLivArea          0
BsmtFullBath       0
BsmtHalfBath       0
FullBath           0
HalfBath           0
BedroomAbvGr       0
KitchenAbvGr       0
KitchenQual        0
TotRmsAbvGrd       0
Functional         0
Fireplaces         0
FireplaceQu      690
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageCars         0
GarageArea         0
GarageQual        81
GarageCond        81
PavedDrive         0
WoodDeckSF         0
OpenPorchSF        0
EnclosedPorch      0
3SsnPorch          0
ScreenPorch        0
PoolArea           0
Fence           1179
MiscVal            0
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
dtype: int64
```

In [ ]:

```
df_new = df_new.drop(['GarageQual', 'GarageCond', 'Fence'], axis=1)
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
  print(df_new.isna().sum())
```

```
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
Street             0
LotShape           0
LandContour        0
Utilities          0
LotConfig          0
LandSlope          0
Neighborhood       0
```

```
Neighborhood      0
Condition1        0
Condition2        0
BldgType          0
HouseStyle        0
OverallQual       0
OverallCond       0
YearBuilt         0
YearRemodAdd      0
RoofStyle         0
RoofMatl          0
Exterior1st       0
Exterior2nd       0
MasVnrType        8
MasVnrArea        8
ExterQual         0
ExterCond         0
Foundation        0
BsmtQual         37
BsmtCond         37
BsmtExposure     38
BsmtFinType1     37
BsmtFinSF1        0
BsmtFinType2     38
BsmtFinSF2        0
BsmtUnfSF         0
TotalBsmtSF       0
Heating           0
HeatingQC         0
CentralAir        0
Electrical        1
1stFlrSF          0
2ndFlrSF          0
LowQualFinSF      0
GrLivArea         0
BsmtFullBath      0
BsmtHalfBath      0
FullBath          0
HalfBath          0
BedroomAbvGr      0
KitchenAbvGr      0
KitchenQual       0
TotRmsAbvGrd      0
Functional        0
Fireplaces        0
FireplaceQu     690
GarageType       81
GarageYrBlt      81
GarageFinish     81
GarageCars        0
GarageArea        0
PavedDrive        0
WoodDeckSF        0
OpenPorchSF       0
EnclosedPorch     0
3SsnPorch         0
ScreenPorch       0
PoolArea          0
MiscVal           0
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
dtype: int64
```

In [ ]:

```
df_new=df_new.drop(['MasVnrArea', 'MasVnrType'], axis=1)
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
```

```
                    ):
  print(df_new.isna().sum())
```

```
MSSubClass          0
MSZoning            0
LotFrontage       259
LotArea             0
Street              0
LotShape            0
LandContour         0
Utilities           0
LotConfig           0
LandSlope           0
Neighborhood        0
Condition1          0
Condition2          0
BldgType            0
HouseStyle          0
OverallQual         0
OverallCond         0
YearBuilt           0
YearRemodAdd        0
RoofStyle           0
RoofMatl            0
Exterior1st         0
Exterior2nd         0
ExterQual           0
ExterCond           0
Foundation          0
BsmtQual           37
BsmtCond           37
BsmtExposure       38
BsmtFinType1       37
BsmtFinSF1          0
BsmtFinType2       38
BsmtFinSF2          0
BsmtUnfSF           0
TotalBsmtSF         0
Heating             0
HeatingQC           0
CentralAir          0
Electrical          1
1stFlrSF            0
2ndFlrSF            0
LowQualFinSF        0
GrLivArea           0
BsmtFullBath        0
BsmtHalfBath        0
FullBath            0
HalfBath            0
BedroomAbvGr        0
KitchenAbvGr        0
KitchenQual         0
TotRmsAbvGrd        0
Functional          0
Fireplaces          0
FireplaceQu       690
GarageType         81
GarageYrBlt        81
GarageFinish       81
GarageCars          0
GarageArea          0
PavedDrive          0
WoodDeckSF          0
OpenPorchSF         0
EnclosedPorch       0
3SsnPorch           0
ScreenPorch         0
PoolArea            0
MiscVal             0
MoSold              0
YrSold              0
```

```
SaleType            0
SaleCondition       0
SalePrice           0
dtype: int64
```

In [ ]:

```
df = df_new
```

In [ ]:

```
df.shape
```

Out[ ]:

```
(1460, 72)
```

In [ ]:

```
df = pd.read_csv('/content/drive/MyDrive/data_python/AGU_rules/house_train.csv')
#df.drop('Id', axis=1, inplace=True)
df.head()
```

Out[ ]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	I
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	

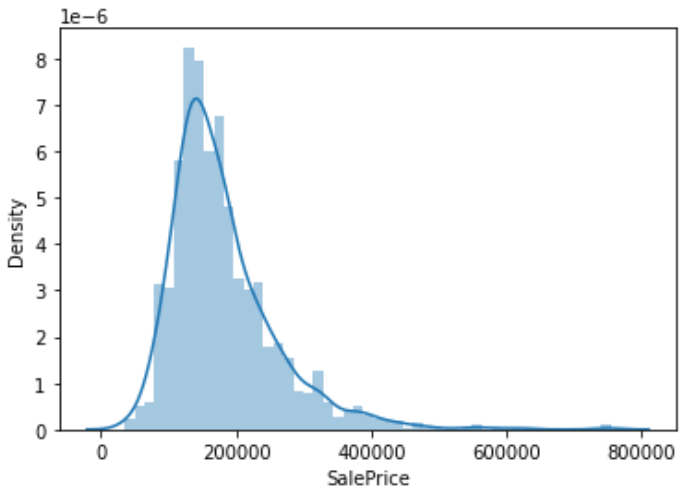
In [ ]:

```
sns.distplot(df['SalePrice'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe32b8d990>
```



Задание По рисунку: цена дома подчиняется нормальному распределению?

In [ ]:

```
# Тест Шапиро-Уилка
from scipy.stats import shapiro
stat, p = shapiro(df['SalePrice'])
print('Statistics=%.3f, p=%.3f' % (stat, p))
# Интерпретация
alpha = 0.05
if p > alpha:
 print('Sample looks Gaussian (fail to reject H0)')
else:
 print('Sample does not look Gaussian (reject H0)')
```

```
Statistics=0.870, p=0.000
Sample does not look Gaussian (reject H0)
```

In [ ]:

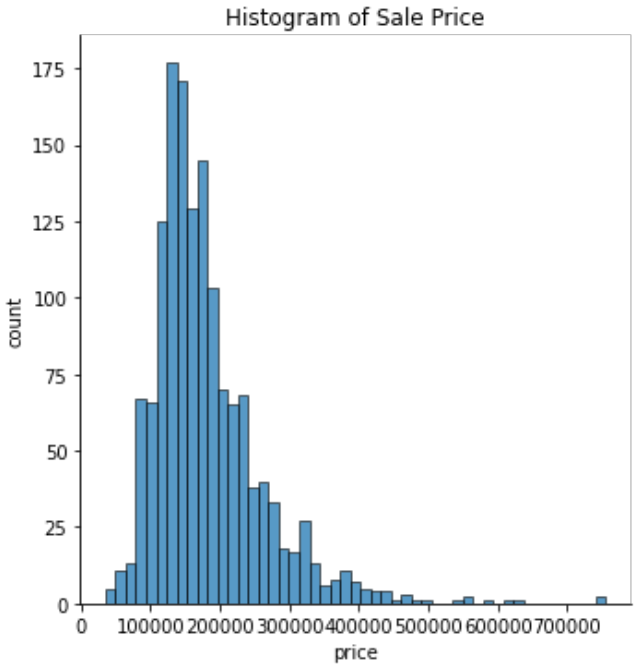
```
#Поскольку набор данных не ялвяется нормальным, то ассиметрию и эксцесс расчитывать не бу
дем
#print("Skewness: %f" % df['SalePrice'].skew())
#print("Kurtosis: %f" % df['SalePrice'].kurt())
```

```
Skewness: 1.882876
Kurtosis: 6.536282
```

Задание: Постройте гистограмму параметра **SalePrice** всех домов **с** заголовком **'Histogram of Sale Price'**, заголовок оси x **– 'price',** заголовок оси y **– 'count'.**

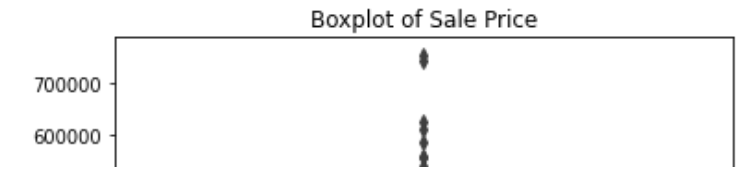
In [ ]:

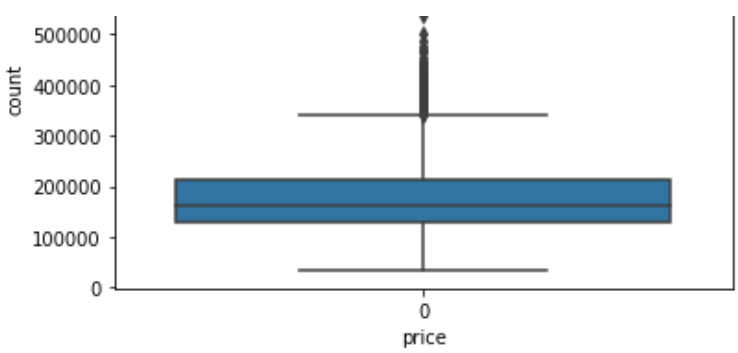
```
sns.displot(data=df['SalePrice'])
plt.xlabel('price')
plt.ylabel('count')
plt.title('Histogram of Sale Price')
plt.show()
```



In [ ]:

```
sns.boxplot(data=df['SalePrice'])
plt.xlabel('price')
plt.ylabel('count')
plt.title('Boxplot of Sale Price')
plt.show()
```





In [ ]:

```
df.groupby('CentralAir')['SalePrice'].describe()
```

Out[ ]:

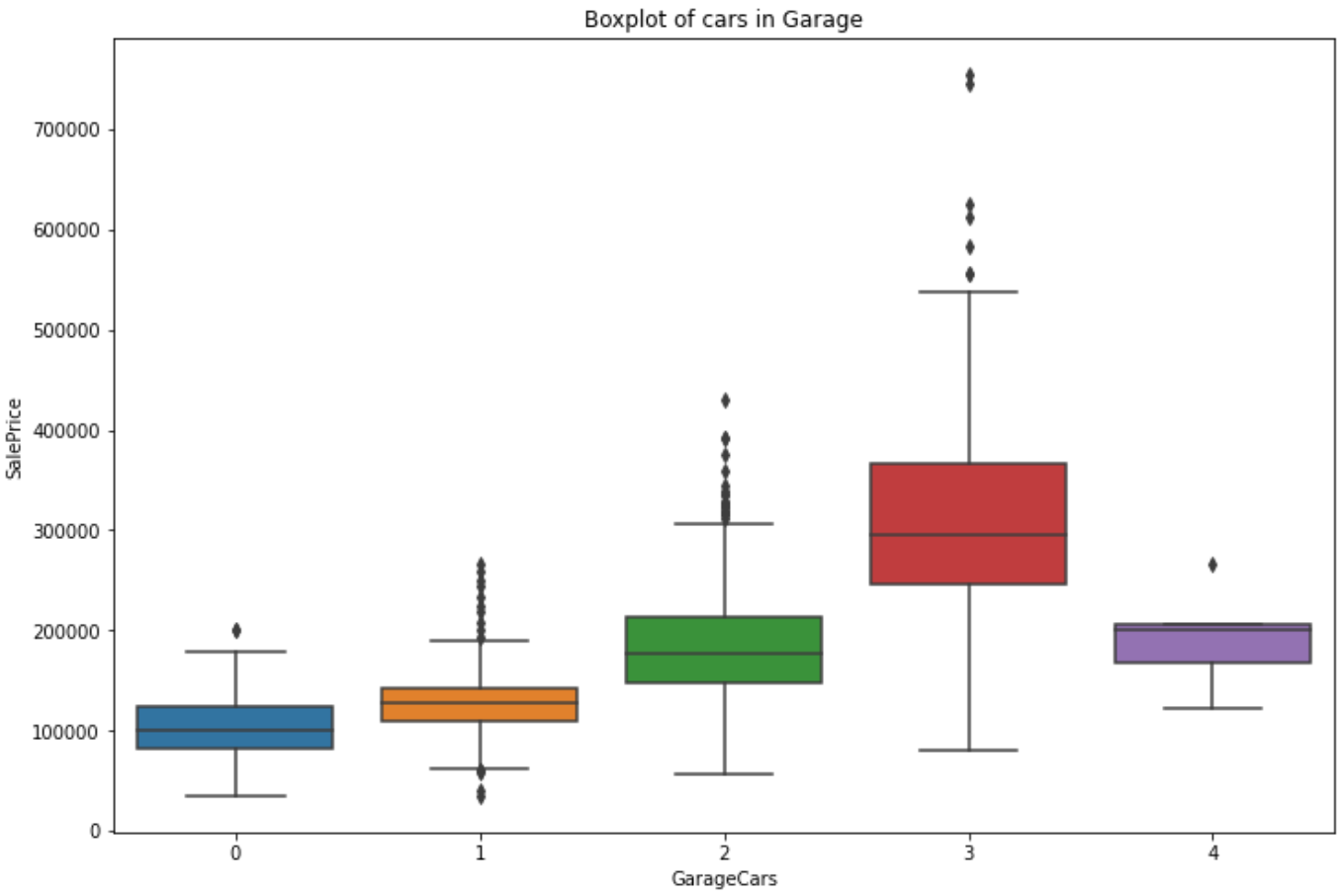
	count	mean	std	min	25%	50%	75%	max
CentralAir								
N	95.0	105264.073684	40671.273961	34900.0	82000.0	98000.0	128500.0	265979.0
Y	1365.0	186186.709890	78805.206820	52000.0	134800.0	168000.0	219210.0	755000.0

In [ ]:

```
plt.figure(figsize=(12, 8))
sns.boxplot(x="GarageCars", y="SalePrice", data=df)
plt.title('Boxplot of cars in Garage')
```

Out[ ]:

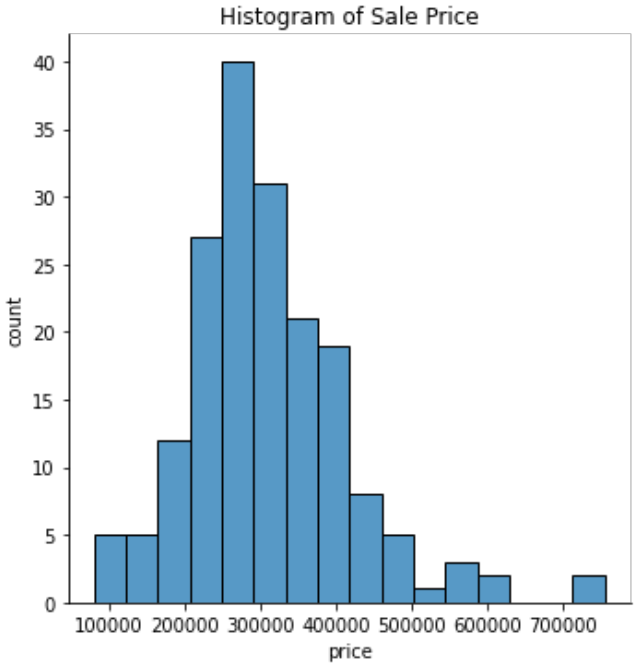
```
Text(0.5, 1.0, 'Boxplot of cars in Garage')
```



In [ ]:

```
#Гистограмму цены продажи дома в зависимости от количество машино-мест в  гараже (взято 3
)
sns.displot(df.loc[df['GarageCars'] == 3]['SalePrice'])
plt.xlabel('price')
```

```
plt.ylabel('count')
plt.title('Histogram of Sale Price')
plt.show()
```



In [ ]:

```
x = df.OverallQual.value_counts()
x/x.sum()
```

Out[ ]:

```
5     0.271918
6     0.256164
7     0.218493
8     0.115068
4     0.079452
9     0.029452
3     0.013699
10    0.012329
2     0.002055
1     0.001370
Name: OverallQual, dtype: float64
```

In [ ]:

```
x = df.GarageCars.value_counts()
x/x.sum()
```

Out[ ]:

```
2    0.564384
1    0.252740
3    0.123973
0    0.055479
4    0.003425
Name: GarageCars, dtype: float64
```

Гаражи какого размера наиболее распространены?

- на **2** машины

In [ ]:

```
x = df.CentralAir.value_counts()
x/x.sum()
```

Out[ ]:

```
Y    0.934932
```

```
N    0.065068
Name: CentralAir, dtype: float64
```

In [ ]:

```
df.SalePrice.describe()
```

Out[ ]:

```
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
```

In [ ]:

```
print('The proportion of the houses with prices between 25th percentile and 75th percenti
le: \n',np.mean((df.SalePrice >= 129975) & (df.SalePrice <= 214000)))
```

```
The proportion of the houses with prices between 25th percentile and 75th percentile:
 0.5020547945205479
```

In [ ]:

```
df.TotalBsmtSF.describe()
```

Out[ ]:

```
count    1460.000000
mean     1057.429452
std       438.705324
min         0.000000
25%       795.750000
50%       991.500000
75%      1298.250000
max      6110.000000
Name: TotalBsmtSF, dtype: float64
```

In [ ]:

```
print('The proportion of house with total square feet of basement area between 25th perce
ntile and 75th percentile: \n', np.mean((df.TotalBsmtSF >= 795.75) & (df.TotalBsmtSF <=
1298.25)))
```

```
The proportion of house with total square feet of basement area between 25th percentile a
nd 75th percentile:
 0.5
```

In [ ]:

```
a = (df.SalePrice >= 129975) & (df.SalePrice <= 214000)
b = (df.TotalBsmtSF >= 795.75) & (df.TotalBsmtSF <= 1298.25)
print(np.mean(a | b))
```

```
0.7143835616438357
```

In [ ]:

```
q75, q25 = np.percentile(df.loc[df['CentralAir']=='N']['SalePrice'], [75,25])
iqr = q75 - q25
print('Sale price IQR for houses with no air conditioning: ', iqr)
```

```
Sale price IQR for houses with no air conditioning:  46500.0
```

In [ ]:

```
q75, q25 = np.percentile(df.loc[df['CentralAir']=='Y']['SalePrice'], [75,25])
```

```
iqr = q75 - q25
print('Sale price IQR for houses with air conditioning: ', iqr)
```

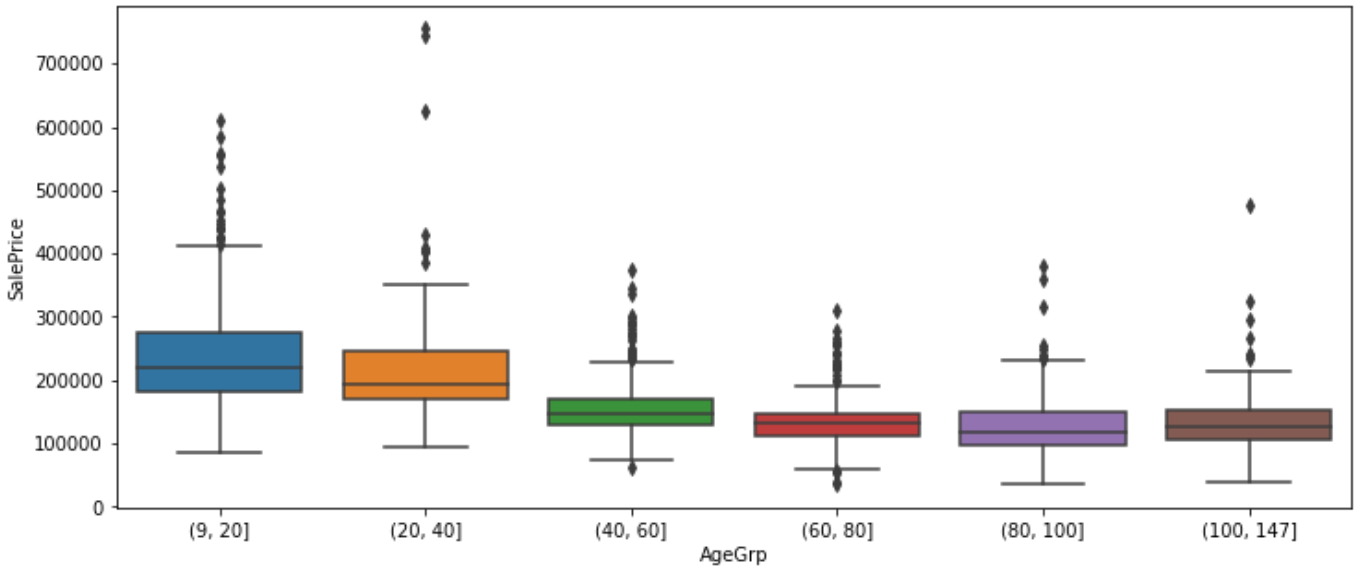
```
Sale price IQR for houses with air conditioning:  84410.0
```

In [ ]:

