```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import preprocessing
from scipy import stats
sns.set()
```

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

```
train.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	Mis
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	

5 rows × 81 columns

```
test.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	Poo
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...	120	0	I
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...	0	0	I
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...	0	0	I
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	...	0	0	I
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	...	144	0	I

5 rows × 80 columns

```
print(train.shape,test.shape)
```

```
(1460, 81) (1459, 80)
```

```
print(train.columns, test.columns)
```

```
Index(['Id', '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'],
      dtype='object') Index(['Id', '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'],
           dtype='object')
```

Start coding or generate with AI.

```
#EDA
```

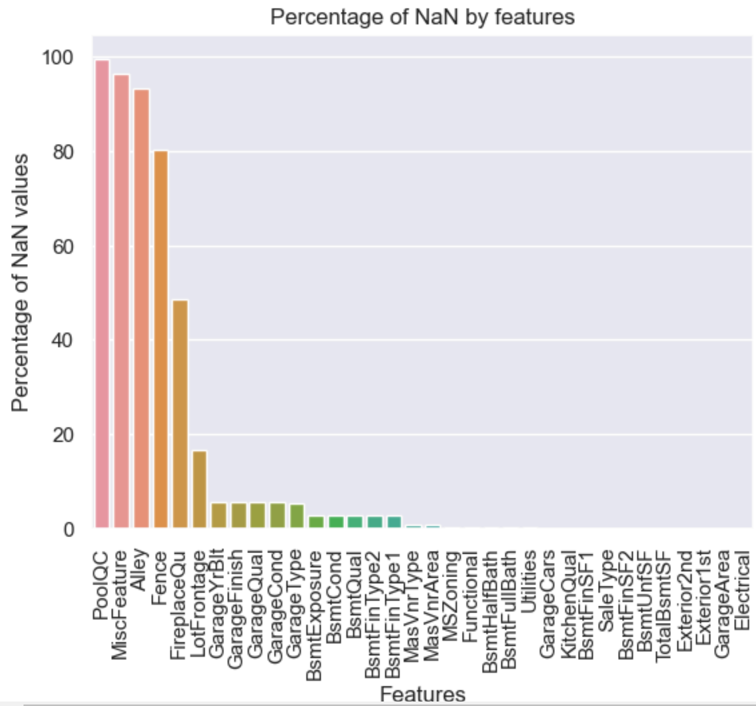
```
train_total_rows = train.shape[0]
test_total_rows = test.shape[0]

concat = pd.concat([train,test]).reset_index(drop=True)
concat.drop(['SalePrice'],axis=1,inplace=True)
print(train.shape)
print(test.shape)
print(concat.shape)
```

```
(1460, 81)
(1459, 80)
(2919, 80)
```

```
concat_na = (concat.isna().sum() / len(concat))*100
concat_na = concat_na.sort_values(ascending=False)
concat_na = concat_na.drop(concat_na[concat_na==0].index)
nan_values = pd.DataFrame({'NaN %':concat_na})
nan_values.head()
```

	NaN %
PoolQC	99.657417
MiscFeature	96.402878
Alley	93.216855
Fence	80.438506
FireplaceQu	48.646797

```
sns.barplot(data=nan_values,x=nan_values.index,y='NaN %')
plt.xlabel('Features')
plt.ylabel('Percentage of NaN values')
plt.title('Percentage of NaN by features')
plt.xticks(rotation=90)
plt.show()
```



```
concat_full = concat.copy()
concat_full['PoolQC'].fillna('None',inplace=True)
nan_values
```

