ML Task-01

Implement a linear regression model to predict the prices of houses based on their square footage and the number of bedrooms and bathrooms.

Dataset : - https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data (https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data)

In [24]:

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

In [25]:

```
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
```

In [26]:

```
df_train.head()
```

Out[26]:

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

5 rows × 81 columns

In [27]:
```
df_test.head()
```

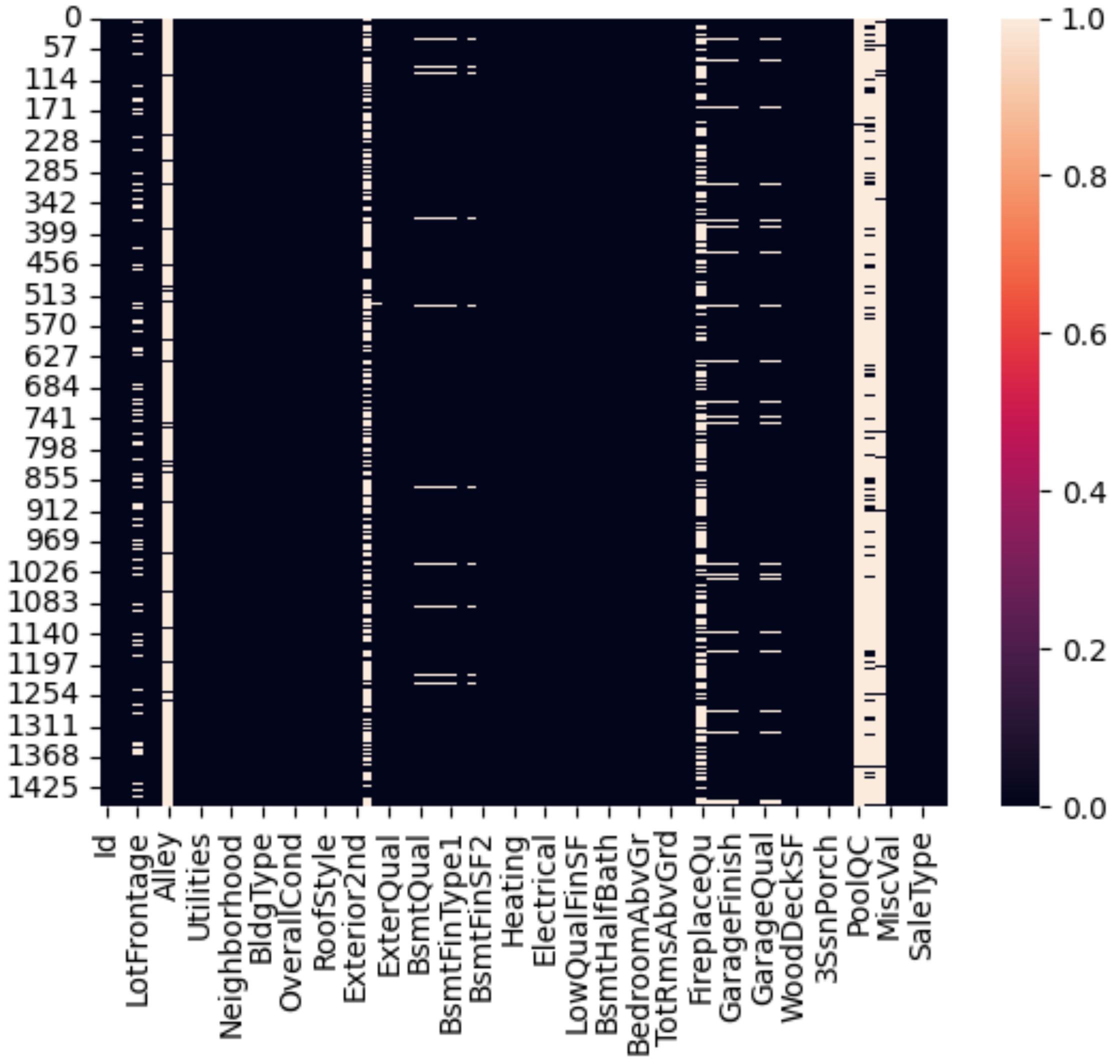
Out[27]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS

5 rows × 80 columns

In [28]:
```
sns.heatmap(df_train.isnull())
```

Out[28]: <Axes: >



In [29]:
```
df_train.columns
```

Out[29]:
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'St
reet',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'Bld
gType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'Yea
rRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVn
rType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQu
al',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'He
ating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrS
F',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'G
arageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'G
arageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'Poo
lQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleT
ype',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

In [30]:
```
null_columns=[]
for i in df_train.columns.tolist():
    if df_train[i].isnull().sum() >= 500:
        null_columns.append(i)
        print(i,df_train[i].isnull().sum())
```

```
Alley 1369
MasVnrType 872
FireplaceQu 690
PoolQC 1453
Fence 1179
MiscFeature 1406
```

In [31]:
```
df_train = df_train.drop(columns=null_columns)
df_test = df_test.drop(columns=null_columns)
```

In [32]:
```
df_train_cleaned = df_train.ffill()
df_test_cleaned = df_test.ffill()
```

In [33]:
```
df_train_cleaned.head()
```

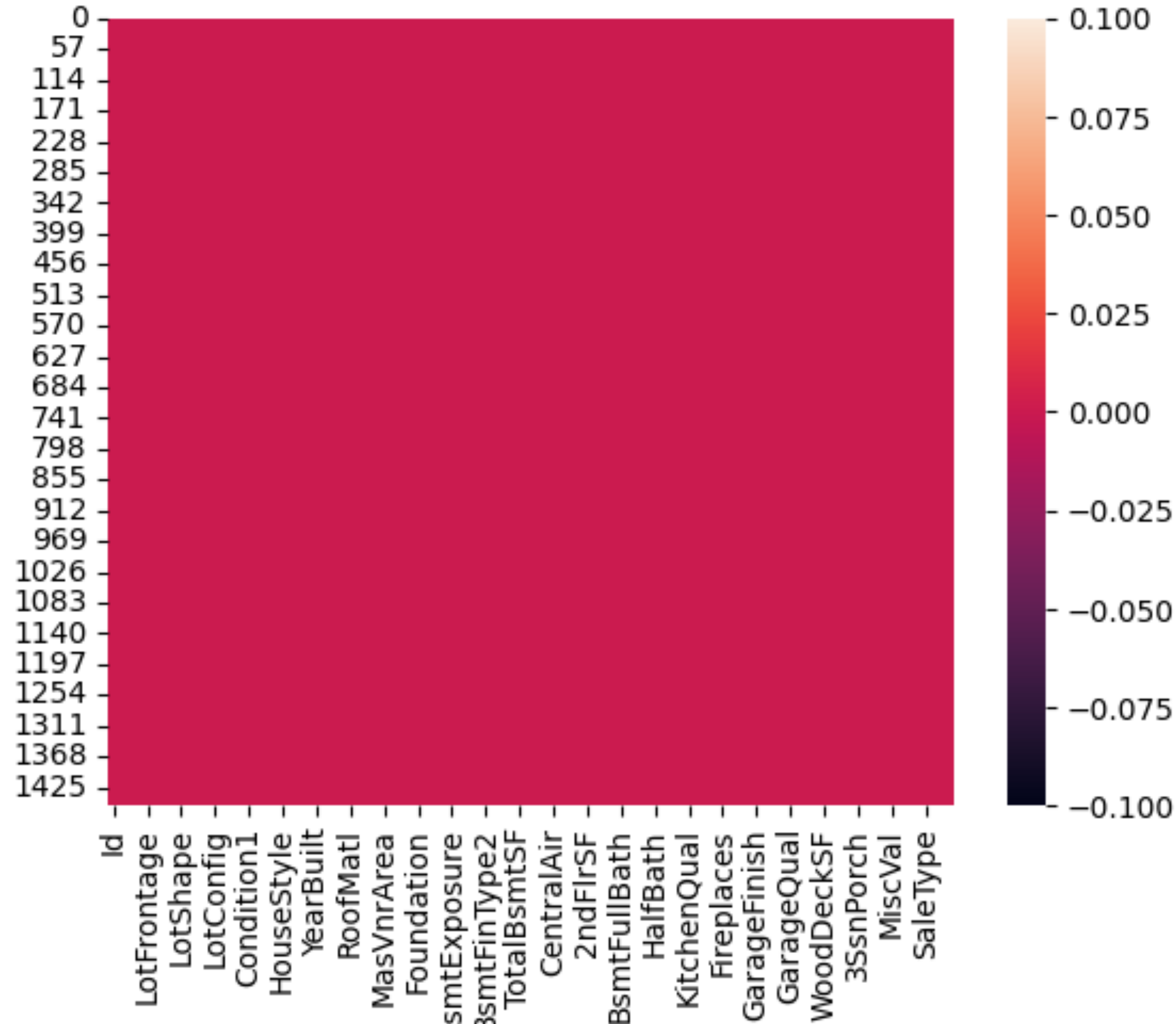
Out[33]:

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

5 rows × 75 columns

In [34]:
```
sns.heatmap(df_train_cleaned.isnull())
```

Out[34]: <Axes: >


0
57
114
171
228
285
342
399
456
513
570
627
684
741
798
855
912
969
1026
1083
1140
1197
1254
1311
1368
1425
Id
LotFrontage
LotShape
LotConfig
Condition1
HouseStyle
YearBuilt
RoofMatl
MasVnrArea
Foundation
BsmtExposure
BsmtFinType2
TotalBsmtSF
CentralAir
2ndFlrSF
BsmtFullBath
HalfBath
KitchenQual
Fireplaces
GarageFinish
GarageQual
WoodDeckSF
3SsnPorch
MiscVal
SaleType
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

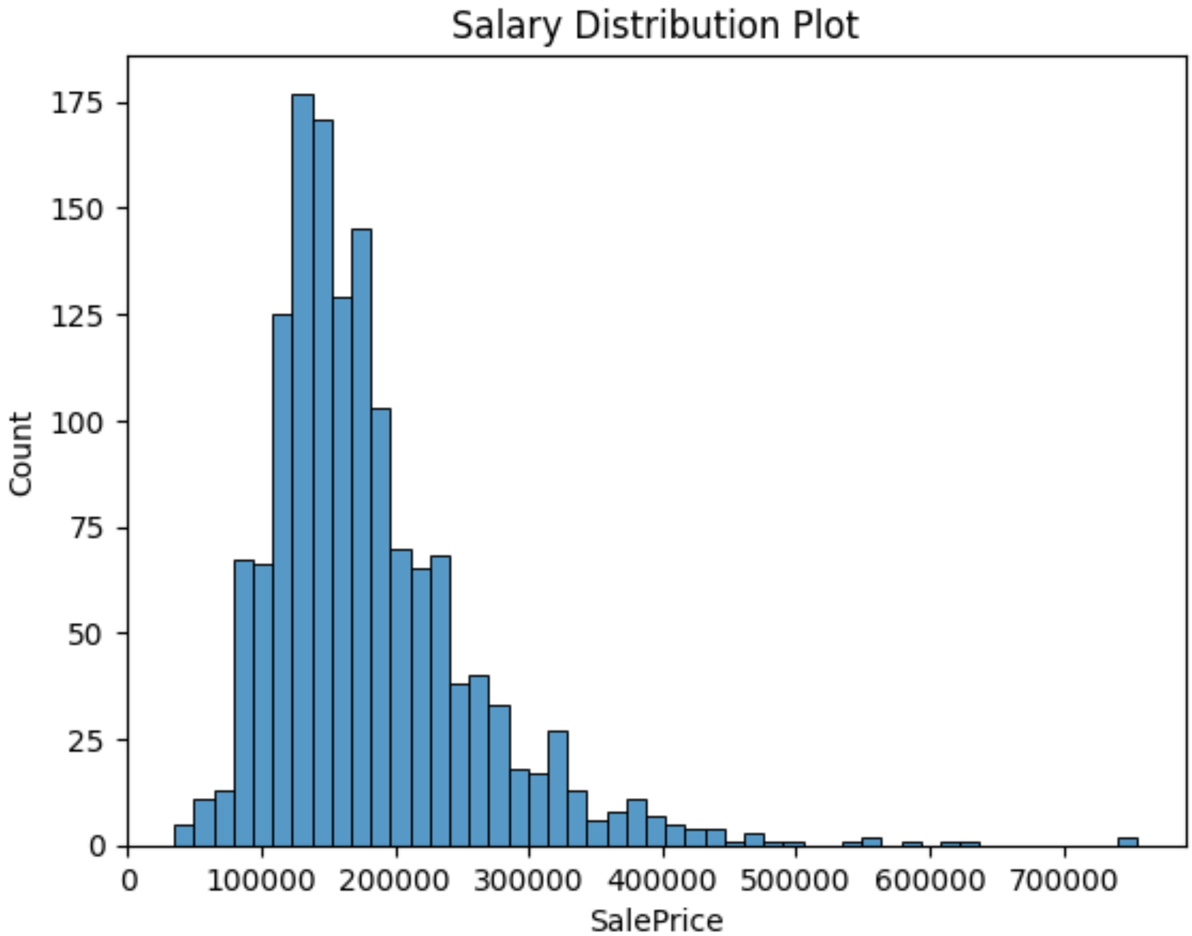
In [35]:
```
df_train_cleaned.describe()
```

Out[35]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Y
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460
mean	730.500000	56.897260	70.104795	10516.828082	6.099315	5.575342	1971
std	421.610009	42.300571	23.846996	9981.264932	1.382997	1.112799	30
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954
50%	730.500000	50.000000	70.000000	9478.500000	6.000000	5.000000	1973
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010

8 rows × 38 columns

In [36]:
```
plt.title('Salary Distribution Plot')
sns.histplot(df_train_cleaned['SalePrice'])
plt.show()
```



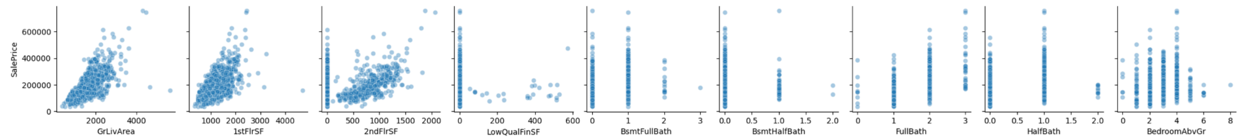
Split data

In [37]:
```
from sklearn.model_selection import train_test_split
```

In [38]:
```
features = ['GrLivArea', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'Bsmt
FullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr']

target = 'SalePrice'
```

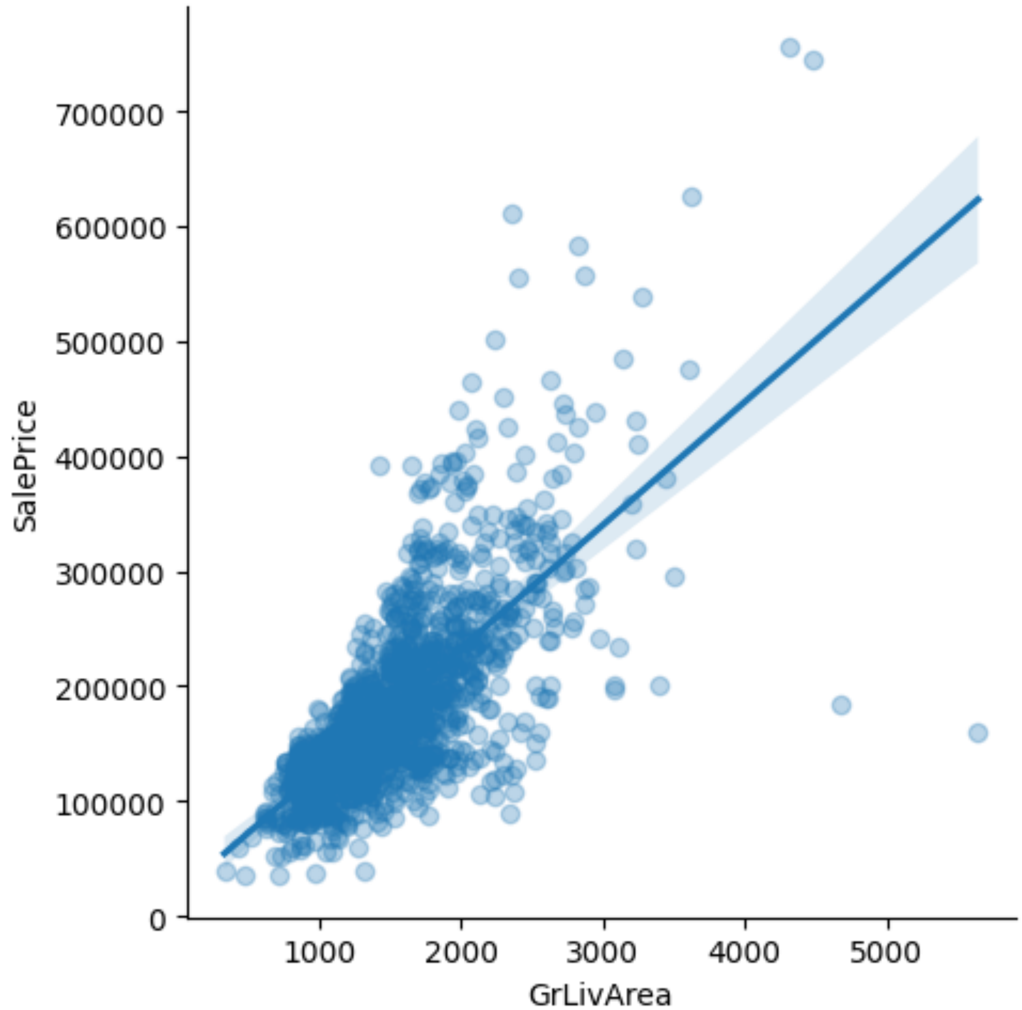
In [39]:
```
X = df_train_cleaned[features]
y = df_train_cleaned[target]
```

In [40]:
```
df = pd.concat([X, y], axis=1)

# Plotting grid of scatter plots for each feature against the target variable
sns.pairplot(df, x_vars=features, y_vars=[target], kind='scatter',
             plot_kws={'alpha':0.4}, diag_kws={'alpha':0.55, 'bins':40})

plt.show()
```



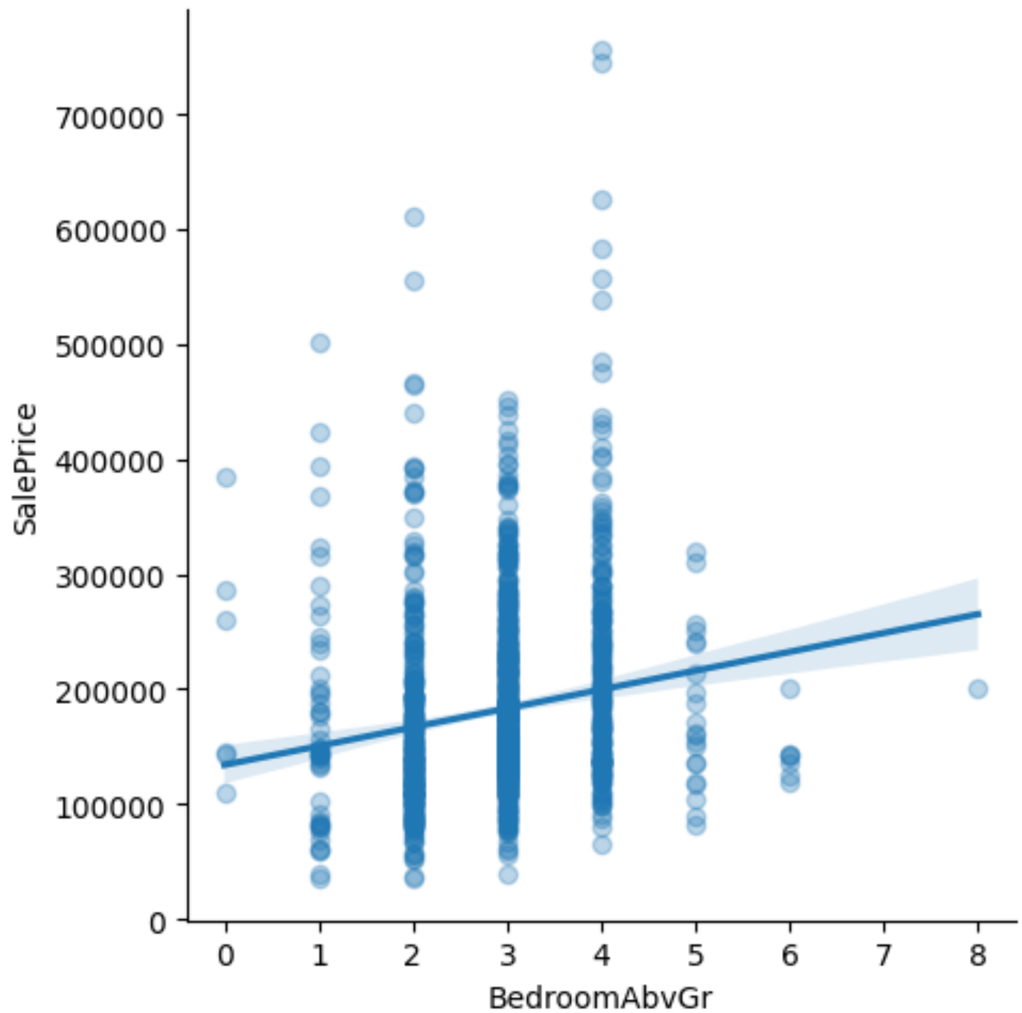
In [41]:
```
sns.lmplot(x='GrLivArea',
           y='SalePrice',
           data=df_train_cleaned,
           scatter_kws={'alpha':0.3})
```

