# REPORT

## Data Cleaning:

After downloading the data from UCI, I cleaned it to fit into the linear regression model.

At first I have selected the numerical type columns from the dataset:-

### Selecting the numerical type columns

```
In [4]: data1=data.select_dtypes(include=np.number)
```

Then I count all NA values from the dataset.

### Counting Null values per column

```
In [5]: data1.isna().sum()

Out[5]: Id                 0
        MSSubClass         0
        LotFrontage      259
        LotArea            0
        OverallQual        0
        OverallCond        0
        YearBuilt          0
        YearRemodAdd       0
        MasVnrArea         8
        BsmtFinSF1         0
        BsmtFinSF2         0
        BsmtUnfSF          0
        TotalBsmtSF        0
        1stFlrSF           0
        2ndFlrSF           0
        LowQualFinSF       0
        GrLivArea          0
        BsmtFullBath       0
        BsmtHalfBath       0
        FullBath           0
        HalfBath           0
        BedroomAbvGr       0
        KitchenAbvGr       0
        TotRmsAbvGrd       0
        Fireplaces         0
        GarageYrBlt       81
        GarageCars         0
        GarageArea         0
        WoodDeckSF         0
        OpenPorchSF        0
        EnclosedPorch      0
        3SsnPorch          0
        ScreenPorch        0
        PoolArea           0
        MiscVal            0
        MoSold             0
        YrSold             0
        SalePrice          0
        dtype: int64
```

Since only few columns have NA values so I dropped them.

### Removing NA Values

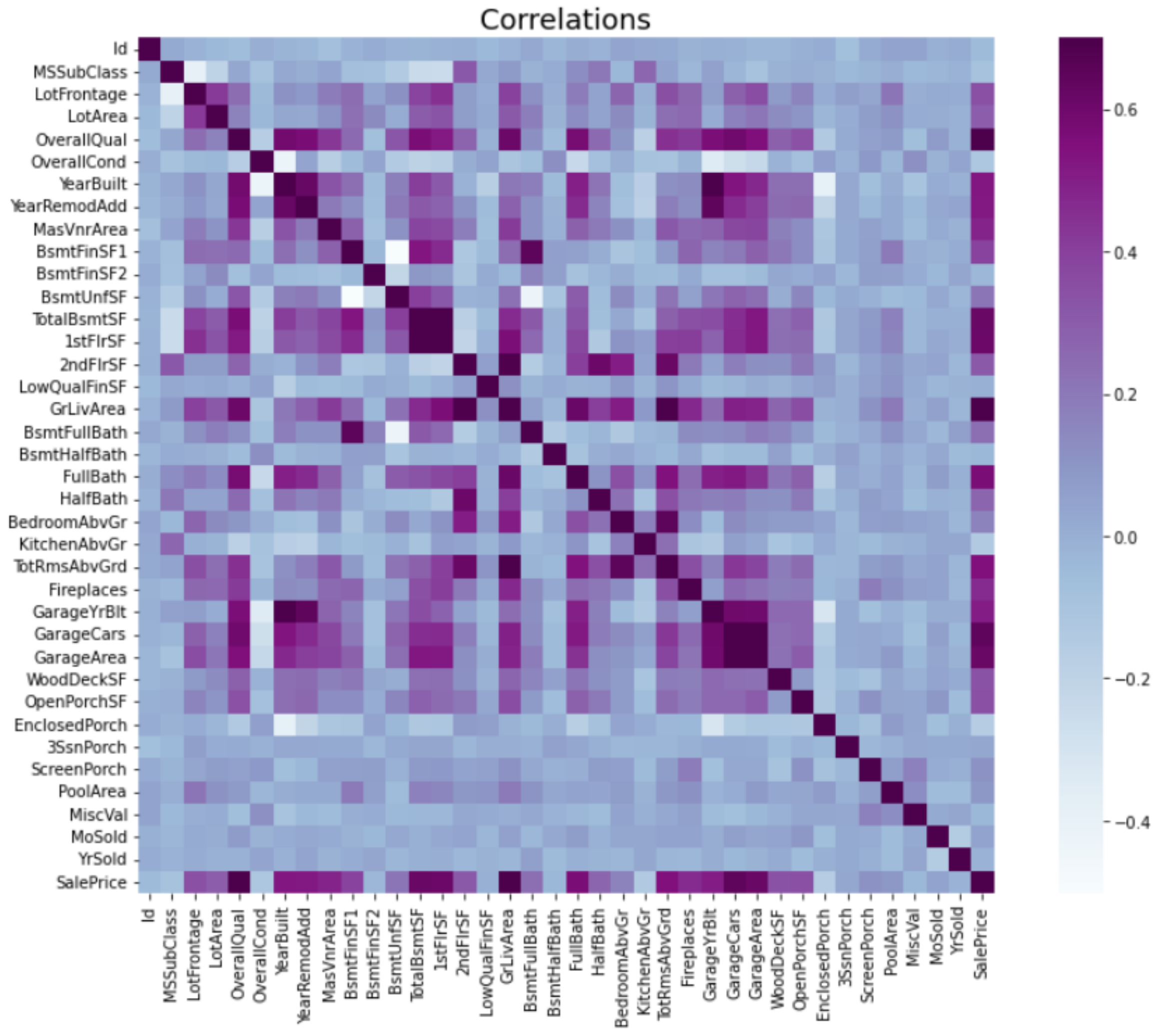
```
In [6]: df =data1.dropna()
        #d.isna().sum()#d.isna().sum()
```

Then I have plotted the heat map of the correlation matrix

## Create correlation matrix and the heatmap

```
In [7]: corr = df.corr()
        plt.subplots(figsize=(15,10))
        ax = sns.heatmap(corr, cmap="BuPu", vmax=0.7, square=True)
        ax.set_title("Correlations", fontsize = 18)

Out[7]: Text(0.5, 1.0, 'Correlations')
```



Using the correlation matrix heat map, I have selected the most suitable columns for linear regression.

Then I have converted the data frame into two numpy arrays, named them X and Y.

## Converting the dataframe into two numpy arrays

```
X = np.array(df[['GarageCars',
                 'GarageArea',
                 'GrLivArea',
                 'OverallQual',
                 'TotRmsAbvGrd',
                 '1stFlrSF',
                 'FullBath',
                 'WoodDeckSF',
                 'OpenPorchSF']])

y = np.array(df['SalePrice'])
```

OLS Regression Results

# Ordinary Least Squares

In [34]:
```
%matplotlib inline
```

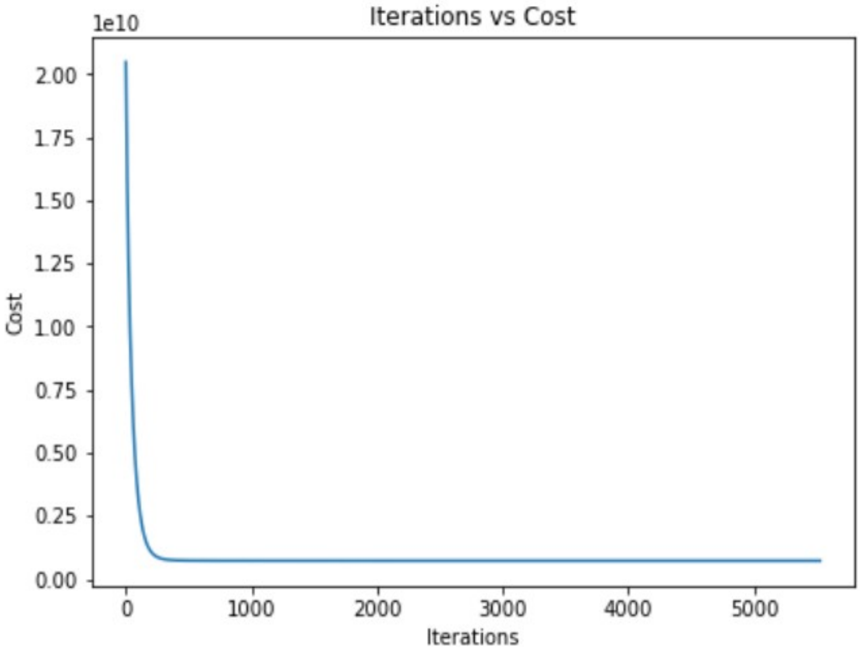
In [35]:
```
res = sm.OLS(y_train, X_train).fit()
print(res.summary())
```

```
                                 OLS Regression Results                                
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.955
Model:                            OLS   Adj. R-squared (uncentered):              0.954
Method:                 Least Squares   F-statistic:                              1808.
Date:                Thu, 17 Mar 2022   Prob (F-statistic):                        0.00
Time:                        15:12:12   Log-Likelihood:                         -9479.8
No. Observations:                 784   AIC:                                  1.898e+04
Df Residuals:                     775   BIC:                                  1.902e+04
Df Model:                           9                                                  
Covariance Type:            nonrobust                                                  
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1           1.31e+04   4781.282      2.740      0.006    3717.256    2.25e+04
x2            37.3806     15.702      2.381      0.018       6.557      68.205
x3            84.4390      5.988     14.102      0.000      72.685      96.193
x4          1.553e+04   1392.134     11.157      0.000    1.28e+04    1.83e+04
x5         -1.517e+04   1454.407    -10.430      0.000    -1.8e+04   -1.23e+04
x6            19.7530      5.271      3.747      0.000       9.405      30.101
x7         -8185.1417   3920.950     -2.088      0.037   -1.59e+04    -488.201
x8            58.4710     13.432      4.353      0.000      32.104      84.838
x9            67.0294     25.666      2.612      0.009      16.646     117.413
==============================================================================
Omnibus:                      308.490   Durbin-Watson:                   2.068
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7148.142
Skew:                           1.222   Prob(JB):                         0.00
Kurtosis:                      17.589   Cond. No.                     6.58e+03
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

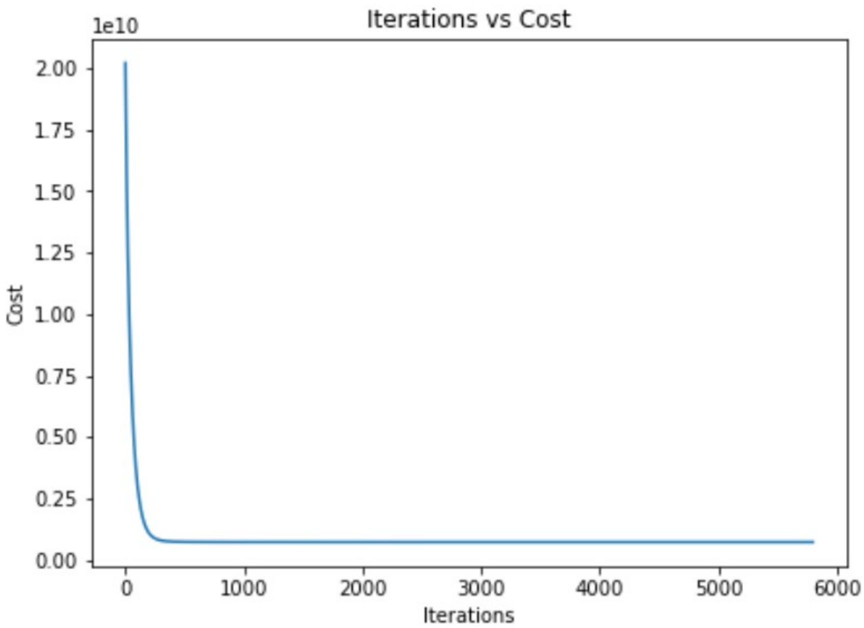
Batched Gradient Descent

```
Train Score: 0.7190043786890068
Test Score:  0.6269585571709697
```




Sequential Gradient Descent

```
Train Score: 0.7193427721554584
Test Score:  0.6273421816020116
```



In this case Sequential Gradient Descent takes more iterations and give slightly better result than Batched Gradient Descent.