Our exploratory data analysis examined the housing market in Ames, Iowa, using the house price [SalePrice] as the dependent variable of interest. The initial dataset includes 1460 data records and 81 fields. The test data has a mean sale price of $180,921.20 and a standard deviation of $79442.50

We found that 19 of the fields had null values. It makes sense that some data would be missing for each; for example, fireplace quantity [FireplaceQu] may have been left blank if a house doesn't have a fireplace. We're going to drop all of the columns with null values except for [Electrical]. For [Electrical], we removed the row with the null value. If our client is particularly interested in one of the removed fields, there would be more time-intensive ways to clean the data and keep any required fields. To address outliers, we trimmed the training data down to only data records where house price falls within two standard deviations of the initial median value. This manipulation primarily removes outliers with higher sales prices. The cleaned data had 1,459 data records and 63 fields and a median sale price of $169,995.87 and a standard deviation of $58,943.80.

From the correlation heatmap, we have identified three potential predictors of SalesPrice based on having the highest correlation coefficient. We investigated [OverallQual], [GrLivingArea], and [GarageCars], which had correlation coefficients of 0.78, 0.66, and 0.63, respectively. Logically, these variables make sense to use as predictors for sales price since the [OverallQual] measures the overall look and build of

the house, and [GrLivingArea] and [GarageCars] measure the usable living space of the home.

We have created two new features to be used as predictor variables. First, we have summed the variables [1stFlrSF], [2ndFlrSF], and [TotalBsmtSF] to create a total square foot variable [tot_sq]. [TotalBsmtSF] has a moderate correlation coefficient of 0.54; however, this space is not included in the [GrLivingArea] variable since that only includes the above-ground area. Therefore, we have created a total square foot variable that incorporates the basement area for prediction purposes. The second predictor variable we created [qual_space] multiplies this new total square foot variable by the [OverallQual] variable. This provides us with a measurement of the quality of the total living space. A correlation heatmap of new predictor variables shows that [total_sq] has a correlation coefficient of 0.73 and [qual_space] has a correlation coefficient of 0.8.

Using the sklearn.preprocessing package, we used the MinMaxScaler and StandardScaler functions to perform both min-max and standard scaling on the [SalePrice] variable. The MinMaxScaler transforms the [SalePrice] data to be within the range of 0 to 1 and the StandardScaler transforms the data to have the mean equal to 0 with a standard deviation equal to 1.

https://canvas.northwestern.edu/courses/167719/assignments/1078596

https://www.kaggle.com/c/house-prices-advanced-regression-techniques

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

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

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   object 
 6   Alley          91 non-null     object 
 7   LotShape       1460 non-null   object 
 8   LandContour    1460 non-null   object 
 9   Utilities      1460 non-null   object 
 10  LotConfig      1460 non-null   object 
 11  LandSlope      1460 non-null   object 
 12  Neighborhood   1460 non-null   object 
 13  Condition1     1460 non-null   object 
 14  Condition2     1460 non-null   object 
 15  BldgType       1460 non-null   object 
 16  HouseStyle     1460 non-null   object 
 17  OverallQual    1460 non-null   int64  
 18  OverallCond    1460 non-null   int64  
 19  YearBuilt      1460 non-null   int64  
 20  YearRemodAdd   1460 non-null   int64  
 21  RoofStyle      1460 non-null   object 
 22  RoofMatl       1460 non-null   object 
 23  Exterior1st    1460 non-null   object 
 24  Exterior2nd    1460 non-null   object 
 25  MasVnrType     1452 non-null   object 
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object 
 28  ExterCond      1460 non-null   object 
 29  Foundation     1460 non-null   object 
 30  BsmtQual       1423 non-null   object 
 31  BsmtCond       1423 non-null   object 
 32  BsmtExposure   1422 non-null   object 
 33  BsmtFinType1   1423 non-null   object 
 34  BsmtFinSF1     1460 non-null   int64  
 35  BsmtFinType2   1422 non-null   object 
 36  BsmtFinSF2     1460 non-null   int64  
 37  BsmtUnfSF      1460 non-null   int64  
 38  TotalBsmtSF    1460 non-null   int64  
 39  Heating        1460 non-null   object 
 40  HeatingQC      1460 non-null   object 
 41  CentralAir     1460 non-null   object 
```

```
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [4]:
```
df.shape
```

Out[4]:
```
(1460, 81)
```

In [5]:
```
# how many unique IDs are there?
df["Id"].unique().shape
```

Out[5]:
```
(1460,)
```

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

Out[6]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	.
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	.
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	

5 rows × 81 columns

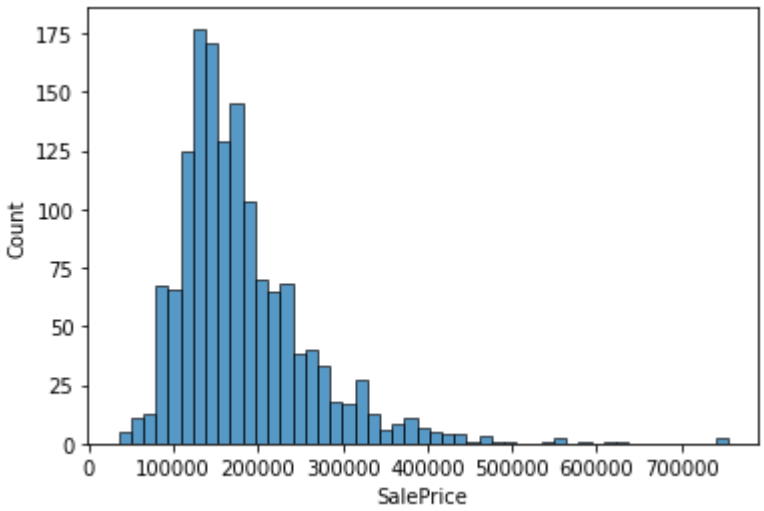
Provide appropriate descriptive statistics and visualizations to help understand the marginal distribution of the dependent variable.

In [7]:
```
df["SalePrice"].describe()
```

Out[7]:
```
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
```

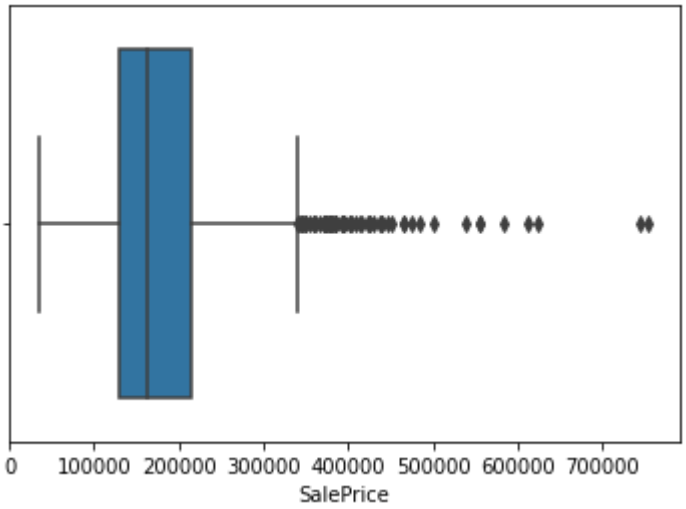
In [8]:
```
sns.histplot(x="SalePrice", data=df)
```

Out[8]:
```
<AxesSubplot:xlabel='SalePrice', ylabel='Count'>
```



In [9]:
```
sns.boxplot(x="SalePrice", data=df)
```

Out[9]:
```
<AxesSubplot:xlabel='SalePrice'>
```



Investigate missing data and outliers.

Missing Data:

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

Out[10]:
```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     259
LotArea           0
                 ...
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
Length: 81, dtype: int64
```

The following categories have null values:

- LotFrontage
- Alley
- MasVnrType
- MasVnrArea
- BsmtQual
- BsmtCond
- BsmtExposure
- BsmtFinType1
- BsmtFinType2
- Electrical
- FireplaceQu
- GarageType
- GarageYrBlt
- GarageFinish
- GarageQual
- GarageCond

- PoolQC
- Fence
- MiscFeature

We're not concerned about most of these columns having null values. It makes sense that some of the data would be missing for each (if a house doesn't have a pool, for example). We're going to drop all of the columns that have null values with the exception of "Electrical." For 'Electrical,' we'll remove the row with the null value.