	NaN %
PoolQC	99.657417
MiscFeature	96.402878
Alley	93.216855
Fence	80.438506
FireplaceQu	48.646797
LotFrontage	16.649538
GarageYrBlt	5.447071
GarageFinish	5.447071
GarageQual	5.447071
GarageCond	5.447071
GarageType	5.378554
BsmtExposure	2.809181
BsmtCond	2.809181
BsmtQual	2.774923
BsmtFinType2	2.740665
BsmtFinType1	2.706406
MasVnrType	0.822199
MasVnrArea	0.787941
MSZoning	0.137033
Functional	0.068517
BsmtHalfBath	0.068517
BsmtFullBath	0.068517
Utilities	0.068517
GarageCars	0.034258
KitchenQual	0.034258
BsmtFinSF1	0.034258
SaleType	0.034258
BsmtFinSF2	0.034258
BsmtUnfSF	0.034258
TotalBsmtSF	0.034258
Exterior2nd	0.034258
Exterior1st	0.034258
GarageArea	0.034258
Electrical	0.034258

```
concat_full['MiscFeature'].fillna('None',inplace=True)
concat_full['Alley'].fillna('None',inplace=True)
concat_full['Fence'].fillna('None',inplace=True)
concat_full['FireplaceQu'].fillna('None',inplace=True)


concat_full["LotFrontage"] = concat_full.groupby("Neighborhood")["LotFrontage"].transform(
    lambda x: x.fillna(x.median()))


for col in ('GarageType', 'GarageFinish', 'GarageQual', 'GarageCond'):
    concat_full[col].fillna('None', inplace=True)


for col in ('GarageYrBlt', 'GarageArea', 'GarageCars'):
    concat_full[col].fillna(0, inplace=True)
```

```
for col in ('BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2','MasVnrType',):
    concat_full[col].fillna('None', inplace=True)

for col in ('BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF','TotalBsmtSF', 'BsmtHalfBath', 'BsmtFullBath','MasVnrArea'):
    concat_full[col].fillna(0, inplace=True)
```

```
nan_values
```

	NaN %
PoolQC	99.657417
MiscFeature	96.402878
Alley	93.216855
Fence	80.438506
FireplaceQu	48.646797
LotFrontage	16.649538
GarageYrBlt	5.447071
GarageFinish	5.447071
GarageQual	5.447071
GarageCond	5.447071
GarageType	5.378554
BsmtExposure	2.809181
BsmtCond	2.809181
BsmtQual	2.774923
BsmtFinType2	2.740665
BsmtFinType1	2.706406
MasVnrType	0.822199
MasVnrArea	0.787941
MSZoning	0.137033
Functional	0.068517
BsmtHalfBath	0.068517
BsmtFullBath	0.068517
Utilities	0.068517
GarageCars	0.034258
KitchenQual	0.034258
BsmtFinSF1	0.034258
SaleType	0.034258
BsmtFinSF2	0.034258
BsmtUnfSF	0.034258
TotalBsmtSF	0.034258
Exterior2nd	0.034258
Exterior1st	0.034258
GarageArea	0.034258
Electrical	0.034258

```
concat_full
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	0
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	0
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	0
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	0
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2914	2915	160	RM	21.0	1936	Pave	NaN	Reg	Lvl	AllPub	...	0	0
2915	2916	160	RM	21.0	1894	Pave	NaN	Reg	Lvl	AllPub	...	0	0
2916	2917	20	RL	160.0	20000	Pave	NaN	Reg	Lvl	AllPub	...	0	0
2917	2918	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	AllPub	...	0	0
2918	2919	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	AllPub	...	0	0

2919 rows × 80 columns

```
for col in ('Electrical', 'Functional', 'Utilities','KitchenQual', 'Exterior1st', 'Exterior2nd','SaleType','MSZoning'):
    concat_full[col].fillna(concat_full[col].mode()[0], inplace=True)
```

```
concat_na = (concat_full.isna().sum() / len(concat_full))*100
concat_na = concat_na.sort_values(ascending=False)
concat_na = concat_na.drop(concat_na[concat_na==0].index)
nan_values = pd.DataFrame({'NaN %':concat_na})
nan_values.head()
```

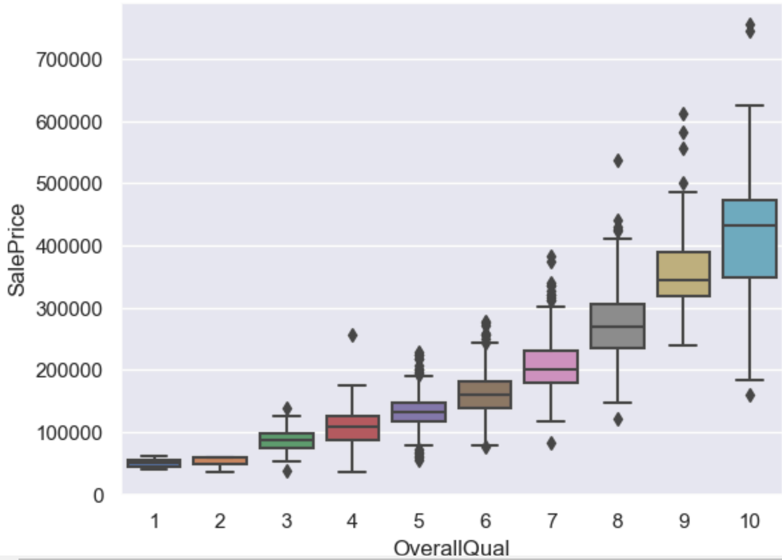
	NaN %
MiscFeature	96.402878
Alley	93.216855
Fence	80.438506
FireplaceQu	48.646797
GarageFinish	5.447071

```
train_full = concat_full[:train_total_rows]
train_full['SalePrice'] = train['SalePrice']
test_full = concat_full[train_total_rows:]
```

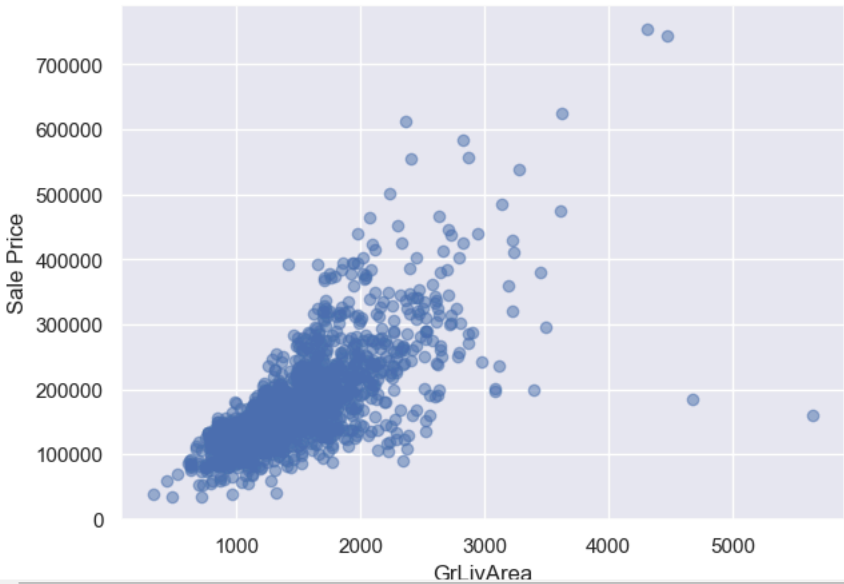
```
C:\Users\jasmi\AppData\Local\Temp\ipykernel_22196\1052480995.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  train_full['SalePrice'] = train['SalePrice']
```

```
#outliers
sns.boxplot(data=train_full,y='SalePrice',x='OverallQual')
```

```
<Axes: xlabel='OverallQual', ylabel='SalePrice'>
```

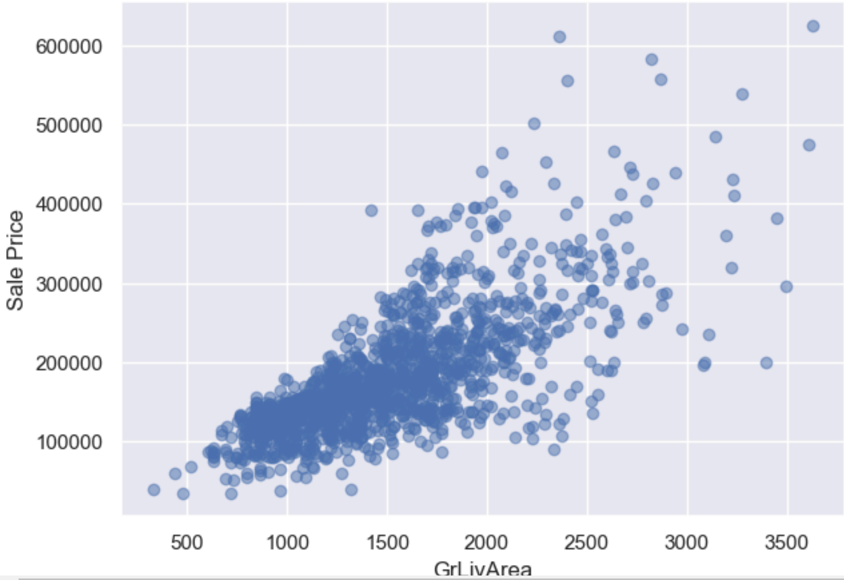



```
fig, ax = plt.subplots(figsize=(7,5))
ax.scatter(train_full['GrLivArea'],train_full['SalePrice'], alpha=0.5)
ax.set_ylabel('Sale Price')
ax.set_xlabel('GrLivArea')
plt.show()
```

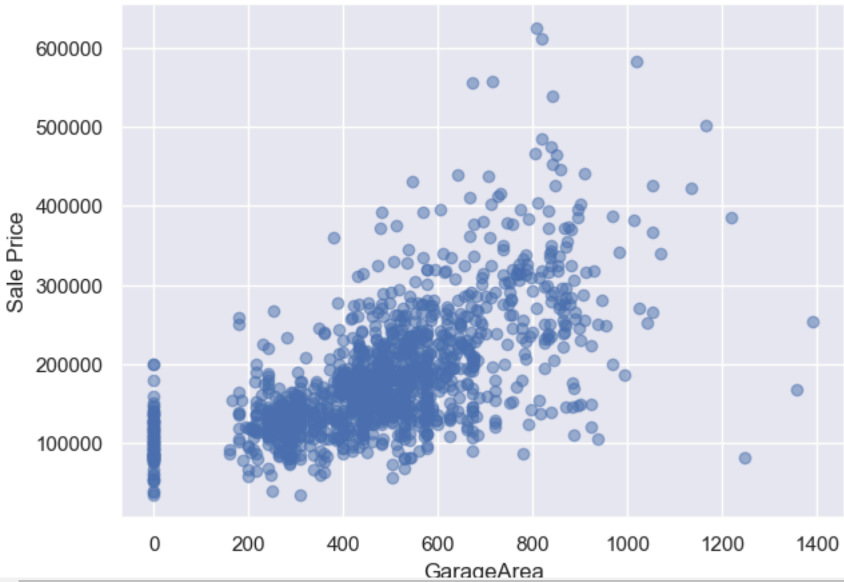



