House Prediction Project

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from google.colab import files
upload=files.upload()
```

Choose Files train.csv

- **train.csv**(text/csv) - 460676 bytes, last modified: 1/5/2025 - 100% done

```
Saving train.csv to train.csv
```

```
df=pd.read_csv("./train.csv")
```

```
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFea
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	

5 rows × 81 columns

```
df.info()
```

```
25  MasVnrType     588 non-null    object
26  MasVnrArea     1452 non-null   float64
27  ExterQual      1460 non-null   object
28  ExterCond      1460 non-null   object
29  Foundation     1460 non-null   object
30  BsmtQual       1423 non-null   object
31  BsmtCond       1423 non-null   object
32  BsmtExposure   1422 non-null   object
33  BsmtFinType1   1423 non-null   object
34  BsmtFinSF1     1460 non-null   int64
35  BsmtFinType2   1422 non-null   object
```

```
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
df=df[["Id","LotArea","FullBath","BedroomAbvGr","SalePrice"]]
```

```
df.head()
```

	Id	LotArea	FullBath	BedroomAbvGr	SalePrice
0	1	8450	2	3	208500
1	2	9600	2	3	181500
2	3	11250	2	3	223500
3	4	9550	1	3	140000
4	5	14260	2	4	250000

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.isnull().sum()
```

	0
Id	0
LotArea	0
FullBath	0
BedroomAbvGr	0
SalePrice	0

dtype: int64

```
df["Id"].duplicated().sum()
```

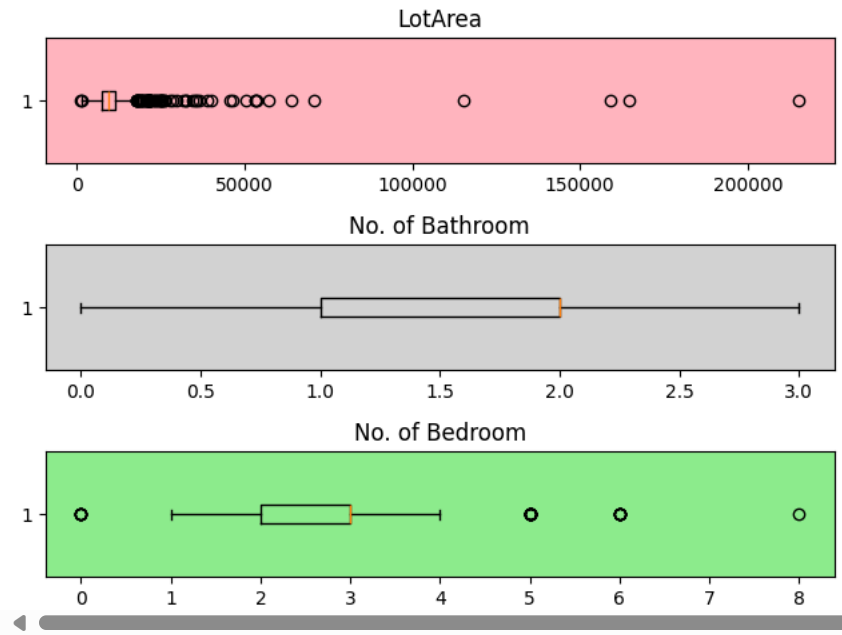
```
0
```

```
plt.subplot(3,1,1,facecolor="lightpink")
plt.boxplot(df['LotArea'], vert=False)
plt.title("LotArea")

plt.subplot(3,1,2,facecolor="lightgrey")
plt.boxplot(df['FullBath'], vert=False)
plt.title("No. of Bathroom")

plt.subplot(3,1,3,facecolor="lightgreen")
plt.boxplot(df['BedroomAbvGr'], vert=False)
plt.title("No. of Bedroom")

plt.tight_layout()
plt.show()
```




```
# calculate summary statistics
mean = df["LotArea"].mean()
std  = df["LotArea"].std()

# Calculate the lower and upper bounds
lower_bound = mean - std*2
upper_bound = mean + std*2

print("LotArea")
print('Lower Bound :',lower_bound)
print('Upper Bound :',upper_bound)

# Drop the outliers
df = df[(df["LotArea"] >= lower_bound) & (df["LotArea"] <= upper_bound)]
```

```
LotArea
Lower Bound : -9445.701782566512
Upper Bound : 30479.357946950076
```

```
# calculate summary statistics
mean = df["BedroomAbvGr"].mean()
std  = df["BedroomAbvGr"].std()

# Calculate the lower and upper bounds
lower_bound = mean - std*2
upper_bound = mean + std*2

print("No. of Bedroom")
print('Lower Bound :',lower_bound)
print('Upper Bound :',upper_bound)

# Drop the outliers
df = df[(df["BedroomAbvGr"] >= lower_bound) & (df["BedroomAbvGr"] <= upper_bound)]
```

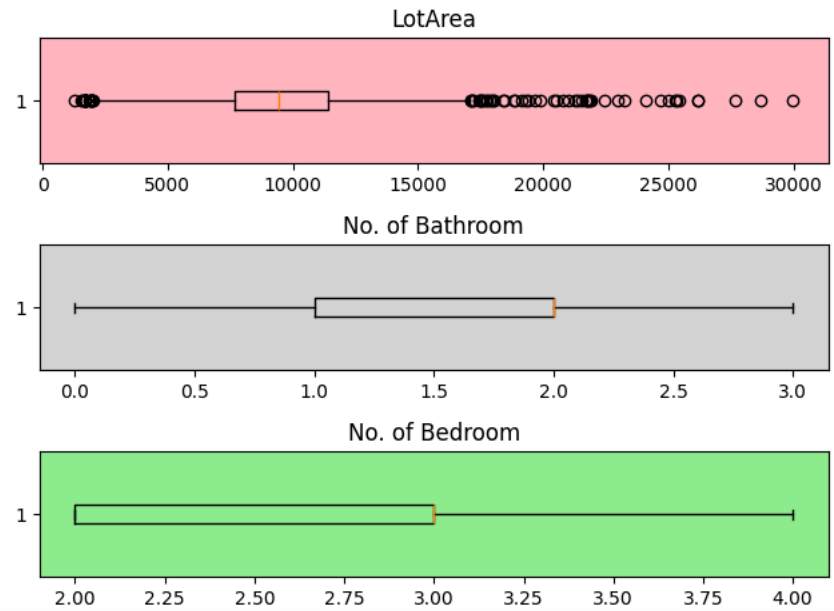
```
No. of Bedroom
Lower Bound : 1.2398643173108006
Upper Bound : 4.487534848196849
```

```
plt.subplot(3,1,1,facecolor="lightpink")
plt.boxplot(df['LotArea'], vert=False)
plt.title("LotArea")

plt.subplot(3,1,2,facecolor="lightgrey")
plt.boxplot(df['FullBath'], vert=False)
plt.title("No. of Bathroom")

plt.subplot(3,1,3,facecolor="lightgreen")
plt.boxplot(df['BedroomAbvGr'], vert=False)
plt.title("No. of Bedroom")
```

```
plt.tight_layout()
plt.show()
```




```
df.head()
```

	Id	LotArea	FullBath	BedroomAbvGr	SalePrice
0	1	8450	2	3	208500
1	2	9600	2	3	181500
2	3	11250	2	3	223500
3	4	9550	1	3	140000
4	5	14260	2	4	250000

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split


X=df.iloc[:,1:4]
Y=df.iloc[:,-1]


X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.33,random_state=45)


lin_reg=LinearRegression()


lin_reg.fit(X_train,Y_train)
```

```
LinearRegression
LinearRegression()
```

```
Y_pred=lin_reg.predict(X_test)


from sklearn.model_selection import cross_val_score
```

```
mse=cross_val_score(lin_reg,X_train,Y_train,scoring="neg_mean_squared_error",cv=10)
print("Cross Validation Score :",mse.mean())
```

```
Cross Validation Score : -3403972963.675959
```

```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

```
linear_mean_absolute_error=mean_absolute_error(Y_test,Y_pred)
print("Mean Absolute Error for this model :",linear_mean_absolute_error)
```

```
Mean Absolute Error for this model : 40180.10168555605
```

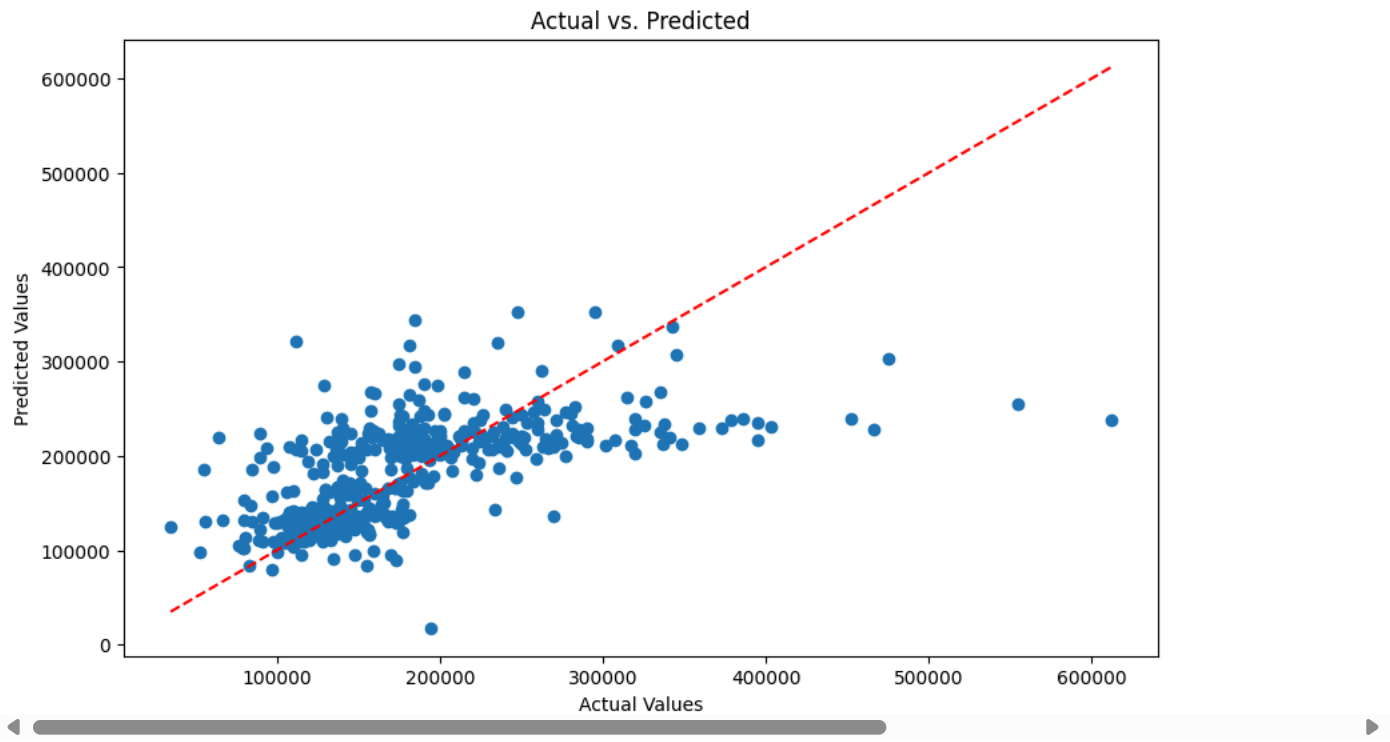
```
linear_mean_squared_error=mean_squared_error(Y_test,Y_pred)
print("Mean Squared Error for this model :",linear_mean_squared_error)
```

```
Mean Squared Error for this model : 3526303064.483335
```

```
linear_r2_score=r2_score(Y_test,Y_pred)
print("R2 Score for this model :",linear_r2_score)
```

```
R2 Score for this model : 0.35047422147944085
```

```
plt.figure(figsize=(10, 6))
plt.scatter(Y_test, Y_pred)
plt.plot([Y_test.min(), Y_test.max()], [Y_test.min(), Y_test.max()], color='red', linestyle='--')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted')
plt.show()
```




```
print("LotArea : 11250")
print("No. of Bathroom : 2")
print("No. of Bedroom : 3")
print("Price (Actual Value) : 223500")

temp = pd.DataFrame({'LotArea': [11250],'FullBath' : [2], 'BedroomAbvGr': [3]})
predicted_price = lin_reg.predict(temp)
print("Price (Predicted Value) :",predicted_price[0])
```

```
LotArea : 11250
No. of Bathroom : 2
No. of Bedroom : 3
```

```
Price (Actual Value) : 223500
Price (Predicted Value) : 224799.32396673865
```