In [11]:
```
col_to_drop = ['LotFrontage','Alley','MasVnrType','MasVnrArea',
               'BsmtQual', 'BsmtCond', 'BsmtExposure','BsmtFinType1', 'BsmtFinType2', '
               'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence', 'MiscFeat

df2 = df.drop(columns=col_to_drop, inplace=False)
df3 = df2.dropna()
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1459 entries, 0 to 1459
Data columns (total 63 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1459 non-null   int64
 1   MSSubClass     1459 non-null   int64
 2   MSZoning       1459 non-null   object
 3   LotArea        1459 non-null   int64
 4   Street         1459 non-null   object
 5   LotShape       1459 non-null   object
 6   LandContour    1459 non-null   object
 7   Utilities      1459 non-null   object
 8   LotConfig      1459 non-null   object
 9   LandSlope      1459 non-null   object
 10  Neighborhood   1459 non-null   object
 11  Condition1     1459 non-null   object
 12  Condition2     1459 non-null   object
 13  BldgType       1459 non-null   object
 14  HouseStyle     1459 non-null   object
 15  OverallQual    1459 non-null   int64
 16  OverallCond    1459 non-null   int64
 17  YearBuilt      1459 non-null   int64
 18  YearRemodAdd   1459 non-null   int64
 19  RoofStyle      1459 non-null   object
 20  RoofMatl       1459 non-null   object
 21  Exterior1st    1459 non-null   object
 22  Exterior2nd    1459 non-null   object
 23  ExterQual      1459 non-null   object
 24  ExterCond      1459 non-null   object
 25  Foundation     1459 non-null   object
 26  BsmtFinSF1     1459 non-null   int64
 27  BsmtFinSF2     1459 non-null   int64
 28  BsmtUnfSF      1459 non-null   int64
 29  TotalBsmtSF    1459 non-null   int64
 30  Heating        1459 non-null   object
 31  HeatingQC      1459 non-null   object
 32  CentralAir     1459 non-null   object
 33  Electrical     1459 non-null   object
 34  1stFlrSF       1459 non-null   int64
 35  2ndFlrSF       1459 non-null   int64
 36  LowQualFinSF   1459 non-null   int64
 37  GrLivArea      1459 non-null   int64
```

```
 38  BsmtFullBath   1459 non-null   int64
 39  BsmtHalfBath   1459 non-null   int64
 40  FullBath       1459 non-null   int64
 41  HalfBath       1459 non-null   int64
 42  BedroomAbvGr   1459 non-null   int64
 43  KitchenAbvGr   1459 non-null   int64
 44  KitchenQual    1459 non-null   object
 45  TotRmsAbvGrd   1459 non-null   int64
 46  Functional     1459 non-null   object
 47  Fireplaces     1459 non-null   int64
 48  GarageCars     1459 non-null   int64
 49  GarageArea     1459 non-null   int64
 50  PavedDrive     1459 non-null   object
 51  WoodDeckSF     1459 non-null   int64
 52  OpenPorchSF    1459 non-null   int64
 53  EnclosedPorch  1459 non-null   int64
 54  3SsnPorch      1459 non-null   int64
 55  ScreenPorch    1459 non-null   int64
 56  PoolArea       1459 non-null   int64
 57  MiscVal        1459 non-null   int64
 58  MoSold         1459 non-null   int64
 59  YrSold         1459 non-null   int64
 60  SaleType       1459 non-null   object
 61  SaleCondition  1459 non-null   object
 62  SalePrice      1459 non-null   int64
dtypes: int64(35), object(28)
memory usage: 729.5+ KB
```

Outliers

In [12]:
```
df3['SalePrice'].describe(percentiles = [.25, .5, .75, .95])
```

Out[12]:
```
count      1459.000000
mean     180930.394791
std       79468.964025
min       34900.000000
25%      129950.000000
50%      163000.000000
75%      214000.000000
95%      326200.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

In [13]:
```
#This code trims data to a certain number of standard deviations from the mean. We went
#You can see in the graphs below that it removes many outliers and normalizes the data.

from scipy import stats
import numpy as np

df4 = df3[(np.abs(stats.zscore(df3['SalePrice'])) < 2)]
```

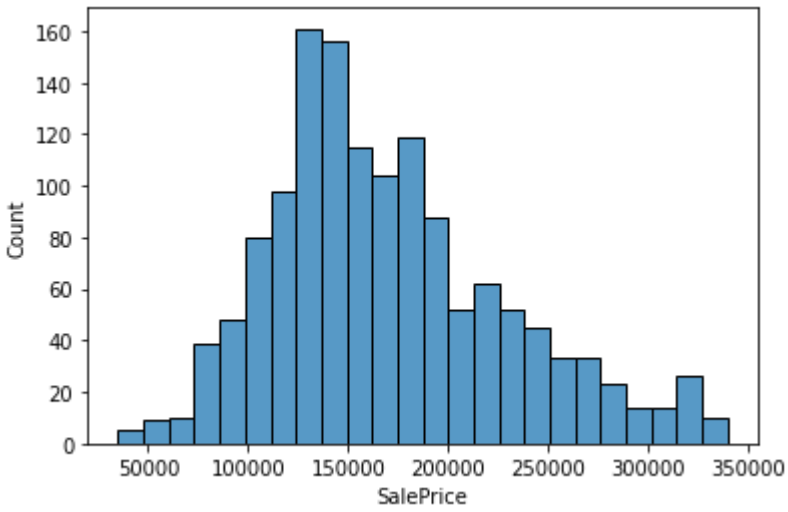
In [14]:
```
df4["SalePrice"].describe()
```

Out[14]:
```
count      1396.000000
mean     169995.874642
std       58943.796385
min       34900.000000
25%      128987.500000
50%      159467.000000
75%      203000.000000
```

```
max      339750.000000
Name: SalePrice, dtype: float64
```

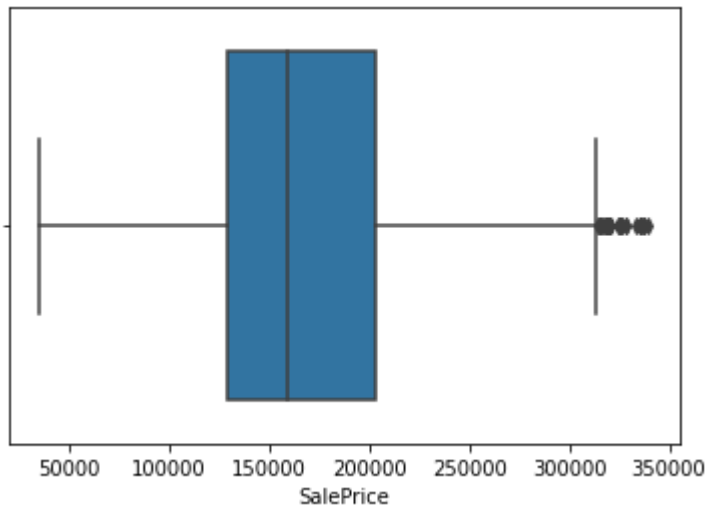
In [15]:
```
sns.histplot(x="SalePrice", data=df4)
```

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



In [16]:
```
sns.boxplot(x="SalePrice", data=df4)
```

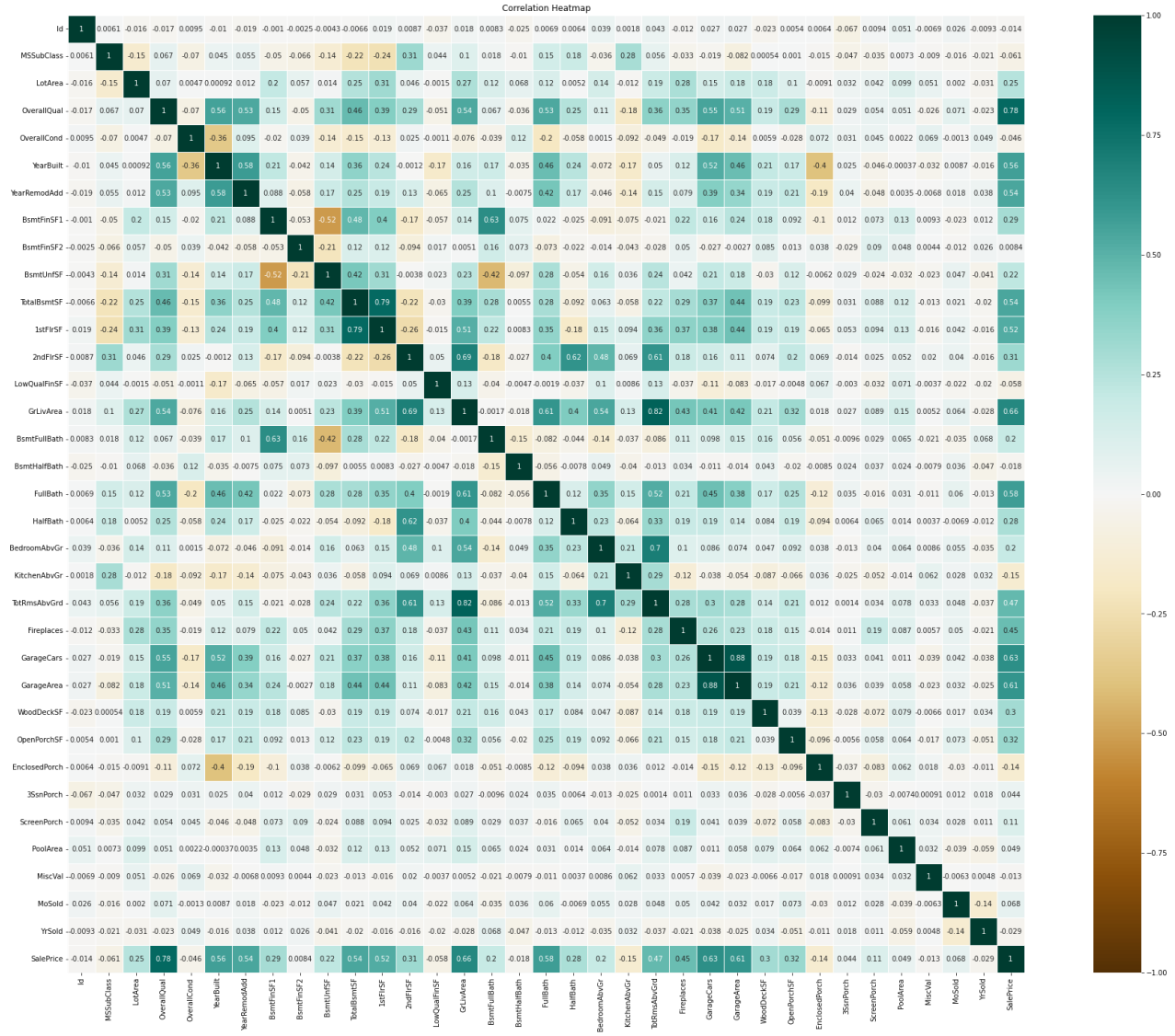
Out[16]: <AxesSubplot:xlabel='SalePrice'>



Investigate at least three potential predictors of the dependent variable and provide appropriate graphs / statistics to demonstrate the relationships.

In [17]:
```
# creating correlation heatmap to determine potential predictor variables
corr_mat = df4.corr()
f, ax = plt.subplots(figsize=(35, 25))
sns.heatmap(corr_mat, vmin=-1, vmax=1, annot=True, square=True, linewidths=.5, cmap='Br
plt.title('Correlation Heatmap')
```

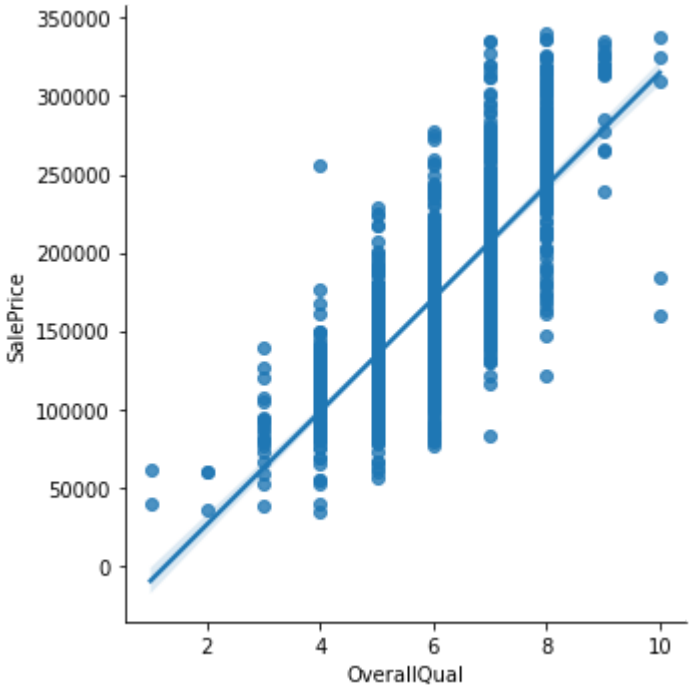
Out[17]: Text(0.5, 1.0, 'Correlation Heatmap')



The 3 variables with the highest correlation to SalesPrice are OverallQual (0.78), GrLivArea (0.66), and GarageCars (0.63). These are potential predictor variables for SalesPrice which is our dependent variable.

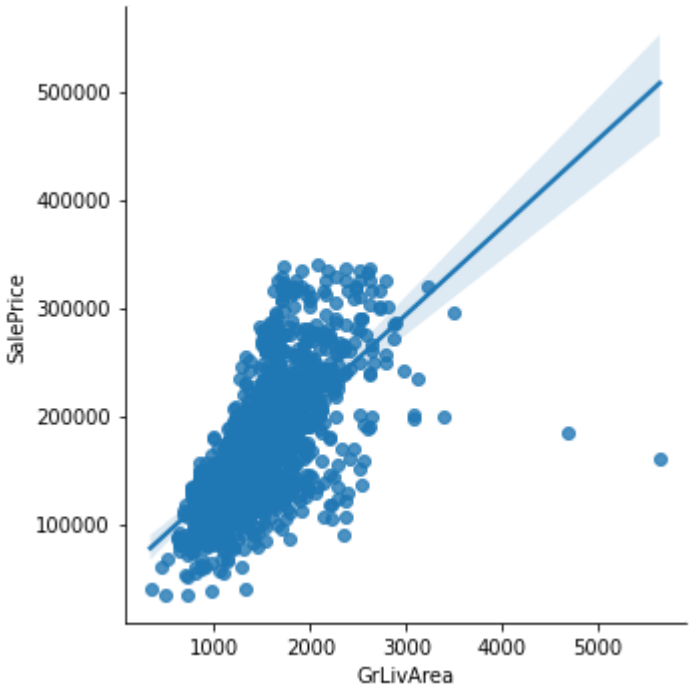
In [18]:
```
#sns.scatterplot(x='OverallQual', y='SalePrice', data=df4)
sns.lmplot(x='OverallQual', y='SalePrice', data=df4)
```

Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x7fb63373fa60>
```