```
df['HouseAge'] = 2019 - df['YearBuilt']
df["AgeGrp"] = pd.cut(df.HouseAge, [9, 20, 40, 60, 80, 100, 147])
# Create age strata based on these cut points
plt.figure(figsize=(12, 5))
sns.boxplot(x="AgeGrp", y="SalePrice", data=df)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe1e6b35d0>
```

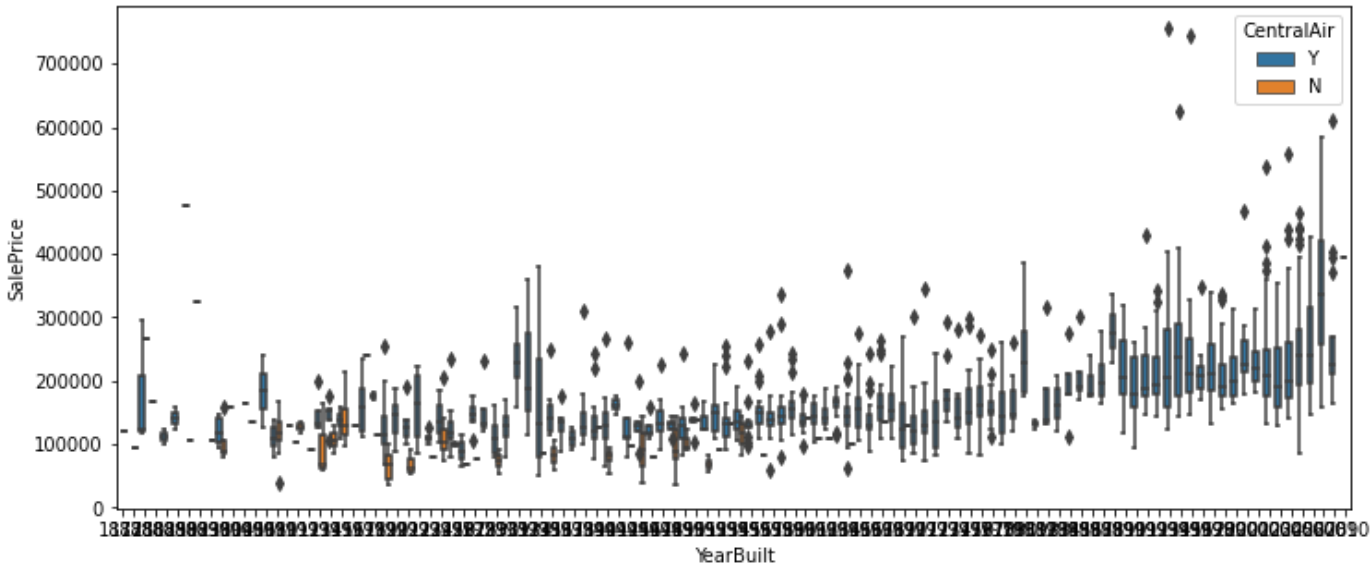


Вопрос: Сравните возраст дома и среднюю цену на него. Какая наблюдается закономерность?

- чем моложе дом, тем он дороже.
- после **60** лет стоимость почти не меняется, но после **100** лет чуть-чуть возрастает

In [ ]:

```
plt.figure(figsize=(12, 5))
sns.boxplot(x="YearBuilt", y="SalePrice", hue="CentralAir", data=df)
plt.show()
```

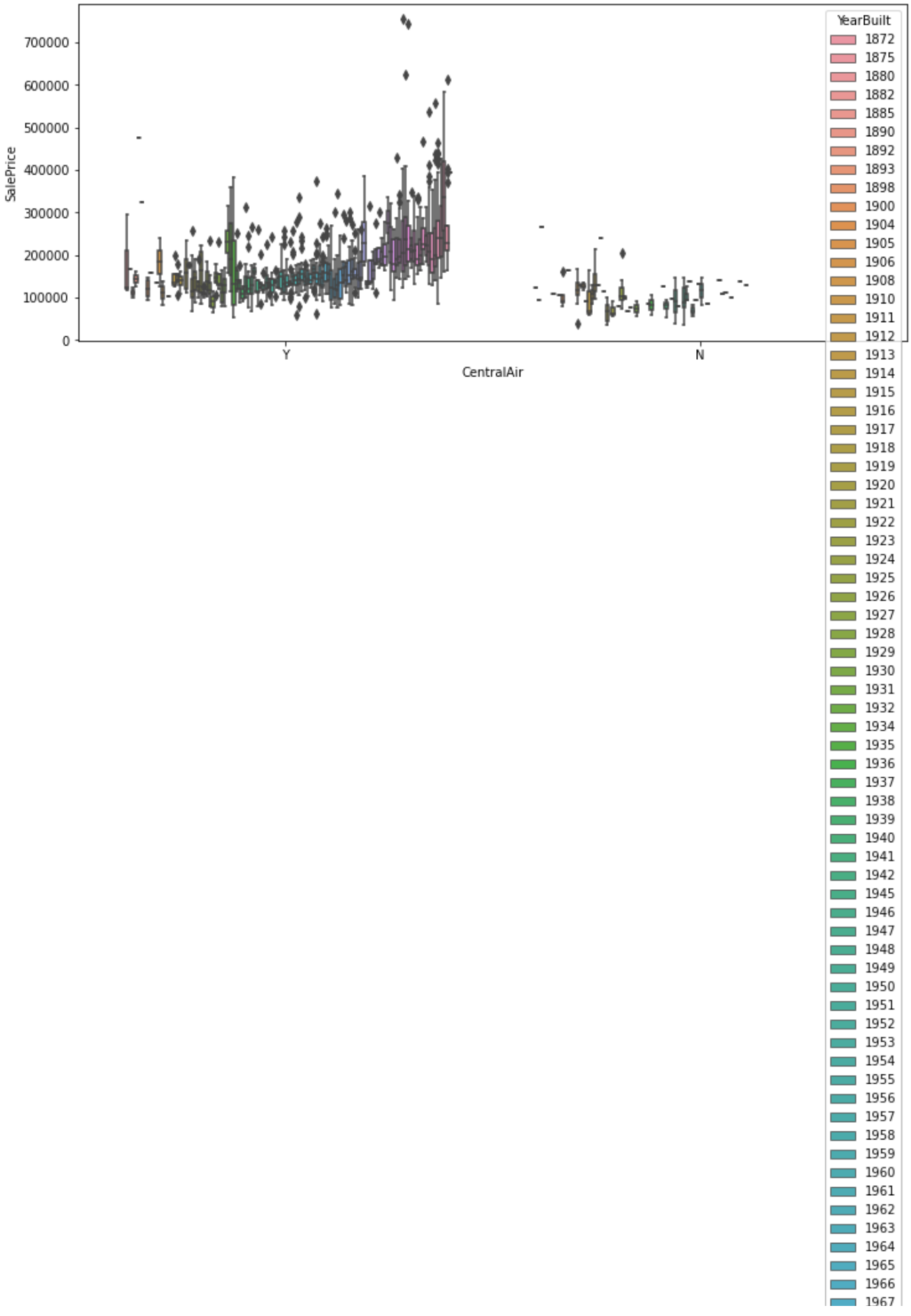


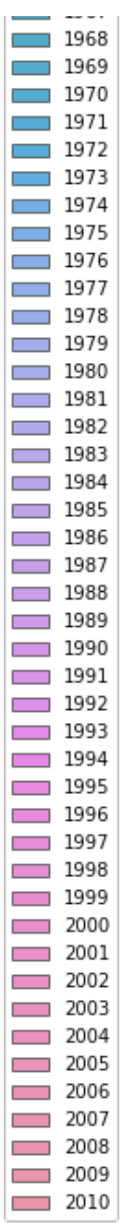
Сделайте вывод: какие по возрасту дома более оснащены кондиционерами.

- сделать вывод невозможно

In [ ]:

```
#очень длинная легенда.. как её сгруппировать?
plt.figure(figsize=(12, 5))
sns.boxplot(x="CentralAir", y="SalePrice", hue="YearBuilt", data=df)
plt.show()
```





In [ ]:

```
df1 = df.groupby(["YearBuilt", "CentralAir"])["BldgType"]
df1 = df1.value_counts()
df1 = df1.unstack()
df1 = df1.apply(lambda x: x/x.sum(), axis=1)
print(df1.to_string(float_format="%.3f"))
```

```
BldgType               1Fam  2fmCon  Duplex  Twnhs  TwnhsE
YearBuilt CentralAir
1872      N           1.000     NaN     NaN    NaN     NaN
1875      N           1.000     NaN     NaN    NaN     NaN
1880      N           1.000     NaN     NaN    NaN     NaN
          Y           1.000     NaN     NaN    NaN     NaN
1882      Y           1.000     NaN     NaN    NaN     NaN
1885      Y           0.500   0.500     NaN    NaN     NaN
1890      Y           1.000     NaN     NaN    NaN     NaN
1892      N           1.000     NaN     NaN    NaN     NaN
          Y           1.000     NaN     NaN    NaN     NaN
1893      Y           1.000     NaN     NaN    NaN     NaN
1898      N           1.000     NaN     NaN    NaN     NaN
1900      N           0.400   0.600     NaN    NaN     NaN
          Y           0.400   0.400   0.200    NaN     NaN
1904      Y           1.000     NaN     NaN    NaN     NaN
1905      N             NaN   1.000     NaN    NaN     NaN
1906      Y           1.000     NaN     NaN    NaN     NaN
1908      Y           1.000     NaN     NaN    NaN     NaN
1910      N           0.625   0.375     NaN    NaN     NaN
          Y           0.889   0.111     NaN    NaN     NaN
1911      N           1.000     NaN     NaN    NaN     NaN
1912      N           0.500   0.500     NaN    NaN     NaN
          Y           1.000     NaN     NaN    NaN     NaN
```

	Y	1.000	NaN	NaN	NaN	NaN
1913	N	1.000	NaN	NaN	NaN	NaN
1914	N	1.000	NaN	NaN	NaN	NaN
	Y	0.750	0.250	NaN	NaN	NaN
1915	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1916	N	1.000	NaN	NaN	NaN	NaN
	Y	0.750	0.250	NaN	NaN	NaN
1917	N	1.000	NaN	NaN	NaN	NaN
1918	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1919	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1920	N	1.000	NaN	NaN	NaN	NaN
	Y	0.958	0.042	NaN	NaN	NaN
1921	Y	1.000	NaN	NaN	NaN	NaN
1922	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1923	Y	1.000	NaN	NaN	NaN	NaN
1924	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1925	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1926	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1927	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1928	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1929	Y	1.000	NaN	NaN	NaN	NaN
1930	N	0.500	0.500	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1931	Y	1.000	NaN	NaN	NaN	NaN
1932	Y	1.000	NaN	NaN	NaN	NaN
1934	Y	1.000	NaN	NaN	NaN	NaN
1935	N	NaN	1.000	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1936	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1937	Y	1.000	NaN	NaN	NaN	NaN
1938	Y	1.000	NaN	NaN	NaN	NaN
1939	Y	0.750	0.250	NaN	NaN	NaN
1940	N	1.000	NaN	NaN	NaN	NaN
	Y	0.941	0.059	NaN	NaN	NaN
1941	N	0.750	0.250	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1942	Y	1.000	NaN	NaN	NaN	NaN
1945	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1946	N	0.500	NaN	0.500	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1947	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1948	Y	0.929	NaN	0.071	NaN	NaN
1949	N	0.500	NaN	0.500	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1950	N	NaN	0.333	0.667	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1951	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1952	N	1.000	NaN	NaN	NaN	NaN
	Y	0.667	NaN	0.333	NaN	NaN
1953	N	NaN	1.000	NaN	NaN	NaN
	Y	0.909	0.091	NaN	NaN	NaN
1954	N	1.000	NaN	NaN	NaN	NaN
	Y	0.957	0.043	NaN	NaN	NaN
1955	N	NaN	0.333	0.667	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1956	Y	1.000	NaN	NaN	NaN	NaN
1957	N	1.000	NaN	NaN	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1958	Y	0.917	0.042	0.042	NaN	NaN
1959	Y	0.962	NaN	0.038	NaN	NaN

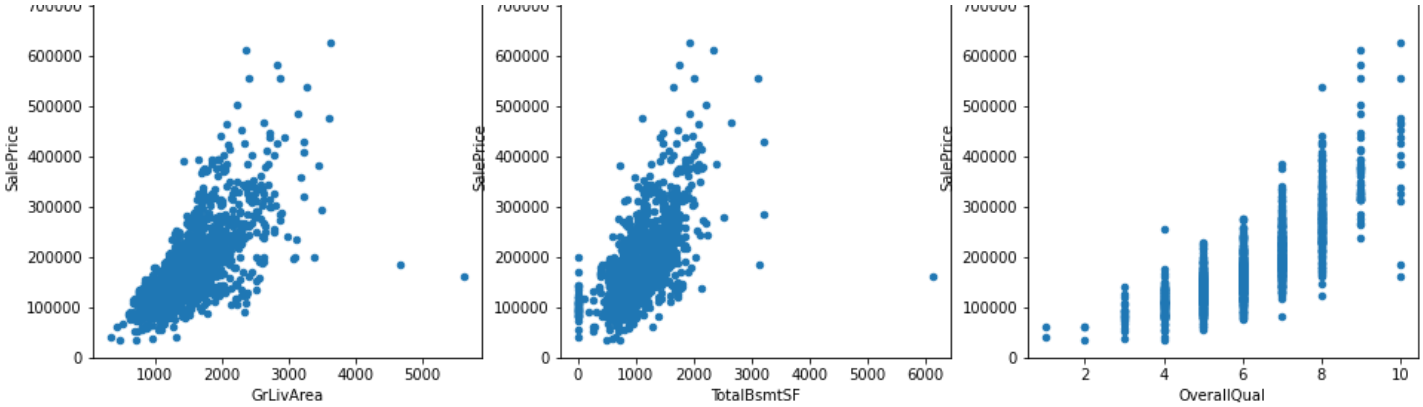
1959	Y	0.962	NaN	0.038	NaN	NaN
1960	Y	0.941	NaN	0.059	NaN	NaN
1961	N	NaN	NaN	1.000	NaN	NaN
	Y	0.923	0.077	NaN	NaN	NaN
1962	N	1.000	NaN	NaN	NaN	NaN
	Y	0.889	NaN	0.111	NaN	NaN
1963	N	NaN	NaN	1.000	NaN	NaN
	Y	0.933	NaN	0.067	NaN	NaN
1964	Y	0.933	NaN	0.067	NaN	NaN
1965	N	NaN	NaN	1.000	NaN	NaN
	Y	0.870	0.087	0.043	NaN	NaN
1966	Y	1.000	NaN	NaN	NaN	NaN
1967	Y	0.812	NaN	0.188	NaN	NaN
1968	N	NaN	NaN	1.000	NaN	NaN
	Y	1.000	NaN	NaN	NaN	NaN
1969	Y	0.857	NaN	0.143	NaN	NaN
1970	N	NaN	1.000	NaN	NaN	NaN
	Y	0.565	NaN	NaN	0.261	0.174
1971	Y	0.682	NaN	0.045	0.182	0.091
1972	Y	0.739	NaN	NaN	0.217	0.043
1973	Y	0.364	NaN	0.091	0.364	0.182
1974	Y	0.800	NaN	0.100	NaN	0.100
1975	Y	0.875	NaN	0.125	NaN	NaN
1976	Y	0.667	NaN	0.091	0.030	0.212
1977	Y	0.750	NaN	0.156	NaN	0.094
1978	Y	0.812	NaN	0.062	0.125	NaN
1979	Y	0.333	NaN	0.556	NaN	0.111
1980	Y	0.400	NaN	0.200	0.100	0.300
1981	Y	1.000	NaN	NaN	NaN	NaN
1982	Y	1.000	NaN	NaN	NaN	NaN
1983	Y	1.000	NaN	NaN	NaN	NaN
1984	Y	0.667	NaN	NaN	NaN	0.333
1985	Y	0.800	NaN	NaN	NaN	0.200
1986	Y	0.800	NaN	NaN	NaN	0.200
1987	Y	0.333	NaN	0.333	NaN	0.333
1988	Y	0.818	NaN	0.091	NaN	0.091
1989	Y	1.000	NaN	NaN	NaN	NaN
1990	Y	0.917	NaN	0.083	NaN	NaN
1991	Y	1.000	NaN	NaN	NaN	NaN
1992	Y	0.923	NaN	NaN	NaN	0.077
1993	Y	0.765	NaN	NaN	NaN	0.235
1994	Y	1.000	NaN	NaN	NaN	NaN
1995	Y	0.889	NaN	NaN	NaN	0.111
1996	Y	0.933	NaN	NaN	NaN	0.067
1997	Y	0.929	NaN	0.071	NaN	NaN
1998	Y	0.800	NaN	NaN	0.040	0.160
1999	Y	0.600	NaN	NaN	0.200	0.200
2000	Y	0.750	NaN	NaN	0.125	0.125
2001	Y	0.900	NaN	NaN	NaN	0.100
2002	Y	1.000	NaN	NaN	NaN	NaN
2003	Y	0.756	NaN	NaN	0.067	0.178
2004	Y	0.685	NaN	NaN	0.056	0.259
2005	Y	0.672	NaN	NaN	0.031	0.297
2006	Y	0.821	NaN	NaN	0.015	0.164
2007	Y	0.878	NaN	NaN	0.041	0.082
2008	Y	0.913	NaN	NaN	NaN	0.087
2009	Y	0.833	NaN	NaN	NaN	0.167
2010	Y	1.000	NaN	NaN	NaN	NaN

In [ ]:

```
output,var,var1,var2 = 'SalePrice', 'GrLivArea', 'TotalBsmtSF', 'OverallQual'
fig, axes = plt.subplots(nrows=1,ncols=3,figsize=(16,5))
df.plot.scatter(x=var,y=output,ylim=(0,800000),ax=axes[0])
df.plot.scatter(x=var1,y=output,ylim=(0,800000),ax=axes[1])
df.plot.scatter(x=var2,y=output,ylim=(0,800000),ax=axes[2])
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe1e950690>
```

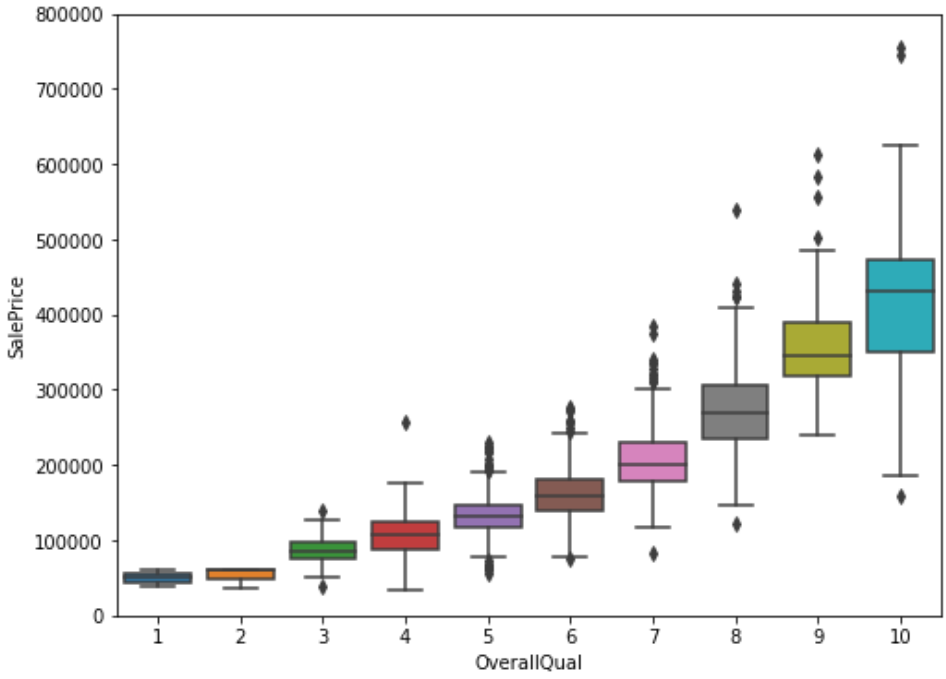


Наблюдается ли взаимосвязь между переменными**?**

-да

In [ ]:

```
fig, ax = plt.subplots(figsize=(8,6))
sns.boxplot(x=var2,y=output,data=df)
ax.set_ylim(0,800000)
plt.show()
```

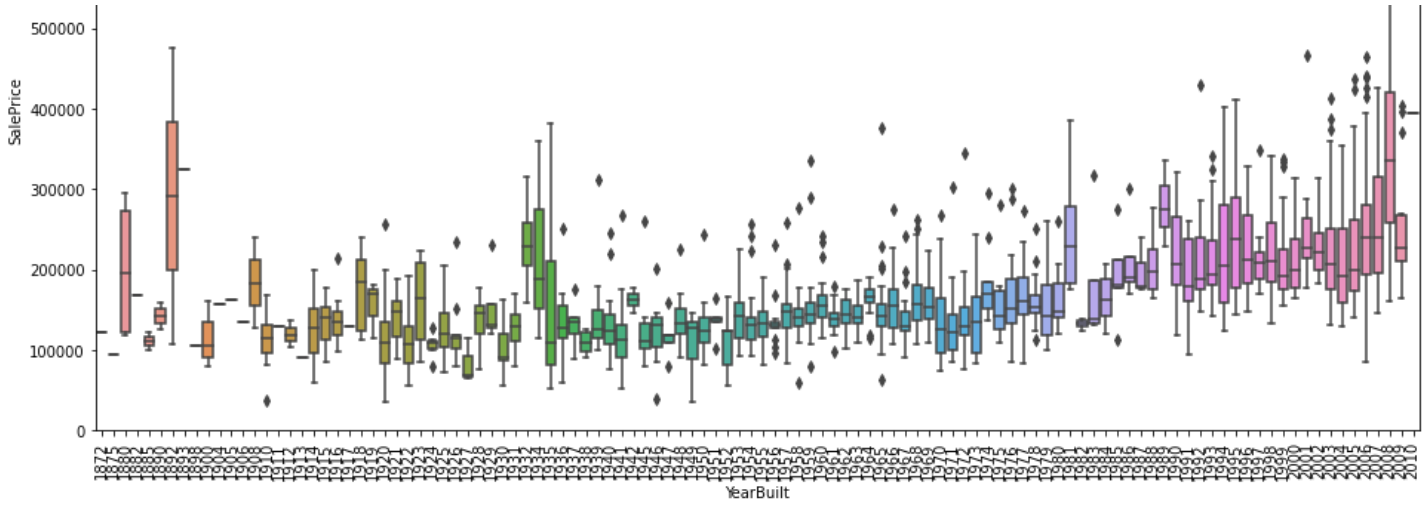


Между какими параметрами построен **boxplot?** Есть ли выбросы в данных**?**

- между ценой и оценкой качества дома**.** Выбросы есть **-** чем выше качество**,** тем больше возникают выбросы

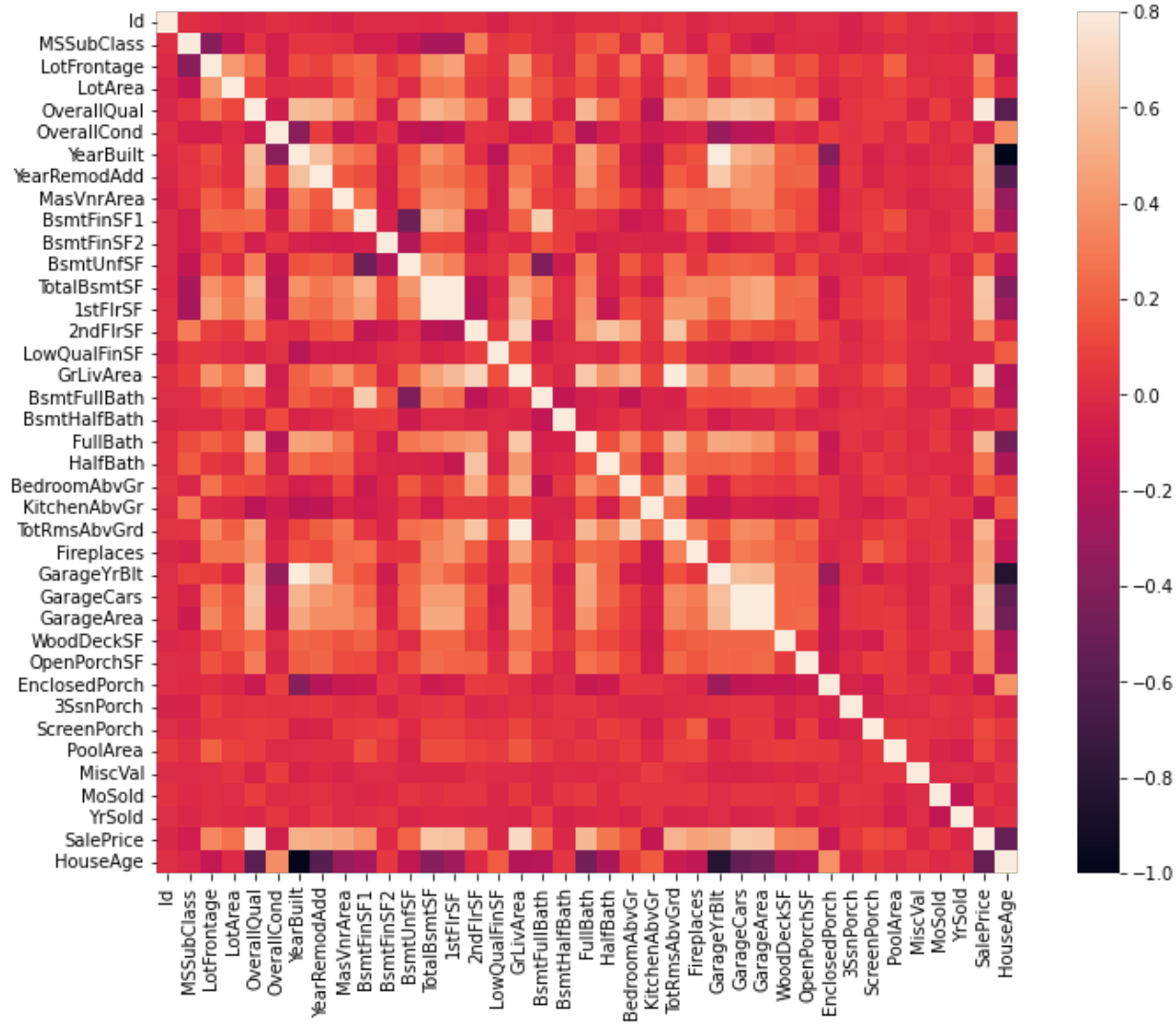
In [ ]:

```
var3 = 'YearBuilt'
fig, ax = plt.subplots(figsize=(16,8))
sns.boxplot(x=var3,y=output,data=df)
ax.set_ylim(0,800000)
plt.xticks(rotation=90)
plt.show()
```



In [ ]:

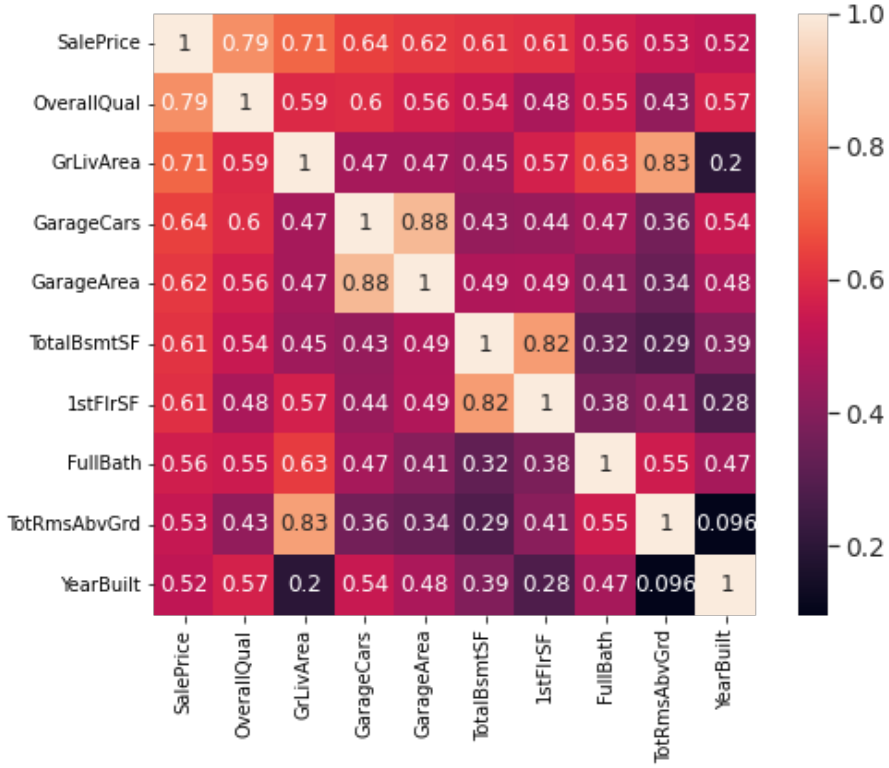
```
corrmat = df.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap (corrmat, vmax = .8, square = True, ax = ax)
# Параметр square гарантирует, что когда corrmat - неквадратная матрица, общий вывод граф
ика по-прежнему будет квадратным
plt.show()
```



In [ ]:

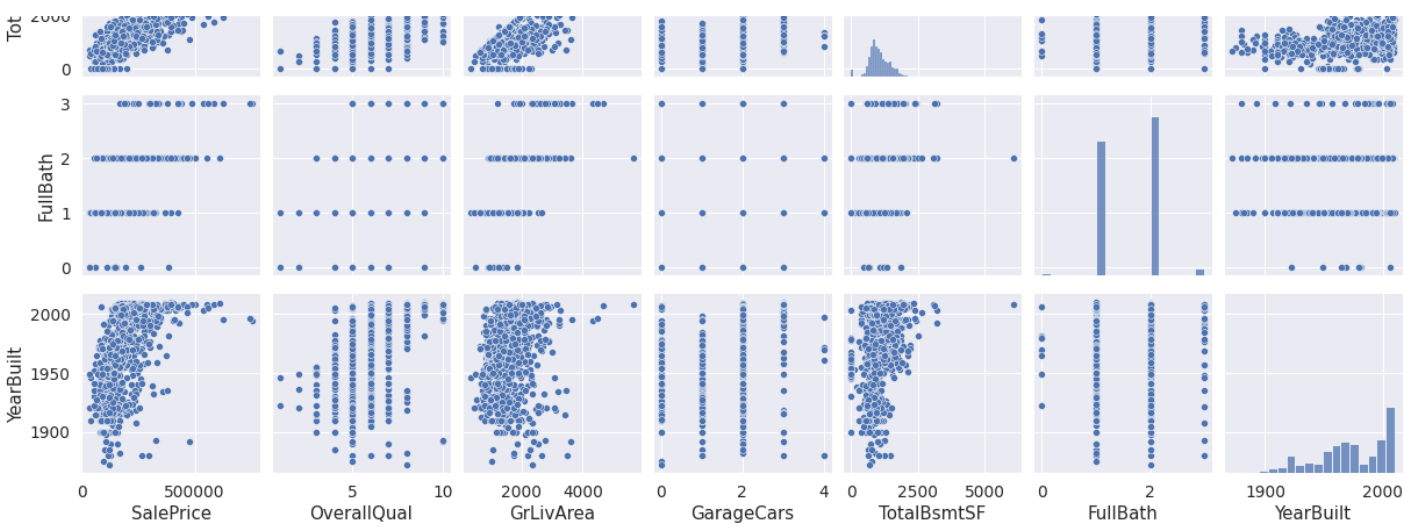
```
k = 10
top10_attr = corrmat.nlargest(k, output).index
top10_mat = corrmat.loc[top10_attr, top10_attr]
fig,ax = plt.subplots(figsize=(8,6))
sns.set(font_scale=1.25)
sns.heatmap(top10_mat, annot=True, annot_kws={'size':12}, square=True)
# Установите аннотацию для отображения чисел в маленьких ячейках и annot_kws для настройк
```

```
и числового формата
plt.show()
```



In [ ]:

```
var_set = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'FullBa
th', 'YearBuilt']
sns.set (font_scale = 1.25)
# Устанавливаем размер шрифта по горизонтальной и вертикальной оси
sns.pairplot (df[var_set])
# 7 * 7 графическая матрица
# Различные типы отображения могут быть установлены в параметрах kind и diag_kind,
# вот диаграммы разброса и гистограммы, и вы также можете установить разные типы
# отображения на каждом графике
plt.show()
```


Tot
2000
0
FullBath
3
2
1
0
YearBuilt
2000
1950
1900
0
500000
SalePrice
5
10
OverallQual
2000
4000
GrLivArea
0
2
4
GarageCars
0
2500
5000
TotalBsmtSF
0
2
FullBath
1900
2000
YearBuilt