PART II Programming (15 Points)

Problem 2.1 (0 Points) Read the CSV file "House_Prices_PRED.CSV"

In [ ]:
```
import pandas as pd
import statsmodels.formula.api as smf
houses = pd.read_csv('House_Prices_PRED.csv')
houses = houses.iloc[: , 1:]
houses.head()
```

Out[ ]:

	P_SalePrice	SalePrice
0	206307.7360	208500
1	179044.5328	181500
2	217258.4337	223500
3	161547.6322	140000
4	272594.2471	250000

Problem 2.2 (3 Points) Write a program to calculate the sum squared of error and the average squared error of the Model (i.e., P_House_Price).

In [ ]:
```
import numpy as np

#sum of differences squared
sumDif2 = np.sum((houses['SalePrice'] - houses['P_SalePrice'])**2)
#sum of differences squared divided by # of rows
print("SSE = ",sumDif2)
print("ASE = ",sumDif2/houses.shape[0])
```

```
SSE =  740014639177.1643
ASE =  506859341.9021673
```

Problem 2.3 (3 Points) Write a program to calculate the R2 of the Model (i.e., P_House_Price).

In [ ]:
```
#Y minus YBar Squared
ySubMean2 = np.sum((houses['SalePrice'] - houses['SalePrice'].mean())**2)
#1 minus sum of differences squared divided by y minus ybar squared
print("R^2 = ", (1 - sumDif2/ySubMean2))
```

```
R^2 =  0.9196327362106914
```

Problem 2.4 (3 Points) Write a program to calculate the MAPE of the Model (i.e., P_House_Price).

In [ ]:
```
dif = np.abs(houses['SalePrice'] - houses['P_SalePrice'])
DifdivY = np.sum(np.divide(dif, houses['SalePrice']))
# sum of differences divided by actual value times, divided by number of rows, times 1
print("MAPE = ", (DifdivY*(1/houses.shape[0]) * 100), "%")
```

```
MAPE =  7.026392138631052 %
```

Problem 2.5 (3 Points) Write a program to calculate the MAE of the Model (i.e., P_House_Price).

In [ ]:
```
dif = np.sum(np.abs(houses['SalePrice'] - houses['P_SalePrice']))
# sum of differences divided by number of rows
print("MAE = ", dif*(1/houses.shape[0]))
```

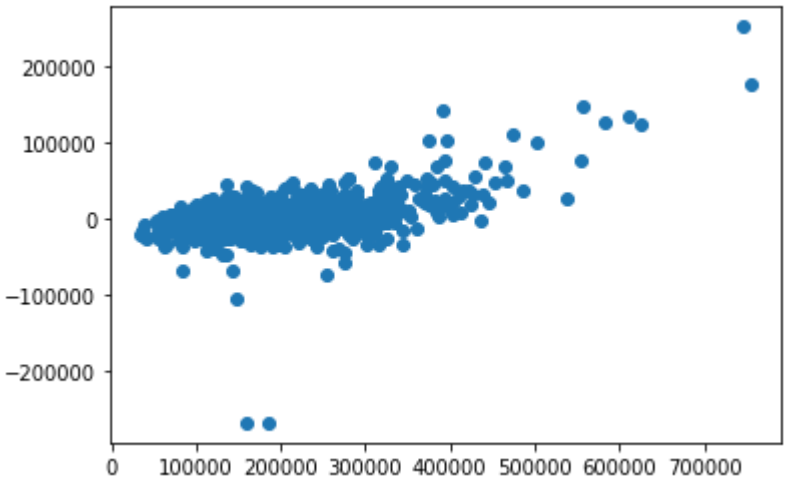
```
MAE =  12470.833673842466
```

Problem 2.6 (3 Points) Write a program to produce a residual plot with residual on the Y-axis and observed value (House_Price) and to impose a loess line on the graph.

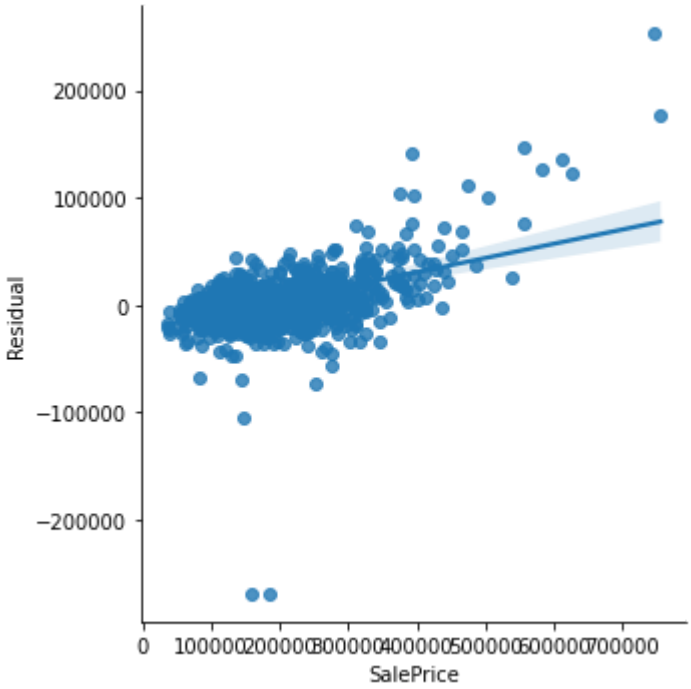
In [17]:
```
from matplotlib import pyplot as plt
import seaborn as sns

houses['Residual'] = houses['SalePrice'] - houses['P_SalePrice']

plt.scatter(houses['SalePrice'], houses['Residual'])
plt.show()
sns.lmplot(x='SalePrice', y='Residual', data=houses)
```



Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x7f88ae09b290>
```


200000
100000
0
−100000
−200000
Residual
0
100000
200000
300000
400000
500000
600000
700000
SalePrice