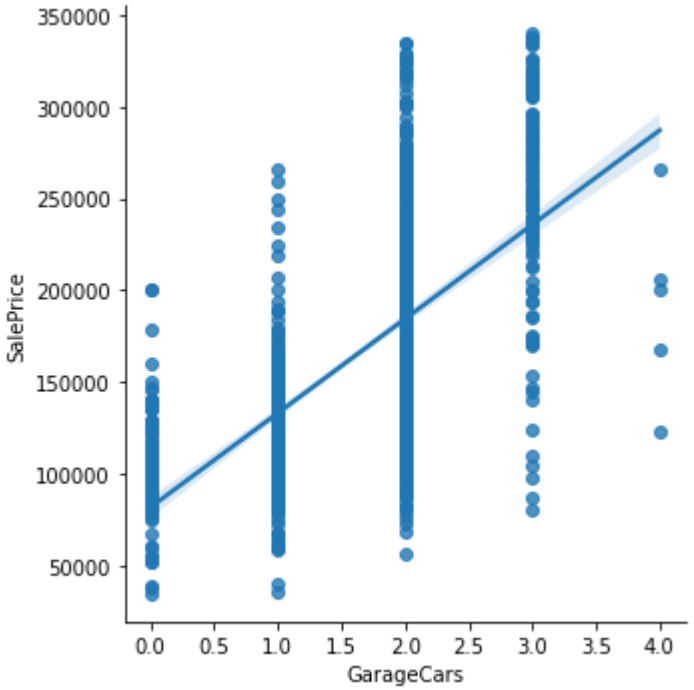
In [19]:
```
#sns.scatterplot(x="GrLivArea", y="SalePrice", data=df4)
sns.lmplot(x="GrLivArea", y="SalePrice", data=df4)
```

