In [45]:
```python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import scipy.stats as stats
```

In [46]:
```python
housing = pd.read_csv(r"D:\PAP_prep\Data Analytics\Assignment 1\house_prices.csv")
```

In [47]:
```python
housing.head()
```

Out[47]:

	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	...	GarageA
0	65.0	8450	7	5	2003	2003	196.0	706	0	150	...	
1	80.0	9600	6	8	1976	1976	0.0	978	0	284	...	
2	68.0	11250	7	5	2001	2002	162.0	486	0	434	...	
3	60.0	9550	7	5	1915	1970	0.0	216	0	540	...	
4	84.0	14260	8	5	2000	2000	350.0	655	0	490	...	

5 rows × 35 columns

Q. Evaluate the methods : shape, info, describe.

shape() tells us the number of rows and columns.

info() tellus the data-type, number of not null values of the DataFrame.

describe() lists the mean, count, standard-deviation, minimum, maximum, and Quartile Ranges and their values in the DataFrame.

In [48]:
```
housing.describe()
```

Out[48]:

	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSI
count	1379.000000	1379.000000	1379.000000	1379.000000	1379.000000	1379.000000	1379.000000	1379.000000	1379.000000	1379.00000
mean	57.766497	10695.812183	6.187092	5.577955	1972.958666	1985.435098	108.364757	455.578680	48.102248	570.76504
std	35.038221	10214.702133	1.345780	1.081031	29.379883	20.444852	184.195220	459.691379	164.324665	443.67784
min	0.000000	1300.000000	2.000000	2.000000	1880.000000	1950.000000	0.000000	0.000000	0.000000	0.00000
25%	41.500000	7741.000000	5.000000	5.000000	1955.000000	1968.000000	0.000000	0.000000	0.000000	228.00000
50%	64.000000	9591.000000	6.000000	5.000000	1976.000000	1994.000000	0.000000	400.000000	0.000000	476.00000
75%	79.000000	11708.500000	7.000000	6.000000	2001.000000	2004.000000	170.500000	732.000000	0.000000	811.00000
max	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000	2336.00000

8 rows × 35 columns

In [49]:
```
housing.info()
housing.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1379 entries, 0 to 1378
Data columns (total 35 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   LotFrontage    1379 non-null   float64
 1   LotArea        1379 non-null   int64  
 2   OverallQual    1379 non-null   int64  
 3   OverallCond    1379 non-null   int64  
 4   YearBuilt      1379 non-null   int64  
 5   YearRemodAdd   1379 non-null   int64  
 6   MasVnrArea     1379 non-null   float64
 7   BsmtFinSF1     1379 non-null   int64  
 8   BsmtFinSF2     1379 non-null   int64  
 9   BsmtUnfSF      1379 non-null   int64  
 10  TotalBsmtSF    1379 non-null   int64  
 11  1stFlrSF       1379 non-null   int64  
 12  2ndFlrSF       1379 non-null   int64  
 13  LowQualFinSF   1379 non-null   int64  
 14  GrLivArea      1379 non-null   int64  
 15  BsmtFullBath   1379 non-null   int64  
 16  BsmtHalfBath   1379 non-null   int64  
 17  FullBath       1379 non-null   int64  
 18  HalfBath       1379 non-null   int64  
 19  BedroomAbvGr   1379 non-null   int64  
 20  KitchenAbvGr   1379 non-null   int64  
 21  TotRmsAbvGrd   1379 non-null   int64  
 22  Fireplaces     1379 non-null   int64  
 23  GarageYrBlt    1379 non-null   float64
 24  GarageCars     1379 non-null   int64  
 25  GarageArea     1379 non-null   int64  
 26  WoodDeckSF     1379 non-null   int64  
 27  OpenPorchSF    1379 non-null   int64  
 28  EnclosedPorch  1379 non-null   int64  
 29  3SsnPorch      1379 non-null   int64  
 30  ScreenPorch    1379 non-null   int64  
 31  PoolArea       1379 non-null   int64  
 32  MiscVal        1379 non-null   int64  
```

```
 33  YrSold         1379 non-null   int64
 34  SalePrice      1379 non-null   int64
dtypes: float64(3), int64(32)
memory usage: 377.2 KB
```

Out[49]:
```
(1379, 35)
```

In [50]:
```
saleP =  housing['SalePrice']
```

Q. For the Saleprice attribute. Evaluate Mean,Median,Mode

Mean is the average value of the data-column.

Median is the central values of a sorted data-column.

Mode is the maximum times occuring values in the dataset.

In [51]:
```
saleP.mean()
```

Out[51]:
```
185479.511240029
```

In [52]:
```
saleP.median()
```

Out[52]:
```
167500.0
```

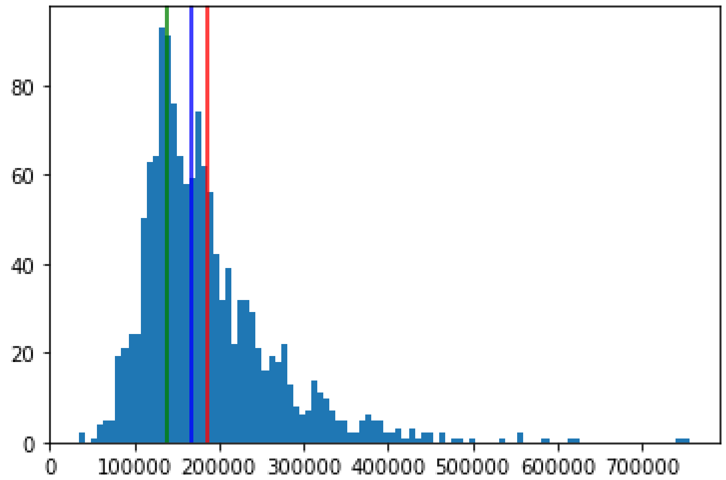
In [53]:
```
saleP.mode()
```

Out[53]:
```
0    140000
dtype: int64
```

Visualize histogram for saleprice. USE THE PLOT OF MEAN,MEDIAN, MODE.

In [54]:
```
plt.hist(housing['SalePrice'],density=False,bins = 100,label="Histogram")
plt.axvline(housing.SalePrice.mean(), color ='r', label="Mean")
plt.axvline(housing.SalePrice.median(), color ='b', label="Median")
plt.axvline(housing.SalePrice.mode()[0], color ='g', label="Mode")
```

Out[54]: <matplotlib.lines.Line2D at 0x12cdc264cd0>



In [55]:
```
np.ptp(saleP)
```

Out[55]: 719689

Q. Evaluate measures of spread :range,variance,standard deviation,skewness and kurtosis

Range is the minimum and the maximum values between which the data is spread. The range is [0-719689]

Variance is the cumulative deviation from the mean.[6244775285.52]

Standard Deviation is the square root of the variance. [79023.890]

Standard Deviation is the square root of the variance: [79023.890]
This data is positively skewed, meaning the majority of the houses were sold below the average price, or the mode of the data is less than the average or mean of the dataset.

Kurtosis value is 6.73, distributions with large kurtosis shows tail data exceeding the tails to that of the normal distribution.

In [56]:
```
print("Variance of the data values from the mean:",saleP.var())
```

```
Variance of the data values from the mean: 6244775285.521461
```

In [57]:
```
saleP.std()
```

Out[57]:
```
79023.89059975129
```

In [58]:
```
saleP.kurtosis()
```

Out[58]:
```
6.735649337267559
```

In [59]:
```
saleP.max() - saleP.min()
```

Out[59]:
```
719689
```

In [60]:
```
saleP.skew()
```

Out[60]:
```
1.935362098363132
```

In [61]:
```
saleP = sorted(saleP)
# saleP.head()
```

Evaluate Visualize the histogram of the normal distribution.

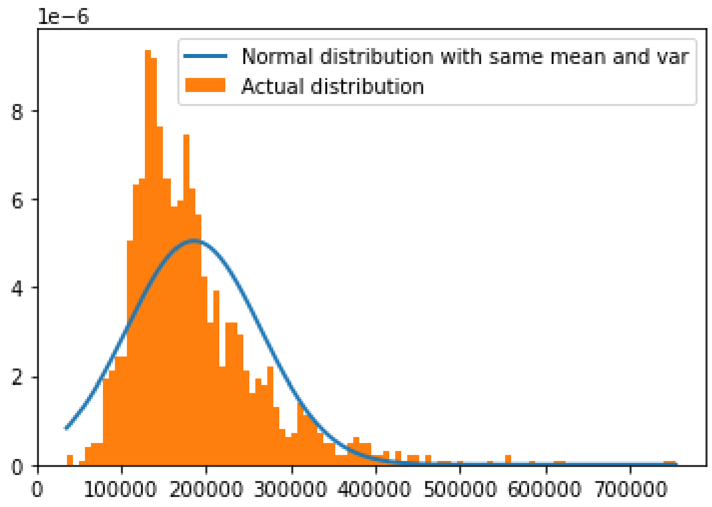
This data is positively skewed, meaning the majority of the houses were sold below the average price, or the mode of the data is less than the average or mean of the dataset.

