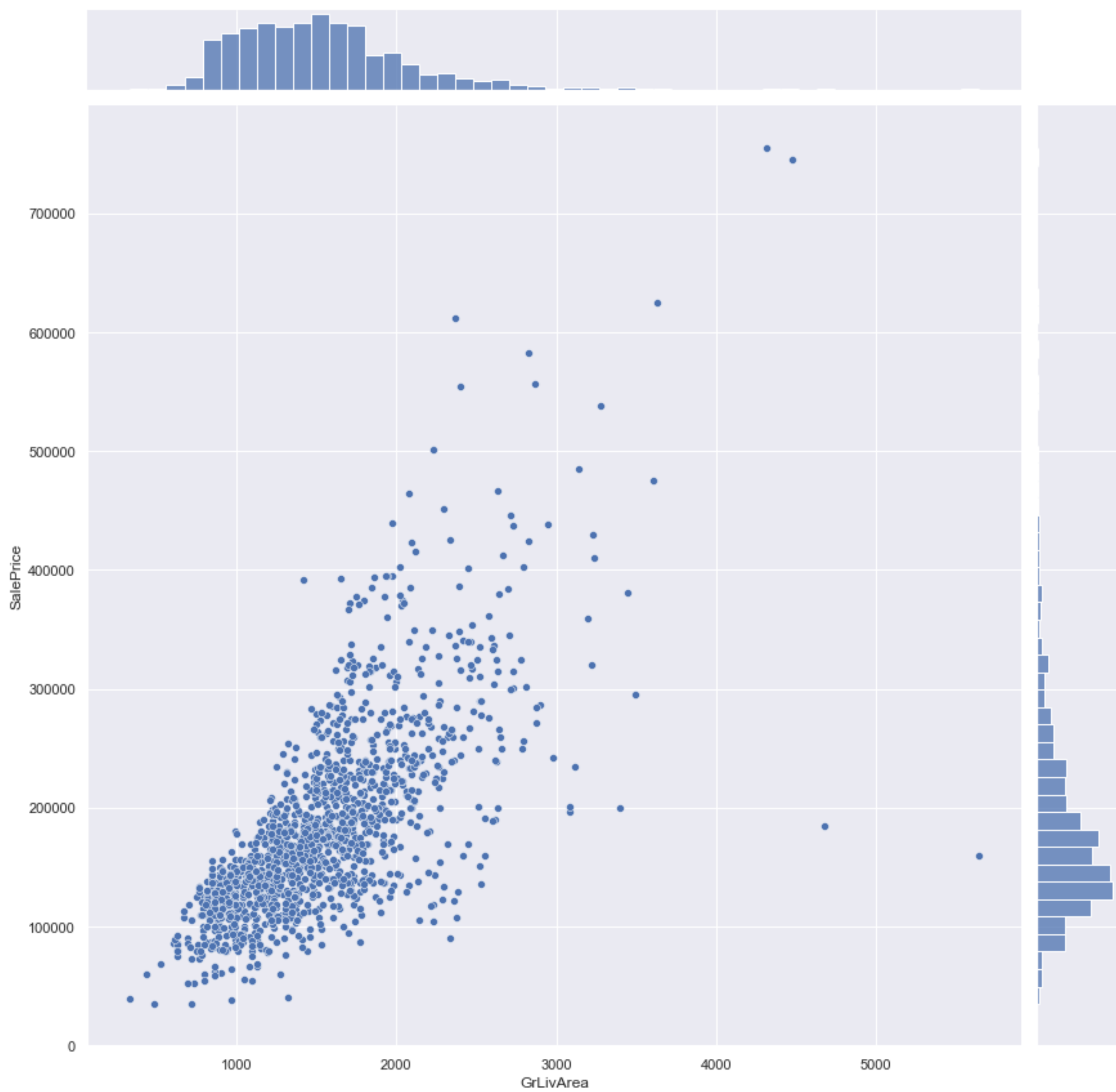
In [1]:
```
#basic libraries
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
sb.set()
```

Problem 1

1a) Plot SalePrice against GrLivArea using any appropriate bivariate plot to note the strong linear relationship.

In [2]:
```
houseData = pd.read_csv('train.csv')
SalePrice_GrLivArea = pd.DataFrame(houseData[['SalePrice', 'GrLivArea']]
sb.jointplot(data=SalePrice_GrLivArea, x="GrLivArea",y="SalePrice", ratio
```

Out[2]: <seaborn.axisgrid.JointGrid at 0x286e5b4ce50>



1b) Print the correlation coefficient between these two variables to get a numerical evidence of the relationship.