Out[19]: <seaborn.axisgrid.FacetGrid at 0x7fb633a6eb20>



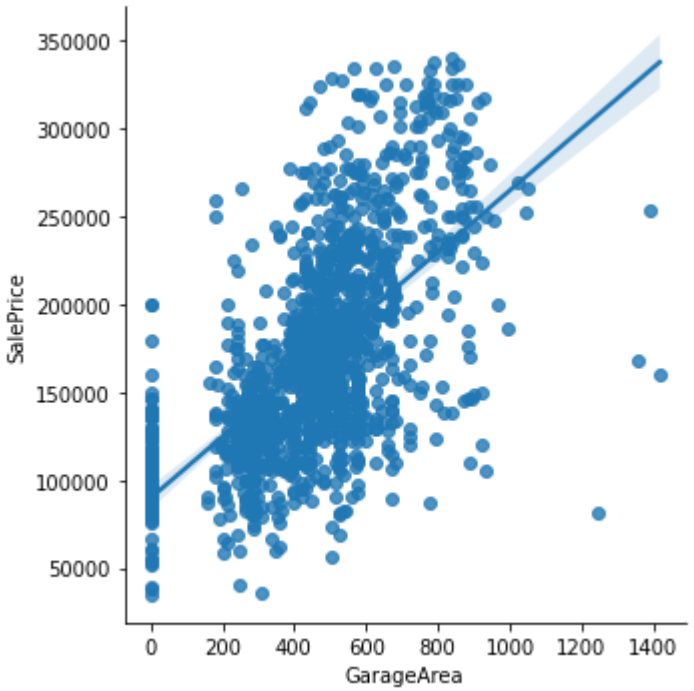
In [20]:
```
#sns.scatterplot(x="GarageCars", y="SalePrice", data=df4)
sns.lmplot(x="GarageCars", y="SalePrice", data=df4)
```

Out[20]: <seaborn.axisgrid.FacetGrid at 0x7fb6337ab6a0>



In [21]:
```
#sns.scatterplot(x="GarageArea", y="SalePrice", data=df4)
sns.lmplot(x="GarageArea", y="SalePrice", data=df4)
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x7fb633a6eeb0>



Engage in feature creation by splitting, merging, or otherwise generating a new predictor.

In [30]:
```
# sum 1st floor, 2nd floor, and basement square footage to get total square footage
sum_column = df4['1stFlrSF'] + df4['2ndFlrSF'] + df4['TotalBsmtSF']

# multiply total square footage by overall quality to generate new predictor variable q
mult_column = sum_column*df4['OverallQual']
```

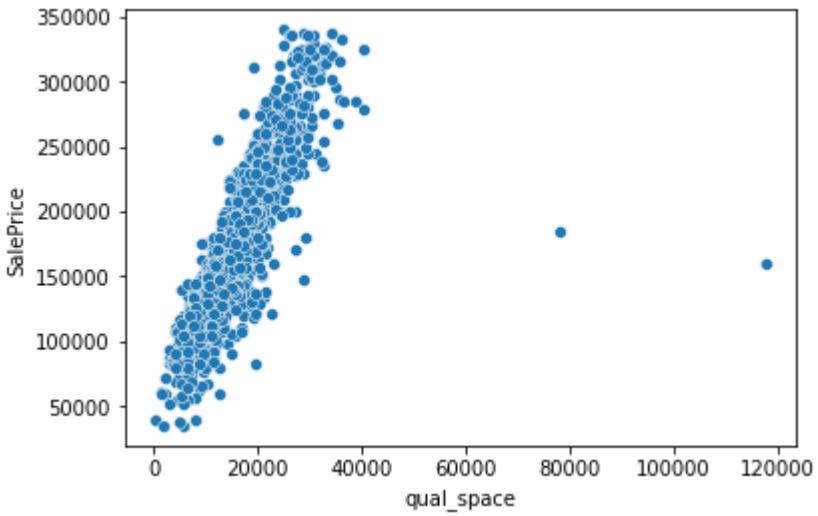
```
# add new predictor variables to dataframe
df4['tot_sq'] = sum_column
df4['qual_space'] = mult_column
```

```
/var/folders/pj/vbpn4qdj317glrhzx8jv6ggm0000gn/T/ipykernel_10929/3359187021.py:8: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df4['tot_sq'] = sum_column
/var/folders/pj/vbpn4qdj317glrhzx8jv6ggm0000gn/T/ipykernel_10929/3359187021.py:9: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df4['qual_space'] = mult_column
```

In [23]:

```
sns.scatterplot(x="qual_space", y="SalePrice", data=df4)
```

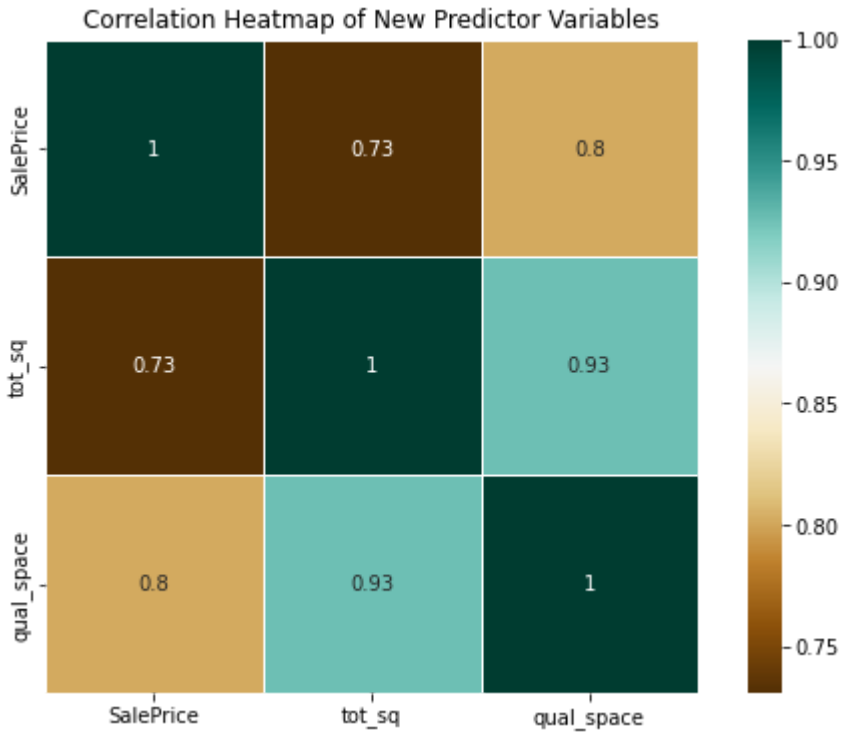
Out[23]: <AxesSubplot:xlabel='qual_space', ylabel='SalePrice'>



In [24]:

```
# setting the columns to correlate
columns = ['SalePrice', 'tot_sq', 'qual_space']
df_corr = df4[columns]
# running the correlation
df_corr.corr()

# setting up the heatmap
corrmat = df_corr.corr()

# set the figure size
f, ax = plt.subplots(figsize=(9, 6))

# pass the data and set the parameters
sns.heatmap(corrmat, vmax=1, square=True, annot=True, cmap='BrBG', linewidths=.5 )
plt.title('Correlation Heatmap of New Predictor Variables')