```
train_full = train_full.drop(train_full[train_full['GrLivArea'] > 4000].index)
```

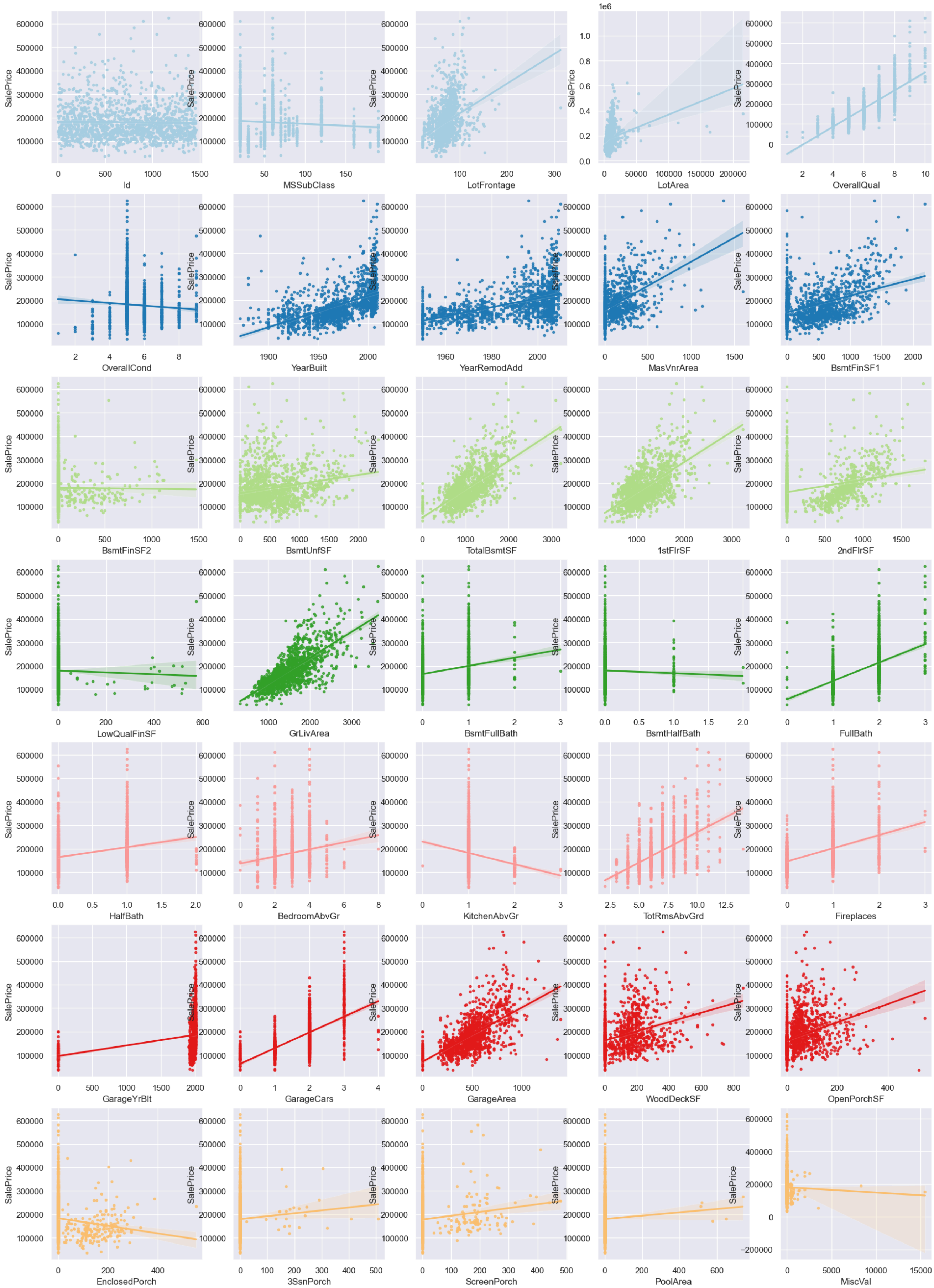
```
fig, ax = plt.subplots(figsize=(7,5))
ax.scatter(train_full['GrLivArea'],train_full['SalePrice'], alpha=0.5)
ax.set_ylabel('Sale Price')
ax.set_xlabel('GrLivArea')
plt.show()
```




```
fig, ax = plt.subplots(figsize=(7,5))
ax.scatter(train_full['GarageArea'],train_full['SalePrice'], alpha=0.5)
ax.set_ylabel('Sale Price')
ax.set_xlabel('GarageArea')
plt.show()
```

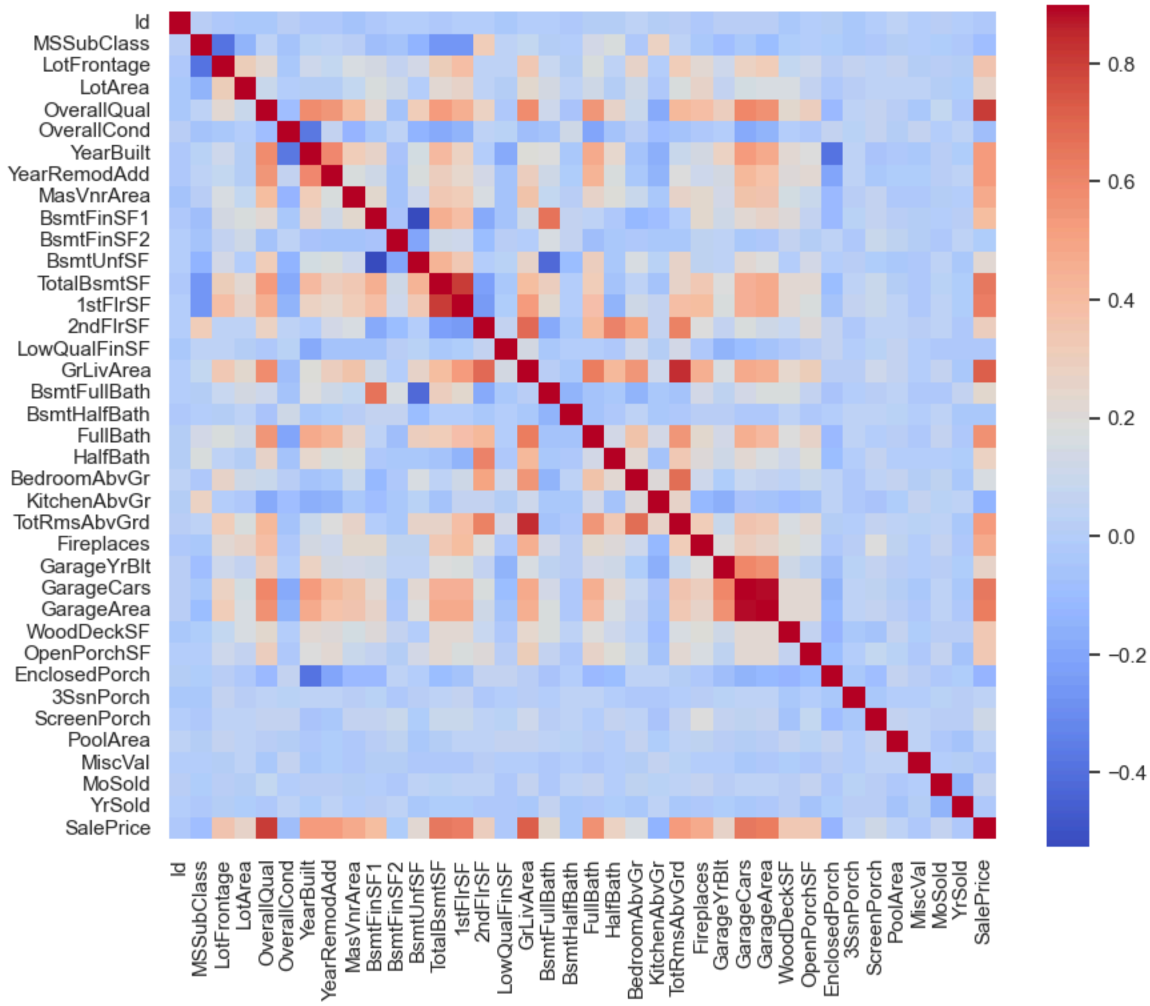



```
sns_rows = 7
sns_cols = 5
fig, axes = plt.subplots(sns_rows, sns_cols,figsize=(21,30))
palette= sns.color_palette("Paired", 10)

#train_full = train_full.drop(columns=['Id'])
num_features = train_full.dtypes[train_full.dtypes != "object"].index
num_list = list(num_features)

for num in range(0,sns_rows):
    for col in range(0, sns_cols):
        i = num * sns_cols + col
        if i < len(num_list):
            sns.regplot(x=num_list[i],y='SalePrice',
            data = train_full, ax = axes[num][col],
            color = palette[num],marker=".")
plt.show()
```


SalePrice
Id
MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal

```
corrmat = train_full.corr()
with sns.axes_style("white"):

    f, ax = plt.subplots(figsize=(10, 10))
    sns.heatmap(corrmat, ax=ax, cbar_kws={"shrink": .82},vmax=.9, cmap='coolwarm', square=True)
```

```
C:\Users\jasmi\AppData\Local\Temp\ipykernel_22196\123730058.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr
  corrmat = train_full.corr()
```



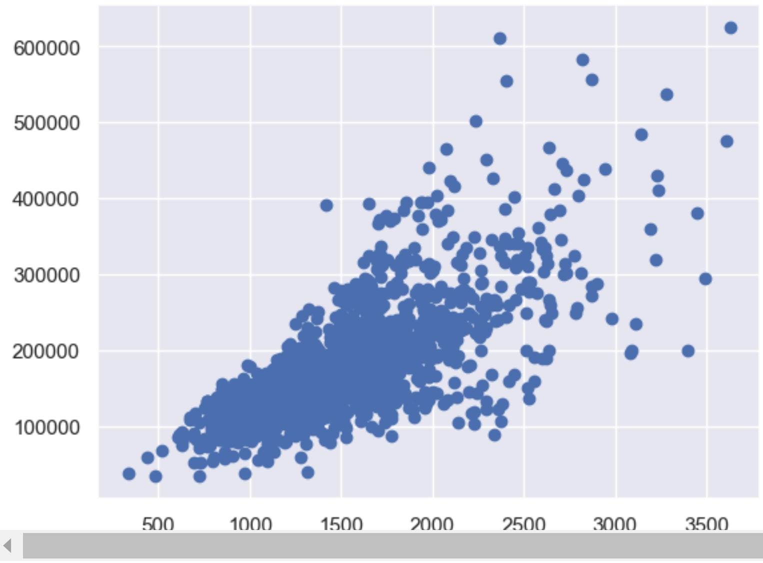

```
from sklearn.preprocessing import LabelEncoder
data = train_full.copy()
categorical_features= data.select_dtypes(include=['object']).copy()
number=[len(data[features].unique()) for features in categorical_features]
data_tuples = list(zip(categorical_features,number))
categorical_data= pd.DataFrame(data_tuples, columns=['Features','Number of distinct values '])
categorical_data
```

	Features	Number of distinct values
0	MSZoning	5
1	Street	2
2	Alley	3
3	LotShape	4
4	LandContour	4
5	Utilities	2
6	LotConfig	5
7	LandSlope	3
8	Neighborhood	25
9	Condition1	9
10	Condition2	8
11	BldgType	5
12	HouseStyle	8
13	RoofStyle	6
14	RoofMatl	7
15	Exterior1st	15
16	Exterior2nd	16
17	MasVnrType	4
18	ExterQual	4
19	ExterCond	5
20	Foundation	6
21	BsmtQual	5
22	BsmtCond	5
23	BsmtExposure	5
24	BsmtFinType1	7
25	BsmtFinType2	7
26	Heating	6
27	HeatingQC	5
28	CentralAir	2
29	Electrical	5
30	KitchenQual	4
31	Functional	7
32	FireplaceQu	6
33	GarageType	7
34	GarageFinish	4
35	GarageQual	6
36	GarageCond	6
37	PavedDrive	3
38	PoolQC	4
39	Fence	5
40	MiscFeature	5
41	SaleType	9
42	SaleCondition	6

```
for cat in categorical_features:
    label_encoder = LabelEncoder()
    label_encoder.fit(list(data[cat].values))
    data[cat] = label_encoder.transform(list(data[cat].values))
training_data=data.copy()

data = test_full.copy()
categorical_features = [features for features in data.columns if data[features].dtype == 'O']

for cat in categorical_features:
    label_encoder = LabelEncoder()
    label_encoder.fit(list(data[cat].values))
    data[cat] = label_encoder.transform(list(data[cat].values))
test_data=data.copy()


#Finding assumptions
def correlatedFeatures(correlation_data, threshold):
    feature=[]
    value=[]
    for i,index in enumerate(correlation_data.index):
        if abs(correlation_data[index]) > threshold:
            feature.append(index)
            value.append(correlation_data[index])
    df = pd.DataFrame(data=value,index=feature,columns=['Corr Value'])
    return df


corr_check = correlatedFeatures(training_data.corr()['SalePrice'],0.5)
corr_check.sort_values(by='Corr Value', ascending=False)
```

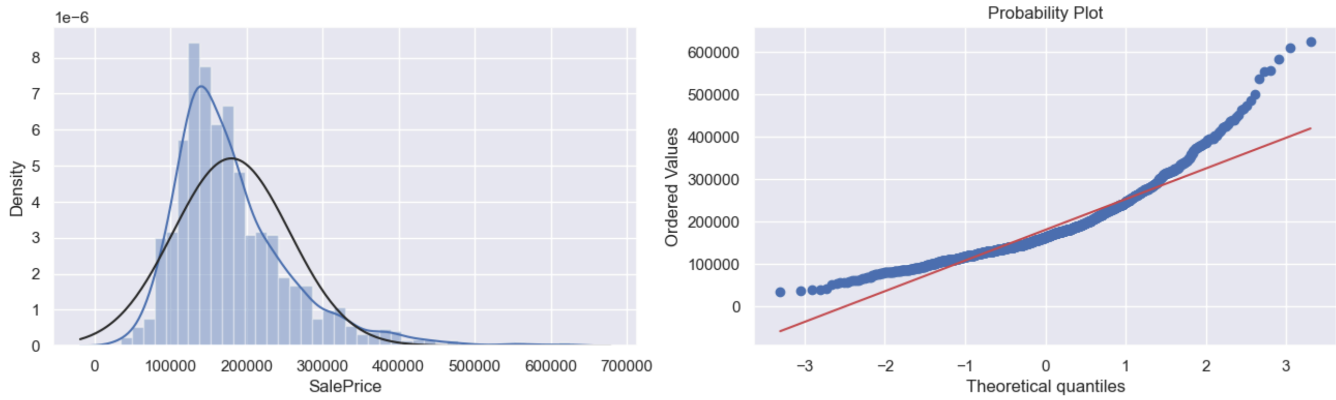
	Corr Value
SalePrice	1.000000
OverallQual	0.800858
GrLivArea	0.720516
GarageCars	0.649256
TotalBsmtSF	0.646584
GarageArea	0.636964
1stFlrSF	0.625235
FullBath	0.559048
TotRmsAbvGrd	0.537462
YearBuilt	0.535279
YearRemodAdd	0.521428
GarageFinish	-0.556808
KitchenQual	-0.589238
BsmtQual	-0.598144
ExterQual	-0.647479

```
plt.scatter(training_data.GrLivArea, training_data.SalePrice);
plt.show()
```




```
from scipy.stats import norm, skew
print("Skewness: %f" % training_data['SalePrice'].skew())
print("Kurtosis: %f" % training_data['SalePrice'].kurt())
print()
fig, ax = plt.subplots(1,2, figsize=(16,4))
sns.distplot(training_data['SalePrice'] , fit=norm, ax=ax[0])

res = stats.probplot(training_data['SalePrice'],plot=ax[1])
plt.show()
```

```
Skewness: 1.565959
Kurtosis: 3.885283

C:\Users\jasmi\AppData\Local\Temp\ipykernel_22196\2009565569.py:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(training_data['SalePrice'] , fit=norm, ax=ax[0])
```




```
training_data['SalePrice'] = np.log(training_data['SalePrice'])


print("Skewness: %f" % training_data['SalePrice'].skew())
print("Kurtosis: %f" % training_data['SalePrice'].kurt())
print()
fig, ax = plt.subplots(1,2, figsize=(16,4))
sns.distplot(training_data['SalePrice'] , fit=norm, ax=ax[0])

res = stats.probplot(training_data['SalePrice'],plot=ax[1])
plt.show()
```