In [3]:
```
print(SalePrice_GrLivArea.corr())
```

```
           SalePrice  GrLivArea
SalePrice   1.000000   0.708624
GrLivArea   0.708624   1.000000
```

1c) Import Linear Regression model from Scikit-Learn : from sklearn.linear_model import LinearRegression

In [4]:
```
# Import LinearRegression model from Scikit-Learn
from sklearn.linear_model import LinearRegression
# other important models and functions to be used later
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

1d) Partition the dataset houseData into two “random” portions : Train Data (1100 rows) and Test Data (360 rows).

In [5]:
```
SalePrice = pd.DataFrame(SalePrice_GrLivArea['SalePrice'])
GrLivArea = pd.DataFrame(SalePrice_GrLivArea['GrLivArea'])
# Split the Dataset into Train and Test
SP_train, SP_test, GL_train, GL_test = train_test_split(SalePrice, GrLiv
```

1e) Training : Fit a Linear Regression model on the Train Dataset to predict or estimate SalePrice using GrLivArea

In [6]:
```
linreg = LinearRegression()         # create the linear regression objec
linreg.fit(GL_train, SP_train)      # train the linear regression model
```

Out[6]:
```
LinearRegression()
```

1f) Print the coefficients of the Linear Regression model you just fit, and plot the regression line on a scatterplot

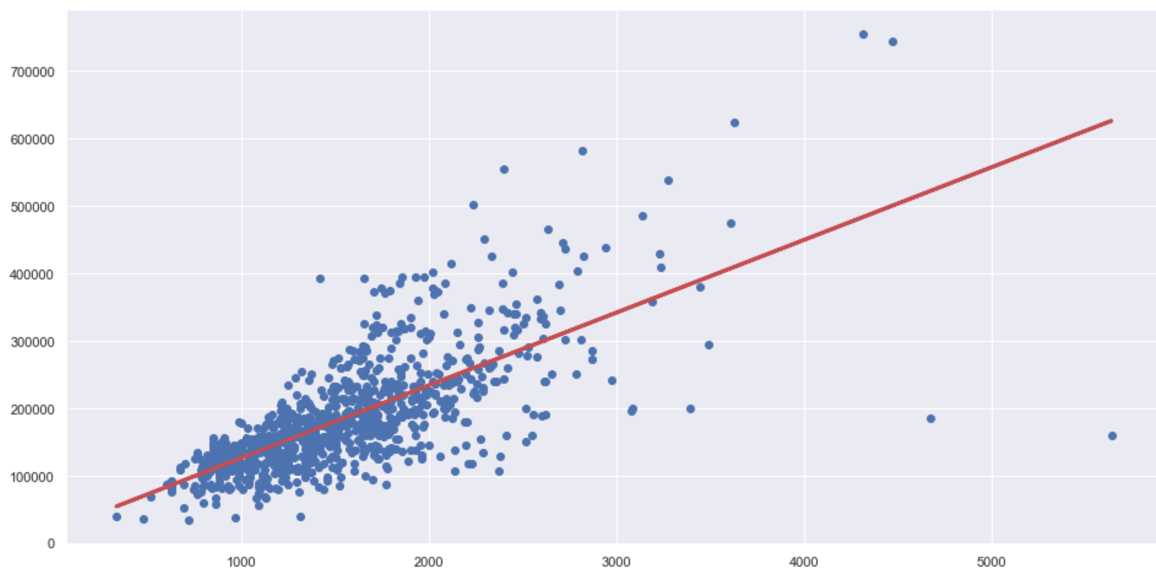
In [7]:

```
print('Coefficients of Regression \t: a = ', linreg.coef_)
print('Intercept of Regression \t: b = ', linreg.intercept_)
print()

# Formula for the Regression line
regline_GL = GL_train
regline_SP = linreg.intercept_ + linreg.coef_ * GL_train

# Plot the Linear Regression line
f = plt.figure(figsize=(16, 8))
plt.scatter(GL_train, SP_train)
plt.plot(regline_GL, regline_SP, 'r-', linewidth = 3)
plt.show()
```

```
Coefficients of Regression      : a =  [[107.79645878]]
Intercept of Regression         : b =  [18282.69458726]
```



1g) Print Explained Variance (R^2) and Mean Squared Error (MSE) on Train Data to check Goodness of Fit of model.

In [8]:

```
# Explained Variance (R^2)
print("Explained Variance (R^2) in Train Data \t:", linreg.score(GL_trai

# Predict SP values corresponding to SP Train
SP_train_pred = linreg.predict(GL_train)

# Mean Squared Error (MSE)
def mean_sq_err(actual, predicted):
    '''Returns the Mean Squared Error of actual and predicted values'''
    return np.mean(np.square(np.array(actual) - np.array(predicted)))


mse = mean_sq_err(SP_train, SP_train_pred)
print("Mean Squared Error (MSE) in Train Data \t:", mse)
print("Root Mean Squared Error (RMSE) in Train Data \t:", np.sqrt(mse))
```

```
Explained Variance (R^2) in Train Data  : 0.5132790199109318
Mean Squared Error (MSE) in Train Data  : 3330222833.9730177
Root Mean Squared Error (RMSE) in Train Data    : 57708.0829171531
```

1h) Predict SalePrice in case of Test Data using the Linear Regression model and the predictor variable GrLivArea

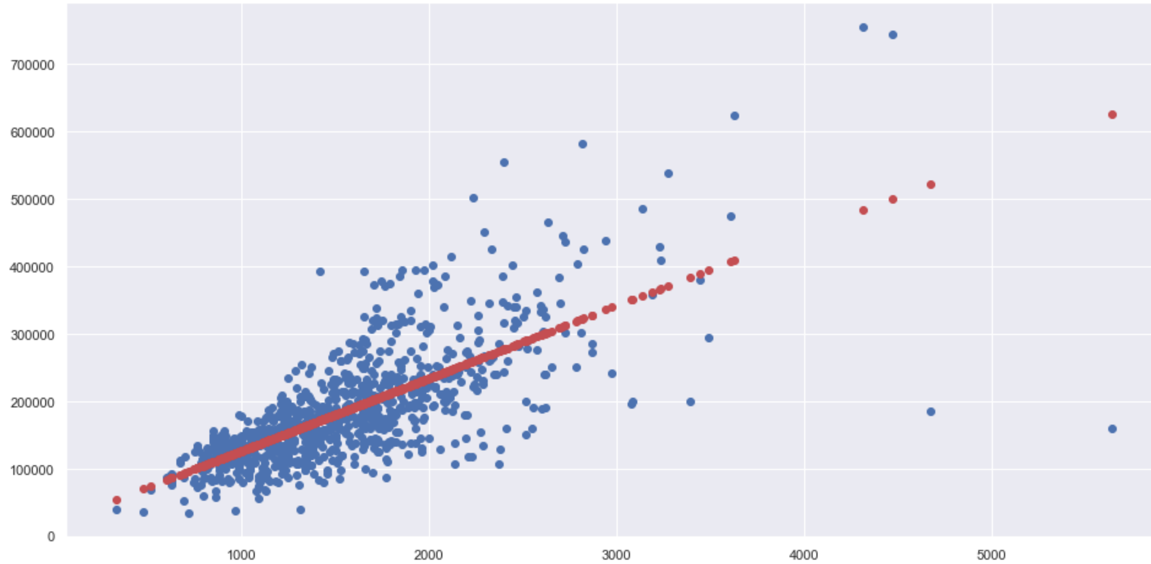
In [9]:

```
# Predict Response corresponding to Predictors
SP_train_pred = linreg.predict(GL_train)
SP_test_pred = linreg.predict(GL_test)
```

1i) Plot the predictions on a Scatterplot of GrLivArea and SalePrice in the Test Data to visualize model accuracy

In [10]:

```python
# Plot the Linear Regression Line
f = plt.figure(figsize=(16, 8))
plt.scatter(GL_train, SP_train)
plt.scatter(GL_train, SP_train_pred, color = "r")
plt.show()
```



1j) Print the Mean Squared Error (MSE) on Test Data to check Goodness of Fit of model, compared to the Training

In [11]:

```python
# Explained Variance (R^2)
print("Explained Variance (R^2) in Test Data \t:", linreg.score(GL_test,

# Mean Squared Error (MSE)
def mean_sq_err(actual, predicted):
    '''Returns the Mean Squared Error of actual and predicted values'''
    return np.mean(np.square(np.array(actual) - np.array(predicted)))

mse = mean_sq_err(SP_test, SP_test_pred)
print("Mean Squared Error (MSE) in Test Data \t: ", mse)
print("Root Mean Squared Error (RMSE) in Test Data \t:", np.sqrt(mse))
```

```
Explained Variance (R^2) in Test Data 	: 0.46759548318681243
Mean Squared Error (MSE) in Test Data 	:  2750699683.5151663
Root Mean Squared Error (RMSE) in Test Data 	: 52447.113204781504
```

Problem 2 : Predicting SalePrice using Other Variables

Perform all the above steps on “SalePrice” against each of the variables “LotArea”, “TotalBsmtSF”, “GarageArea” oneby-oneto perform individual Linear Regressions and obtain individual univariate Linear Regression Models in each case

SalePrice AGAINST LotArea

In [12]:

```
houseData = pd.read_csv('train.csv')
SalePrice_LotArea = pd.DataFrame(houseData[['SalePrice', 'LotArea']])
sb.jointplot(data=SalePrice_LotArea, x="LotArea",y="SalePrice", ratio =1(

print(SalePrice_LotArea.corr())

# Import LinearRegression model from Scikit-Learn
from sklearn.linear_model import LinearRegression
# other important models and functions to be used later
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

SalePrice = pd.DataFrame(SalePrice_LotArea['SalePrice'])
LotArea = pd.DataFrame(SalePrice_LotArea['LotArea'])
# Split the Dataset into Train and Test
SP_train, SP_test, LA_train, LA_test = train_test_split(SalePrice, LotAr(

linreg = LinearRegression()         # create the linear regression object
linreg.fit(LA_train, SP_train)      # train the linear regression model

print('Coefficients of Regression \t: a = ', linreg.coef_)
print('Intercept of Regression \t: b = ', linreg.intercept_)
print()

# Formula for the Regression line
regline_LA = LA_train
regline_SP = linreg.intercept_ + linreg.coef_ * LA_train

# Plot the Linear Regression line
f = plt.figure(figsize=(16, 8))
plt.scatter(LA_train, SP_train)
plt.plot(regline_LA, regline_SP, 'r-', linewidth = 3)
plt.show()

# Explained Variance (R^2)
print("Explained Variance (R^2) in Train Data \t:", linreg.score(LA_trai(

# Predict SP values corresponding to SP Train
SP_train_pred = linreg.predict(LA_train)

# Mean Squared Error (MSE)
def mean_sq_err(actual, predicted):
    '''Returns the Mean Squared Error of actual and predicted values'''
    return np.mean(np.square(np.array(actual) - np.array(predicted)))


mse = mean_sq_err(SP_train, SP_train_pred)
print("Mean Squared Error (MSE) in Train Data \t:", mse)
print("Root Mean Squared Error (RMSE) in Train Data \t:", np.sqrt(mse))

# Predict Response corresponding to Predictors
SP_train_pred = linreg.predict(LA_train)
SP_test_pred = linreg.predict(LA_test)

# Plot the Linear Regression line
f = plt.figure(figsize=(16, 8))
```

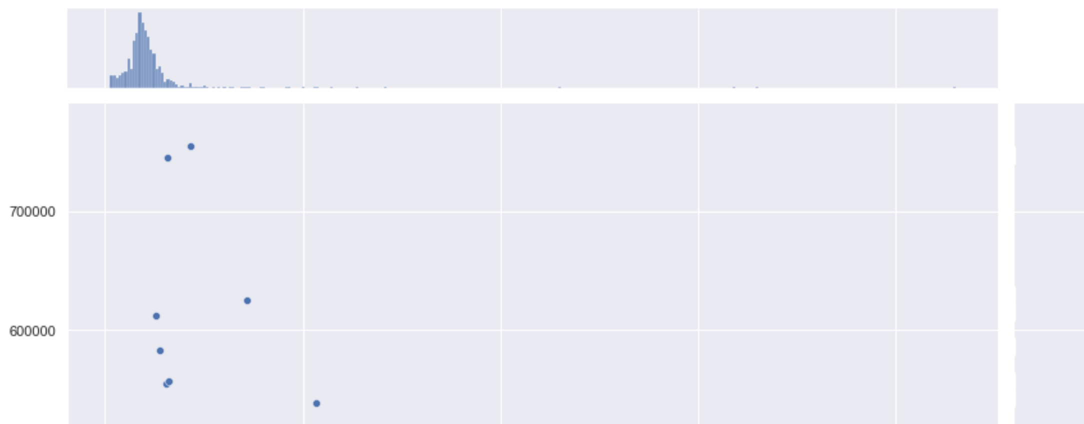
```python
plt.scatter(LA_train, SP_train)
plt.scatter(LA_train, SP_train_pred, color = "r")
plt.show()

# Explained Variance (R^2)
print("Explained Variance (R^2) in Test Data \t:", linreg.score(LA_test,

# Mean Squared Error (MSE)
def mean_sq_err(actual, predicted):
    '''Returns the Mean Squared Error of actual and predicted values'''
    return np.mean(np.square(np.array(actual) - np.array(predicted)))

mse = mean_sq_err(SP_test, SP_test_pred)
print("Mean Squared Error (MSE) in Test Data \t: ", mse)
print("Root Mean Squared Error (RMSE) in Test Data \t:", np.sqrt(mse))
```

```
           SalePrice   LotArea
SalePrice   1.000000  0.263843
LotArea     0.263843  1.000000
Coefficients of Regression      : a =  [[2.60125751]]
Intercept of Regression         : b =  [154036.03147069]
```



SalePrice AGAINST TotalBsmtSF

In [13]:

```
houseData = pd.read_csv('train.csv')
SalePrice_TotalBsmtSF = pd.DataFrame(houseData[['SalePrice', 'TotalBsmtSl
sb.jointplot(data=SalePrice_TotalBsmtSF, x="TotalBsmtSF",y="SalePrice", 

print(SalePrice_TotalBsmtSF.corr())

# Import LinearRegression model from Scikit-Learn
from sklearn.linear_model import LinearRegression
# other important models and functions to be used later
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

SalePrice = pd.DataFrame(SalePrice_TotalBsmtSF['SalePrice'])
TotalBsmtSF = pd.DataFrame(SalePrice_TotalBsmtSF['TotalBsmtSF'])
# Split the Dataset into Train and Test
SP_train, SP_test, BS_train, BS_test = train_test_split(SalePrice, Totall

linreg = LinearRegression()         # create the linear regression objec
linreg.fit(BS_train, SP_train)      # train the linear regression model

print('Coefficients of Regression \t: a = ', linreg.coef_)
print('Intercept of Regression \t: b = ', linreg.intercept_)
print()

# Formula for the Regression line
regline_BS = BS_train
regline_SP = linreg.intercept_ + linreg.coef_ * BS_train

# Plot the Linear Regression line
f = plt.figure(figsize=(16, 8))
plt.scatter(BS_train, SP_train)
plt.plot(regline_BS, regline_SP, 'r-', linewidth = 3)
plt.show()

# Explained Variance (R^2)
print("Explained Variance (R^2) in Train Data \t:", linreg.score(BS_trai

# Predict SP values corresponding to SP Train
SP_train_pred = linreg.predict(BS_train)

# Mean Squared Error (MSE)
def mean_sq_err(actual, predicted):
    '''Returns the Mean Squared Error of actual and predicted values'''
    return np.mean(np.square(np.array(actual) - np.array(predicted)))


mse = mean_sq_err(SP_train, SP_train_pred)
print("Mean Squared Error (MSE) in Train Data \t:", mse)
print("Root Mean Squared Error (RMSE) in Train Data \t:", np.sqrt(mse))

# Predict Response corresponding to Predictors
SP_train_pred = linreg.predict(BS_train)
SP_test_pred = linreg.predict(BS_test)

# Plot the Linear Regression line
f = plt.figure(figsize=(16, 8))
```

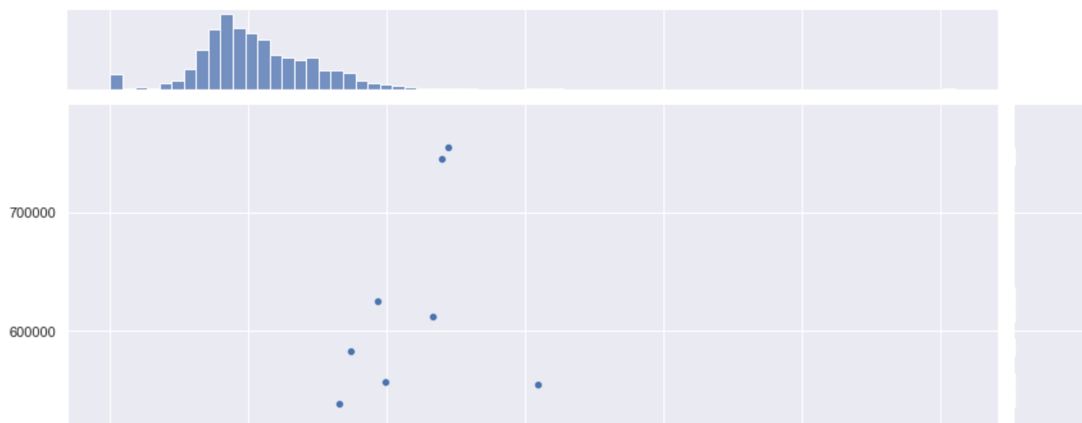
```python
plt.scatter(BS_train, SP_train)
plt.scatter(BS_train, SP_train_pred, color = "r")
plt.show()

# Explained Variance (R^2)
print("Explained Variance (R^2) in Test Data \t:", linreg.score(BS_test,

# Mean Squared Error (MSE)
def mean_sq_err(actual, predicted):
    '''Returns the Mean Squared Error of actual and predicted values'''
    return np.mean(np.square(np.array(actual) - np.array(predicted)))

mse = mean_sq_err(SP_test, SP_test_pred)
print("Mean Squared Error (MSE) in Test Data \t: ", mse)
print("Root Mean Squared Error (RMSE) in Test Data \t:", np.sqrt(mse))
```

```
             SalePrice  TotalBsmtSF
SalePrice     1.000000     0.613581
TotalBsmtSF   0.613581     1.000000
Coefficients of Regression      : a =  [[125.2136203]]
Intercept of Regression         : b =  [50174.58448806]
```



SalePrice AGAINST GarageArea

In [14]:

```
houseData = pd.read_csv('train.csv')
SalePrice_GarageArea = pd.DataFrame(houseData[['SalePrice', 'GarageArea'
sb.jointplot(data=SalePrice_GarageArea, x="GarageArea",y="SalePrice", ra

print(SalePrice_GarageArea.corr())

# Import LinearRegression model from Scikit-Learn
from sklearn.linear_model import LinearRegression
# other important models and functions to be used later
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

SalePrice = pd.DataFrame(SalePrice_GarageArea['SalePrice'])
GarageArea = pd.DataFrame(SalePrice_GarageArea['GarageArea'])
# Split the Dataset into Train and Test
SP_train, SP_test, GA_train, GA_test = train_test_split(SalePrice, Garag

linreg = LinearRegression()         # create the linear regression objec
linreg.fit(GA_train, SP_train)      # train the linear regression model

print('Coefficients of Regression \t: a = ', linreg.coef_)
print('Intercept of Regression \t: b = ', linreg.intercept_)
print()

# Formula for the Regression line
regline_GA = GA_train
regline_SP = linreg.intercept_ + linreg.coef_ * GA_train

# Plot the Linear Regression line
f = plt.figure(figsize=(16, 8))
plt.scatter(GA_train, SP_train)
plt.plot(regline_GA, regline_SP, 'r-', linewidth = 3)
plt.show()

# Explained Variance (R^2)
print("Explained Variance (R^2) in Train Data \t:", linreg.score(GA_trai

# Predict SP values corresponding to SP Train
SP_train_pred = linreg.predict(GA_train)

# Mean Squared Error (MSE)
def mean_sq_err(actual, predicted):
    '''Returns the Mean Squared Error of actual and predicted values'''
    return np.mean(np.square(np.array(actual) - np.array(predicted)))


mse = mean_sq_err(SP_train, SP_train_pred)
print("Mean Squared Error (MSE) in Train Data \t:", mse)
print("Root Mean Squared Error (RMSE) in Train Data \t:", np.sqrt(mse))

# Predict Response corresponding to Predictors
SP_train_pred = linreg.predict(GA_train)
SP_test_pred = linreg.predict(GA_test)

# Plot the Linear Regression line
f = plt.figure(figsize=(16, 8))
```

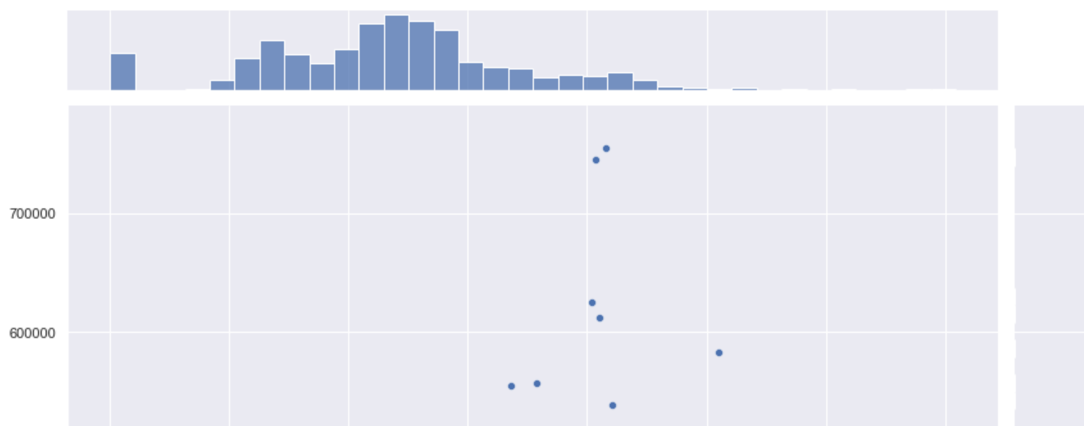
```python
plt.scatter(GA_train, SP_train)
plt.scatter(GA_train, SP_train_pred, color = "r")
plt.show()

# Explained Variance (R^2)
print("Explained Variance (R^2) in Test Data \t:", linreg.score(GA_test,

# Mean Squared Error (MSE)
def mean_sq_err(actual, predicted):
    '''Returns the Mean Squared Error of actual and predicted values'''
    return np.mean(np.square(np.array(actual) - np.array(predicted)))

mse = mean_sq_err(SP_test, SP_test_pred)
print("Mean Squared Error (MSE) in Test Data \t: ", mse)
print("Root Mean Squared Error (RMSE) in Test Data \t:", np.sqrt(mse))
```

```
            SalePrice  GarageArea
SalePrice    1.000000    0.623431
GarageArea   0.623431    1.000000
Coefficients of Regression      : a =  [[222.69494204]]
Intercept of Regression         : b =  [76880.79075444]
```



Problem 3 : Best Uni-Variate Model to Predict SalePrice

Compare and contrast the four models in terms of Explained Variance (R^2) and Mean Squared Error (MSE) on Train Data, the accuracy of prediction on Test Data, and comment on which model you think is the best to predict “SalePrice”.

List of data for comparison:

Sale Price AGAINST GarageArea

Explained Variance (R^2) in Train Data : 0.38933996560521944

Mean Squared Error (MSE) in Train Data : 3973943978.9564185

Explained Variance (R^2) in Test Data : 0.3866598851658336

Mean Squared Error (MSE) in Test Data : 3613856732.978006

correlation coefficient: 0.623431

Sale Price AGAINST TotalBsmtSF

Explained Variance (R^2) in Train Data : 0.35939721014762194

Mean Squared Error (MSE) in Train Data : 4054299177.5425606

Explained Variance (R^2) in Test Data : 0.4062893219785225

Mean Squared Error (MSE) in Test Data : 3716418198.550616

correlation coefficient: 0.613581

Sale Price AGAINST LotArea

Explained Variance (R^2) in Train Data : 0.08575685803584088

Mean Squared Error (MSE) in Train Data : 6151755244.025555

Explained Variance (R^2) in Test Data : 0.01032795318525248

Mean Squared Error (MSE) in Test Data : 5339477191.4394

correlation coefficient: 0.263843

Sale Price AGAINST GrLivArea

Explained Variance (R^2) in Train Data : 0.5181125334882235

Mean Squared Error (MSE) in Train Data : 3288172028.320085

Explained Variance (R^2) in Test Data : 0.43130664449136913

Mean Squared Error (MSE) in Test Data : 3699144756.8458285

correlation coefficient: 0.708624

Conclusion :

The higher the R^2 , the better the regression predictions approximate the real data points.

The lower the MSE, the lesser the error and the model is more accurate and closer to the actual data.

Therefore, the lower the MSE and the higher the R^2, the more accurate the prediction of Test Data.

SalePrice Against GrLivArea has the highest R^2 in Train Data of 0.5181125334882235 and lowest MSE in Train Data of 3288172028.320085 and thus has the most accurate prediction. It also has the highest correlation with SalePrice, hence, would be the best to predict SalePrice. However, the Train and Test data sets are based on raw data, if the data were cleaned, the other models may predict SalePrice better.