In [62]:

```
#convert pandas DataFrame object to numpy array and sort
h = np.asarray(saleP)
h = sorted(h)

#use the scipy stats module to fit a normal distirbution with same mean and standard deviation
nor_dist = stats.norm.pdf(h, np.mean(h), np.std(h))

#plot both series on the histogram
plt.plot(h,nor_dist,'-',linewidth = 2,label="Normal distribution with same mean and var")
plt.hist(h,density=True,bins = 100,label="Actual distribution")
plt.legend()
plt.show()
```



Q. Evaluate visualize boxplot

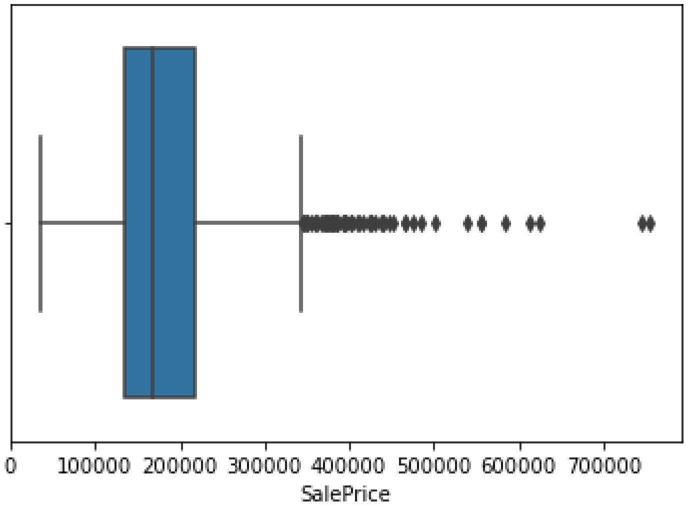
BoxPlots tells us what percent of data points are areater of less than 25%, 50%, 75% and 100%.

BoxPlot graphically demonstratrates the locality, spread and skewness groups of numerical data

The datapoints that are outside the BoxPlots are generally outliers (We take 1.5 time the IQR to find the outliers.)

In [63]:
```
sns.boxplot(x = 'SalePrice',data = housing)
```

Out[63]: <AxesSubplot:xlabel='SalePrice'>



Q. Evaluate the Quartiles q1, q3 and iqr

A quartile is a type of quantile which divides the number of data points into four parts, or quarters.

In [64]:
```
#Quartiles
saleprice = housing['SalePrice']
saleprice = pd.DataFrame(saleprice)
Q1 = saleprice.quantile(0.25)
Q2 = saleprice.quantile(0.5)
Q3 = saleprice.quantile(0.75)

print(Q1)
print(Q2)
print(Q3)
```

```
SalePrice    134000.0
Name: 0.25, dtype: float64
SalePrice    167500.0
Name: 0.5, dtype: float64
SalePrice    217750.0
Name: 0.75, dtype: float64
```

In [65]:
```
IQR = Q3-Q1
IQR
```

Out[65]:
```
SalePrice    83750.0
dtype: float64
```

Statistically; datapoints that are less than or greater than 1.5 times the IQR are termed as outliers.

In [66]:
```
df = saleprice[((saleprice < (Q1 - 1.5 * IQR)) |(saleprice > (Q3 + 1.5 * IQR))).any(axis=1)]
```

USE IQR rule to detect outliers

In [67]:

```
print(df.sort_values(by='SalePrice'))
```

```
      SalePrice
11       345000
601      345000
934      348000
828      350000
612      350000
300      354000
573      359100
288      360000
660      361919
1159     367294
550      369900
603      370878
140      372402
636      372500
453      374000
292      375000
315      377426
1311     377500
1196     378500
449      380000
1197     381000
105      383970
51       385000
778      385000
208      386250
646      392000
1114     392500
356      394432
1359     394617
728      395000
931      395192
619      402000
486      402861
215      403000
1276     410000
149      412500
260      415298
622      423000
```

```
1078     424870
366      426000
467      430000
328      437154
56       438780
445      440000
497      446261
556      451950
1173     465000
1296     466500
171      475000
751      485000
164      501837
723      538000
414      555000
986      556581
756      582933
849      611657
1104     625000
1115     745000
649      755000
```

Q. Evaluate the Correlation and covariance for the attributes in the dataset

lotarea, grlivarea,garagearea,saleprice

Covariance indicates the direction of the linear relationship between variables. [Ranged from negative infinity to positive infinity.]

Correlation measures both the strength and direction of the linear relationship between two variables. (Ranged from -1 to 1).. Negative means if one variable increases the other would decrease with the strength mentioned by coeff or correlation.

GrLivArea and SalePrice have highest correlation, that means if one of them increases other would increase.

In [68]:
```
#covariance
covariance = housing[['LotArea','GrLivArea','GarageArea','SalePrice']].cov().head()
# plt.plot(covariance)
```

In [69]:
```
#checking correlation of 4 countinous variables
import seaborn as sns
%matplotlib inline
corelation=housing[['LotArea','GrLivArea','GarageArea','SalePrice']].corr()
print (corelation)

sns.heatmap(corelation)
```

	LotArea	GrLivArea	GarageArea	SalePrice
LotArea	1.000000	0.257243	0.167622	0.252921
GrLivArea	0.257243	1.000000	0.478811	0.708172
GarageArea	0.167622	0.478811	1.000000	0.608405
SalePrice	0.252921	0.708172	0.608405	1.000000

Out[69]: <AxesSubplot:>