# images can be saved - default is .png
```

```
# https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.savefig.html
plt.savefig('Correlation Heatmap of New Predictor Variables')
```



Using the dependent variable, perform both min-max and standard scaling in Python.

In [25]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
mm_scaler = MinMaxScaler()
std_scaler = StandardScaler()

# reshape to SalePrice 2D array and perform min-max scaling
mmscaled_data = mm_scaler.fit_transform(df4['SalePrice'].values.reshape(-1,1))

print(mmscaled_data)
```

```
[[0.56946039]
 [0.48089224]
 [0.61866492]
 ...
 [0.75971789]
 [0.35173036]
 [0.36936198]]
```

In [26]:
```
# checking min and max
print(mmscaled_data.min())
print(mmscaled_data.max())
```

```
0.0
1.0
```

In [27]:
```
# reshape to SalePrice 2D array and perform standard scaling
stdscaled_data = std_scaler.fit_transform(df4['SalePrice'].values.reshape(-1,1))

print(stdscaled_data)
```

```
[[ 0.65346866]
 [ 0.19524104]
 [ 0.90803956]
 ...
 [ 1.63780948]
 [-0.47300758]
 [-0.38178634]]
```

In [28]:
```
# checking mean and standard deviation
print(stdscaled_data.mean())
print(stdscaled_data.std())
```

```
2.1440983340898438e-16
1.0
```

In [29]:
```
# add scaled sale price to dataframe
df4['MinMaxScaled_SalePrice'] = mmscaled_data
df4['StdScaled_SalePrice'] = stdscaled_data
```

```
/var/folders/pj/vbpn4qdj317glrhzx8jv6ggm0000gn/T/ipykernel_10929/647750776.py:2: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df4['MinMaxScaled_SalePrice'] = mmscaled_data
/var/folders/pj/vbpn4qdj317glrhzx8jv6ggm0000gn/T/ipykernel_10929/647750776.py:3: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df4['StdScaled_SalePrice'] = stdscaled_data
```

In [31]:
```
df4.info
```

Out[31]:
```
<bound method DataFrame.info of         Id  MSSubClass MSZoning  LotArea Street LotShape
LandContour  \
0        1          60       RL     8450   Pave      Reg         Lvl
1        2          20       RL     9600   Pave      Reg         Lvl
2        3          60       RL    11250   Pave      IR1         Lvl
3        4          70       RL     9550   Pave      IR1         Lvl
4        5          60       RL    14260   Pave      IR1         Lvl
...    ...         ...      ...      ...    ...      ...         ...
1455  1456          60       RL     7917   Pave      Reg         Lvl
1456  1457          20       RL    13175   Pave      Reg         Lvl
1457  1458          70       RL     9042   Pave      Reg         Lvl
1458  1459          20       RL     9717   Pave      Reg         Lvl
1459  1460          20       RL     9937   Pave      Reg         Lvl

     Utilities LotConfig LandSlope  ... MiscVal MoSold YrSold SaleType  \
0       AllPub    Inside       Gtl  ...       0      2   2008       WD
1       AllPub       FR2       Gtl  ...       0      5   2007       WD
2       AllPub    Inside       Gtl  ...       0      9   2008       WD
3       AllPub    Corner       Gtl  ...       0      2   2006       WD
4       AllPub       FR2       Gtl  ...       0     12   2008       WD
...        ...       ...       ...  ...     ...    ...    ...      ...
1455    AllPub    Inside       Gtl  ...       0      8   2007       WD
1456    AllPub    Inside       Gtl  ...       0      2   2010       WD
```

```
1457    AllPub    Inside       Gtl  ...    2500      5   2010       WD
1458    AllPub    Inside       Gtl  ...       0      4   2010       WD
1459    AllPub    Inside       Gtl  ...       0      6   2008       WD

     SaleCondition  SalePrice  tot_sq  qual_space  MinMaxScaled_SalePrice  \
0           Normal     208500    2566       17962                0.569460
1           Normal     181500    2524       15144                0.480892
2           Normal     223500    2706       18942                0.618665
3          Abnorml     140000    2473       17311                0.344760
4           Normal     250000    3343       26744                0.705593
...            ...        ...     ...         ...                     ...
1455        Normal     175000    2600       15600                0.459570
1456        Normal     210000    3615       21690                0.574381
1457        Normal     266500    3492       24444                0.759718
1458        Normal     142125    2156       10780                0.351730
1459        Normal     147500    2512       12560                0.369362

     StdScaled_SalePrice
0               0.653469
1               0.195241
2               0.908040
3              -0.509072
4               1.357781
...                  ...
1455            0.084927
1456            0.678926
1457            1.637809
1458           -0.473008
1459           -0.381786

[1396 rows x 67 columns]>
```

In [ ]: