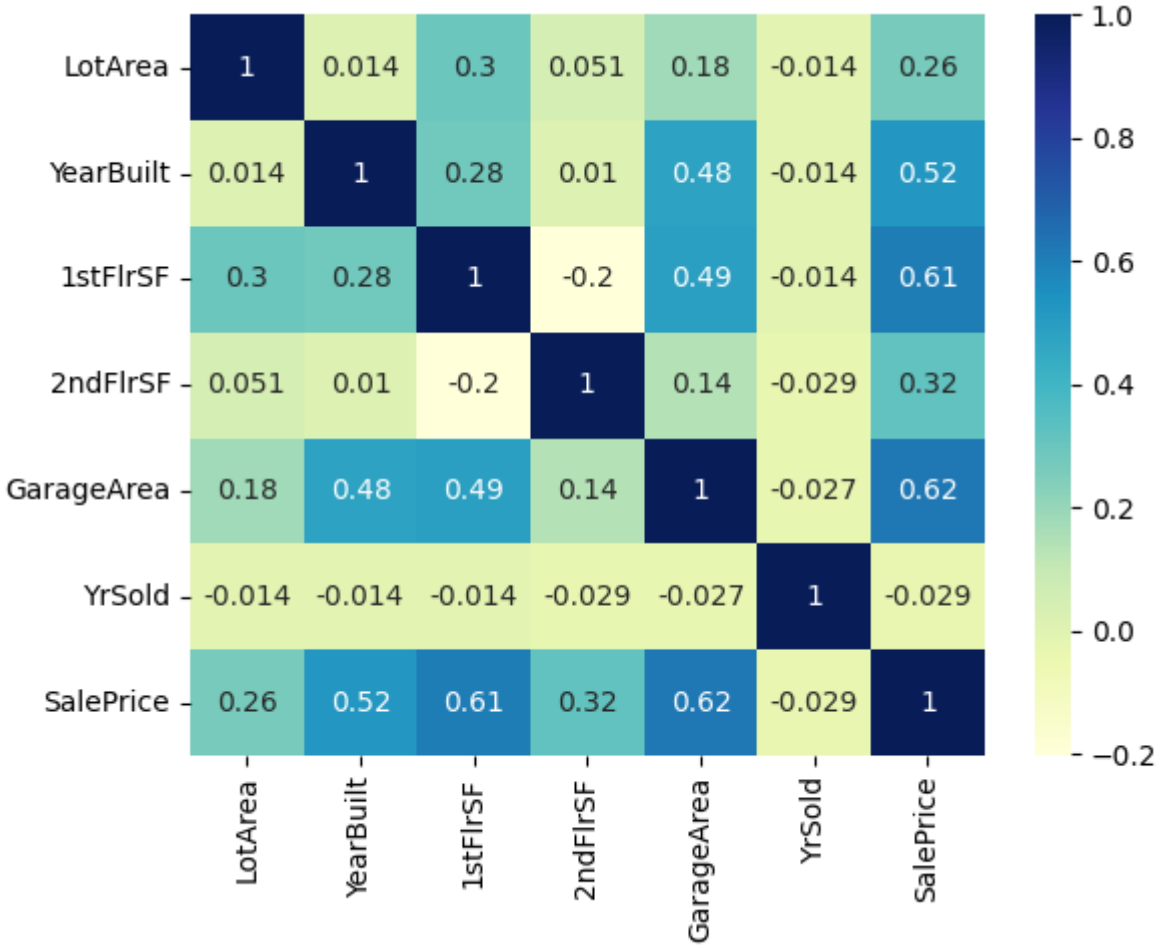
In [1]:
```
import pandas as  pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [2]:
```
df=pd.read_csv("train.csv")
```

In [3]:
```
cols = ['LotArea', 'Street','Alley','Utilities','LandSlope','Neighborhood','Condition1','Year
       'CentralAir','Electrical','1stFlrSF','2ndFlrSF','Functional','GarageArea','YrSold','Sa
```

In [4]:
```
data = df[cols]
```

In [5]:
```
sns.heatmap(data.corr(), annot=True, cmap="YlGnBu")
plt.show()
```



In [6]:
```
sf = ['LotArea','YearBuilt','1stFlrSF','2ndFlrSF','GarageArea']
```

In [7]:
```
x = df[sf]
y = df['SalePrice']
```

In [8]:
```
x.head()
```

Out[8]:

	LotArea	YearBuilt	1stFlrSF	2ndFlrSF	GarageArea
0	8450	2003	856	854	548
1	9600	1976	1262	0	460
2	11250	2001	920	866	608
3	9550	1915	961	756	642
4	14260	2000	1145	1053	836

In [9]:
```
x.isnull().sum()
```

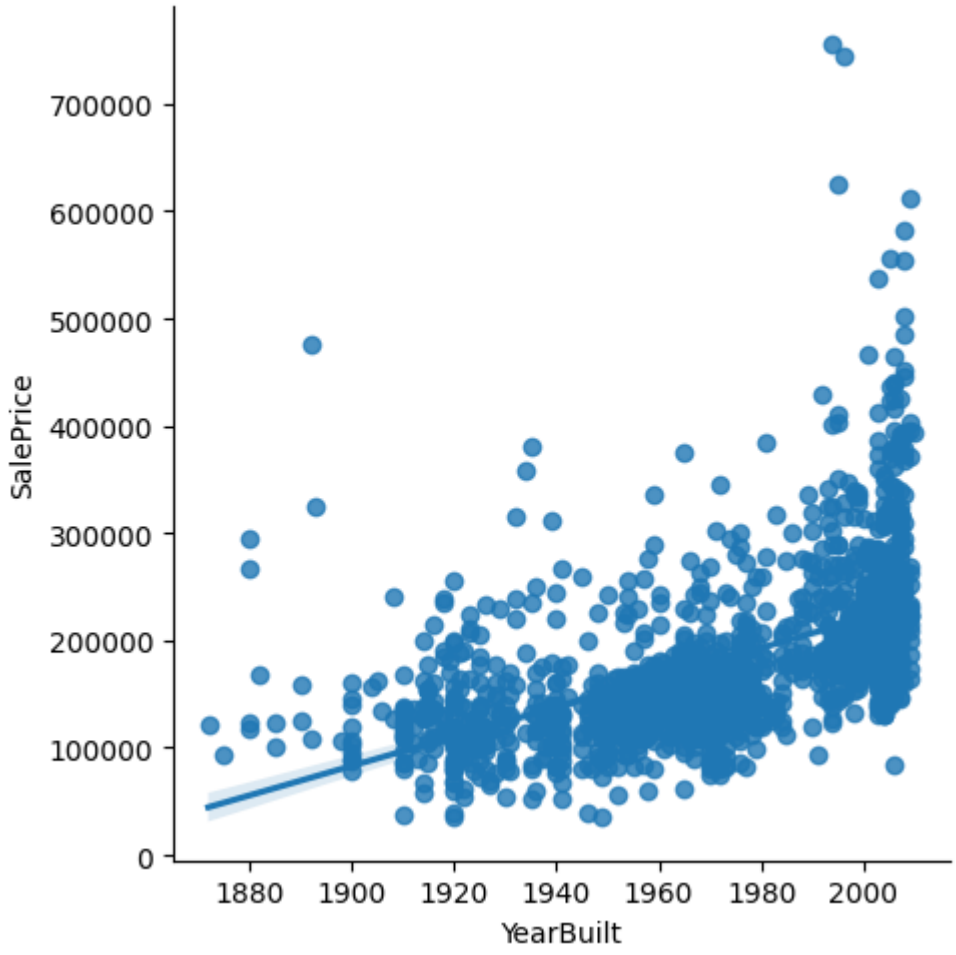
Out[9]:
```
LotArea       0
YearBuilt     0
1stFlrSF      0
2ndFlrSF      0
GarageArea    0
dtype: int64
```

In [10]:
```
y.isnull().sum()
```

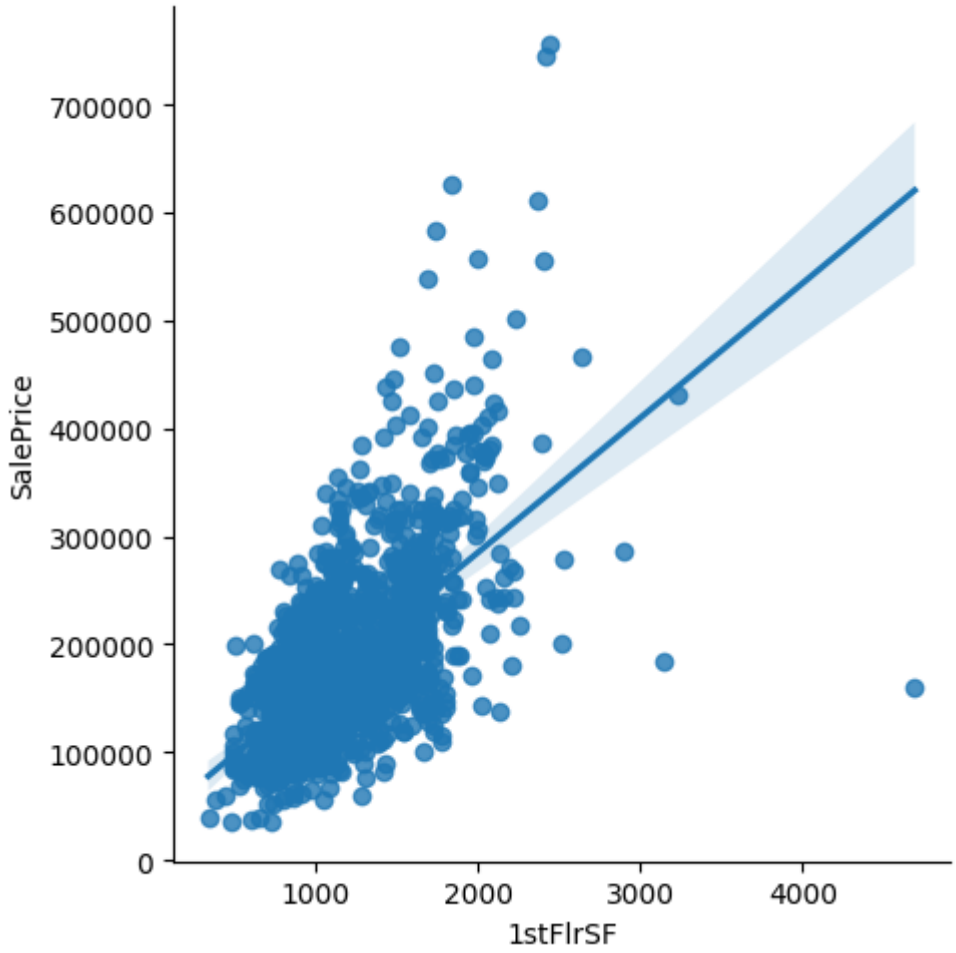
Out[10]:
```
0
```

In [11]:
```
#sns.lmplot(x="LotArea", y="SalePrice", data=df);
```

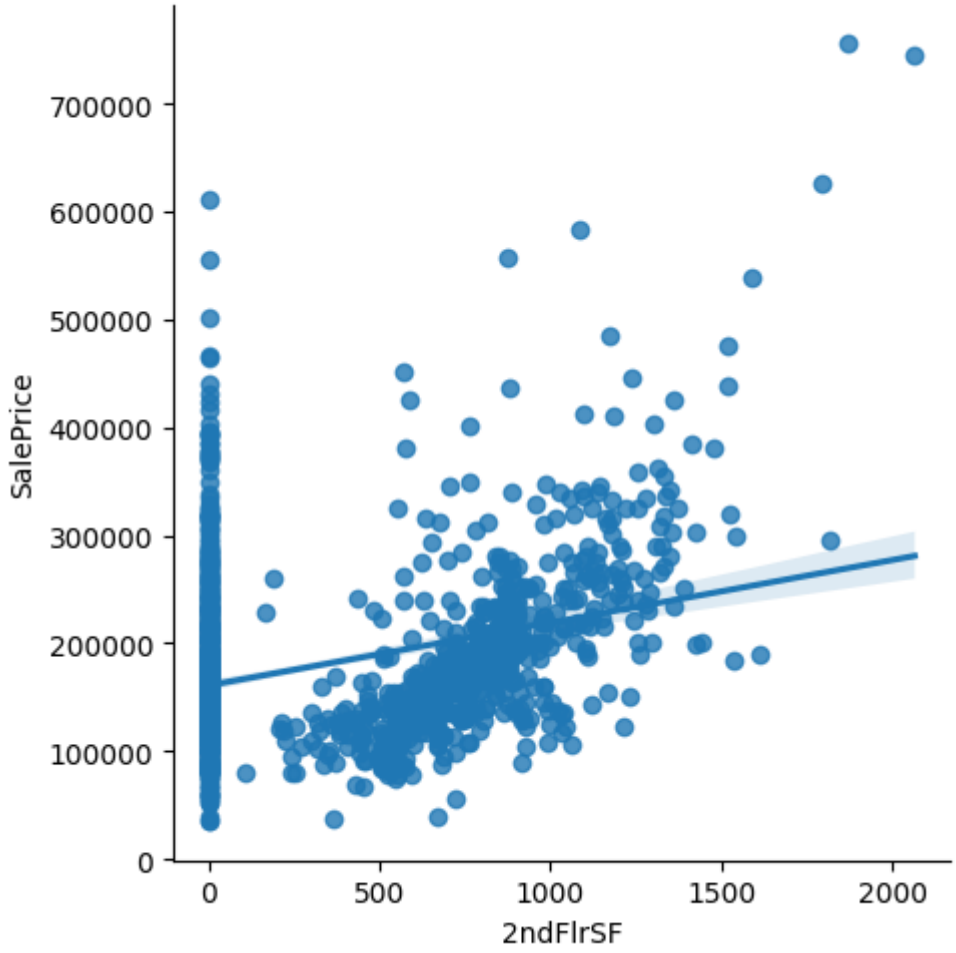
In [12]:
```
sns.lmplot(x="YearBuilt", y="SalePrice", data=df);
```



In [13]:
```
sns.lmplot(x="1stFlrSF", y="SalePrice", data=df);
```

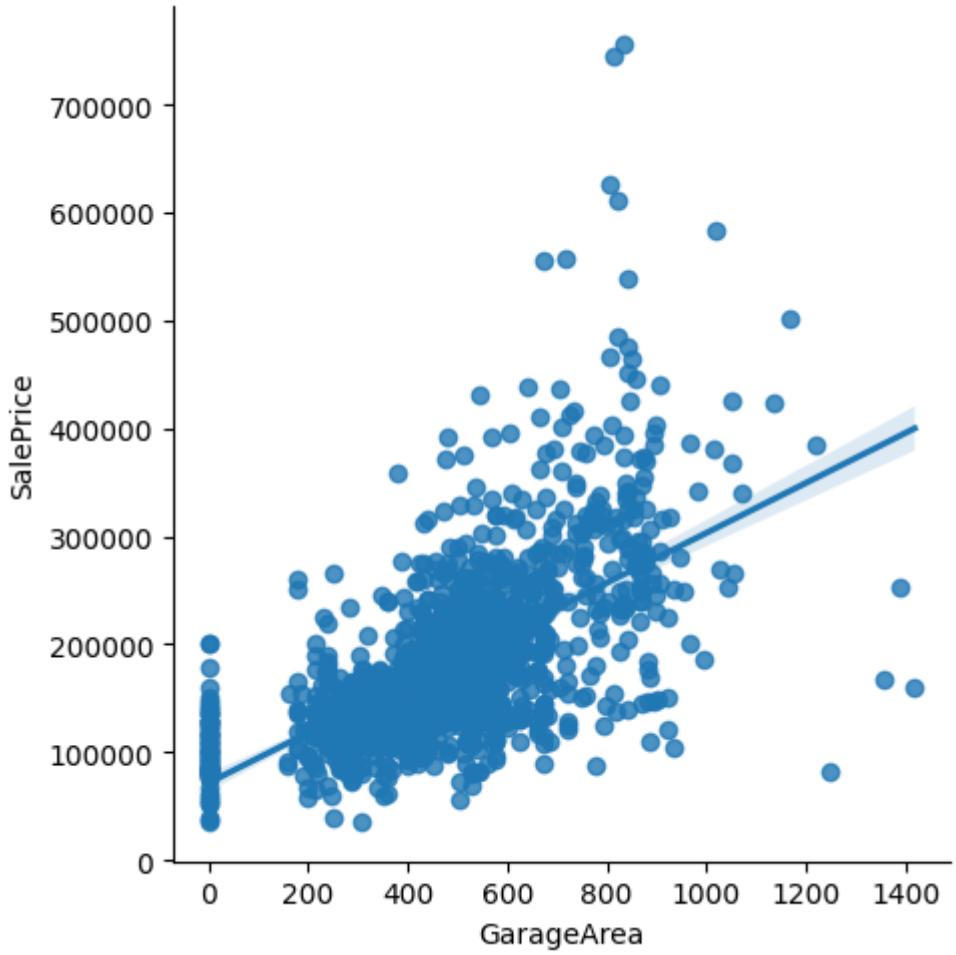


In [14]:
```
sns.lmplot(x="2ndFlrSF", y="SalePrice", data=df);
```



In [15]:
```
sns.lmplot(x="GarageArea", y="SalePrice", data=df)
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0x1d3f6d18940>



In [16]:
```
# Calculating accuracy of all Regression Models
```

In [17]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 1)
```

In [18]:
```
y_train
```

Out[18]:
```
632      82500
208     277000
83      126500
1174    239000
250      76500
         ...
715     165000
905     128000
1096    127000
235      89500
1061     81000
Name: SalePrice, Length: 1022, dtype: int64
```

In [19]:
```
# Multiple Linear Regression
```

In [20]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[20]:
```
▾ LinearRegression
LinearRegression()
```

In [21]:
```
y_pred = regressor.predict(x_test)
```

In [22]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[22]:
```
0.7668673994774332
```

In [23]:
```
# polynomial Regression
```

In [24]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(x_train)
regressor = LinearRegression()
regressor.fit(X_poly, y_train)
```

Out[24]:
```
▾ LinearRegression
LinearRegression()
```

In [25]:
```
y_pred_poly = regressor.predict(poly_reg.transform(x_test))
```

In [26]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred_poly)
```

Out[26]:
```
-49.59965246653583
```

In [27]:
```
# Support Vector Regression
```

In [28]:
```
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(x_train, y_train)
```

Out[28]:
```
▾ SVR
SVR()
```

In [29]:
```
y_pred_svr = regressor.predict(x_test)
```

In [30]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_svr)
```

Out[30]:
```
-0.03769825564676732
```

In [31]:
```
# Decision Trees Regression
```

In [32]:
```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(x_train, y_train)
```

Out[32]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

In [33]:
```
y_pred_decision = regressor.predict(x_test)
```

In [34]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred_decision)
```

Out[34]:
```
0.7278545266174449
```

In [35]:
```
# Random Forest Regression
```

In [36]:
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(x_train, y_train)
```

Out[36]:
```
RandomForestRegressor
RandomForestRegressor(n_estimators=10, random_state=0)
```

In [37]:
```
y_pred_random = regressor.predict(x_test)
```

In [38]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred_random)
```

Out[38]:
```
0.8191957026372843
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: