Question 1

1. **What is the optimal value of alpha for ridge and lasso regression?**

```
ridge = Ridge(alpha = 8.0)

ridge.fit(x_train, y_train)

y_pred_r_train = ridge.predict(x_train)
y_pred_r_test = ridge.predict(x_test)

metric_ri = displayR2_RSS_MSE(y_pred_r_train, y_pred_r_test)
```

```
R2_Train : 0.9312879006573223
R2_Test : 0.8747429984254674

RSS_Train : 1.166500154601748
RSS_Test : 0.958936176063856

MSE_Train : 0.0011425074971613595
MSE_Test : 0.0021843648657491025
```

```
lasso = Lasso(alpha=0.0004)

lasso.fit(x_train, y_train)

y_pred_l_train = lasso.predict(x_train)
y_pred_l_test = lasso.predict(x_test)

metric_la = displayR2_RSS_MSE(y_pred_l_train,y_pred_l_test)
```

```
R2_Train : 0.8993421710545795
R2_Test : 0.869239004979881

RSS_Train : 1.7088311105316556
RSS_Test : 1.00107336888697

MSE_Train : 0.0016736837517450105
MSE_Test : 0.0022803493596 51412
```

Optimal value of alpha for Ridge Regression: "8.0"

Optimal value of alpha for Lasso Regression: "0.0004"

2. **What will be the changes in the model if you choose double the value of alpha for both ridge and lasso?**

```
ridge2x = Ridge(alpha=16)

ridge2x.fit(x_train, y_train)

y_pred_r_train = ridge2x.predict(x_train)
y_pred_r_test = ridge2x.predict(x_test)

metric_ri_2x = displayR2_RSS_MSE(y_pred_r_train, y_pred_r_test)
```

```
R2_Train : 0.9175274236701499
R2_Test : 0.8676889833657089

RSS_Train : 1.400106734614369
RSS_Test : 1.012939945452162

MSE_Train : 0.0013713092405625554
MSE_Test : 0.0023073802857680225
```

```
lasso2x = Lasso(alpha=0.0008)

lasso2x.fit(x_train, y_train)

y_pred_l_train = lasso2x.predict(x_train)
y_pred_l_test = lasso2x.predict(x_test)

metric_la_2x = displayR2_RSS_MSE(y_pred_l_train,y_pred_l_test)
```

```
R2_Train : 0.872621519129784б
R2_Test : 0.8409485262973453

RSS_Train : 2.1624578356573863
RSS_Test : 1.217658175371454

MSE_Train : 0.002117980250399007З
MSE_Test : 0.0027737088277254075
```

We observe that overall accuracy of a model in both Ridge and lasso regression decrease slightly.

3. **What will be the most important predictor variables after the change is implemented?**

```
In [63]: betas['Ridge2x'] = ridge2x.coef_
         betas['Lasso2x'] = lasso2x.coef_

In [64]: betas["Ridge2x"].sort_values()[-1:]

Out[64]: GrLivArea    0.047598
         Name: Ridge2x, dtype: float64

In [65]: betas["Ridge2x"].sort_values()[:1]

Out[65]: OverallCond_FA   -0.02702
         Name: Ridge2x, dtype: float64

In [66]: betas["Lasso2x"].sort_values()[-1:]

Out[66]: GrLivArea    0.30607
         Name: Lasso2x, dtype: float64

In [67]: betas["Lasso2x"].sort_values()[:1]

Out[67]: FireplaceQu_NA   -0.027481
         Name: Lasso2x, dtype: float64
```

The most important predictor variable after the change is implemented is "GrLivArea" (Above grade (ground) living area in square feet).

Question 2

You have determined the optimal value of lambda for ridge and lasso regression during the assignment. Now, which one will you choose to apply and why?

```
In [69]: final_metric
```

Out[69]:

	Metric	Linear Regression	Ridge Regression	Lasso Regression	Ridge 2x	Lasso 2x
0	R2 Score (Train)	9.744735e-01	0.931288	0.899342	0.917527	0.872622
1	R2 Score (Test)	-1.889971e+23	0.874743	0.869239	0.867689	0.840949
2	RSS (Train)	4.333533e-01	1.166500	1.708831	1.400107	2.162458
3	RSS (Test)	1.446914e+24	0.958936	1.001073	1.012940	1.217658
4	MSE (Train)	4.244400e-04	0.001143	0.001674	0.001371	0.002118
5	MSE (Test)	3.295932e+21	0.002184	0.002280	0.002307	0.002774

Note: Here we are comparing **Ridge Regression** column with **Lasso Regression** column.

If take the closer look at the matrix table Ridge has slightly more accurate in both (Train and Test data), But lasso has feature elimination feature by doing coefficient zero. Means lasso is taking very less features to predict the data which also means it's a simpler model than ridge. So, I will go with lasso regression if there no specific requirement from business that specific features should be there in the model.

Question 3

After building the model, you realised that the five most important predictor variables in the lasso model are not available in the incoming data. You will now have to create another model excluding the five most important predictor variables. Which are the five most important predictor variables now?

Step 1: Getting 5 most important predictor variables and update new feature list by removing them.

```
: betas["Lasso"].sort_values()[:5]
```

```
: OverallCond_FA        -0.040238
  MSSubClass_30         -0.027165
  BsmtExposure_NA       -0.022992
  FireplaceQu_NA        -0.021503
  Neighborhood_Edwards  -0.020686
  Name: Lasso, dtype: float64
```

```
: betas["Lasso"].sort_values()[-5:]
```

```
: OverallQual_VG    0.046114
  FullBath          0.047523
  GarageCars        0.076874
  OverallQual_EX    0.078985
  GrLivArea         0.333503
  Name: Lasso, dtype: float64
```

```
: currentTop5var = ["GrLivArea","OverallQual_VG","OverallQual_EX","GarageCars","FullBath"]
```

```
: updatedCols = list(set(list(x_train.columns)).difference(set(currentTop5var)))
  len(updatedCols)
```

```
: 560
```

Step 2: Building model with new feature list.

```
: lassoQ3 = Lasso(alpha=0.0004)

  lassoQ3.fit(x_train[updatedCols], y_train)

  y_pred_l_train = lassoQ3.predict(x_train[updatedCols])
  y_pred_l_test = lassoQ3.predict(x_test[updatedCols])

  metric_la_Q3 = displayR2_RSS_MSE(y_pred_l_train,y_pred_l_test)
```

```
R2_Train : 0.8916128912830926
R2_Test : 0.8556692150379317

RSS_Train : 1.8400482634734447
RSS_Test : 1.1049602759129198

MSE_Train : 0.0018022020210317774
MSE_Test : 0.00251699379479025
```

Step 3: Getting 5 most important predictor variables.

```
Q3 = pd.Series(lassoQ3.coef_,index=updatedCols)
```

```
Q3.sort_values()[:5]
```

```
PoolQC_Gd              -0.059595
OverallCond_FA         -0.040961
MSSubClass_30          -0.028432
Neighborhood_Edwards   -0.027849
KitchenQual_TA         -0.026977
dtype: float64
```

```
Q3.sort_values()[-5:]
```

```
Neighborhood_NridgHt    0.043402
TotRmsAbvGrd            0.068355
GarageArea              0.084342
2ndFlrSF                0.130429
1stFlrSF                0.271785
dtype: float64
```

According to above data after removing earlier top 5 variables the current 5 most important variables are:

1. **"1stFlrSF" (First Floor in square feet)**
2. **"2ndFlrSF" (Second floor in square feet)**
3. **"GarageArea" (Size of garage in square feet)**
4. **"TotRmsAbvGrd" (Total rooms above grade)**
5. **"PoolQC_Gd" (Pool quality: Good)**

Question 4

1. **How can you make sure that a model is robust and generalisable?**

Robust can be achieve by decreasing bias and generalisation can be achieved by decreasing variance. Since bias and variance are inversely proportionate to each other we can achieve robustness and generalisation by balancing them, and we can do that by regularisation.

2. **What are the implications of the same for the accuracy of the model and why?**

Implications are as follows: When there is a high variance and Low bias, model get overfitted and in overfitted model we get high accuracy on training data (Seen Data) but very low accuracy on test data (Unseen Data) means there is huge difference between of train and test accuracy which result to failure of a model.