Out[41]: <seaborn.axisgrid.FacetGrid at 0x17727b890>



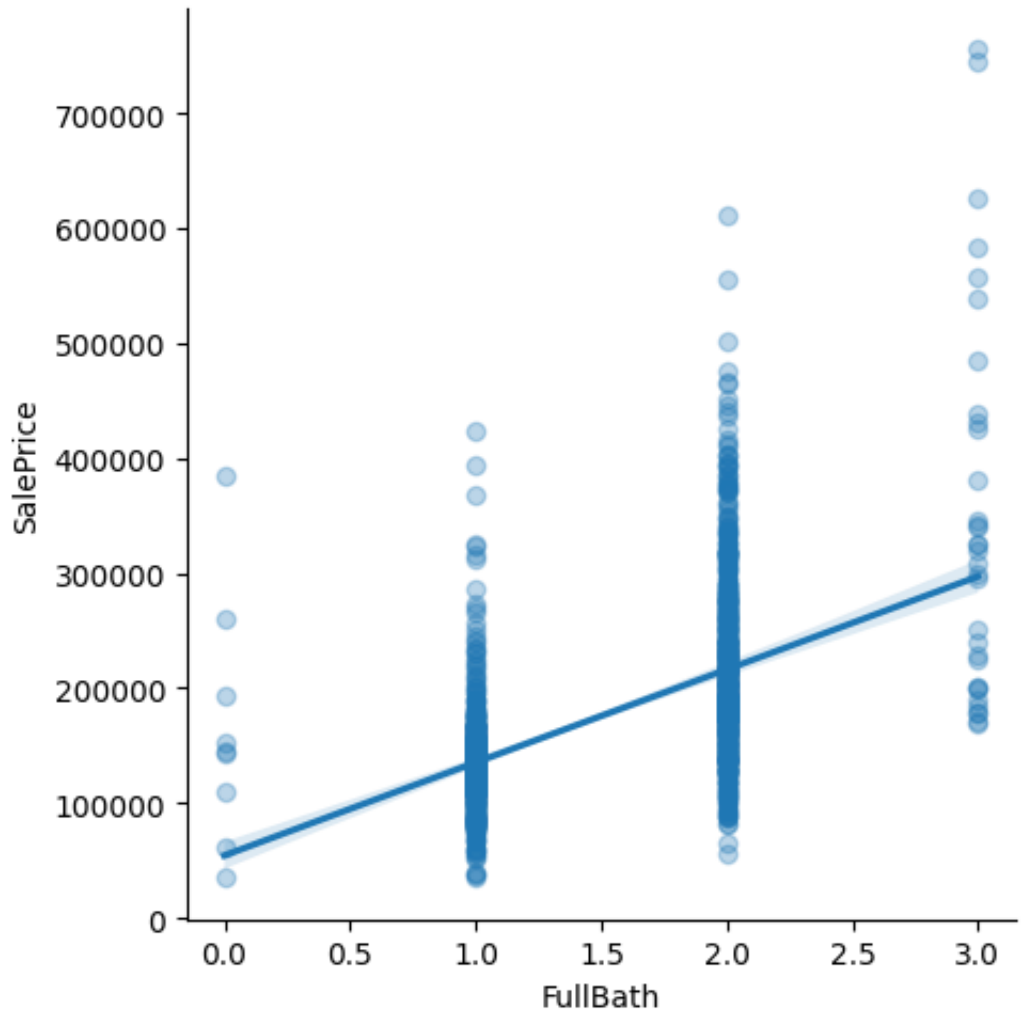
In [42]:
```
sns.lmplot(x='BedroomAbvGr',
           y='SalePrice',
           data=df_train_cleaned,
           scatter_kws={'alpha':0.3})
```

Out[42]: <seaborn.axisgrid.FacetGrid at 0x177079670>



In [43]:
```
sns.lmplot(x='FullBath',
           y='SalePrice',
           data=df_train_cleaned,
           scatter_kws={'alpha':0.3})
```

Out[43]: <seaborn.axisgrid.FacetGrid at 0x176f73ec0>



In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 42)
```

Train model (Linear Regression)

In [45]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
```

In [46]:
```
model.fit(X_train, y_train)
```
Out[46]:
```
▾ LinearRegression ⓘ ?
(https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.Lin
LinearRegression()
```

In [47]:
```
coefficients = model.coef_
```

In [48]:
```
model.score(X, y)
```
Out[48]:
```
0.6541686106425739
```

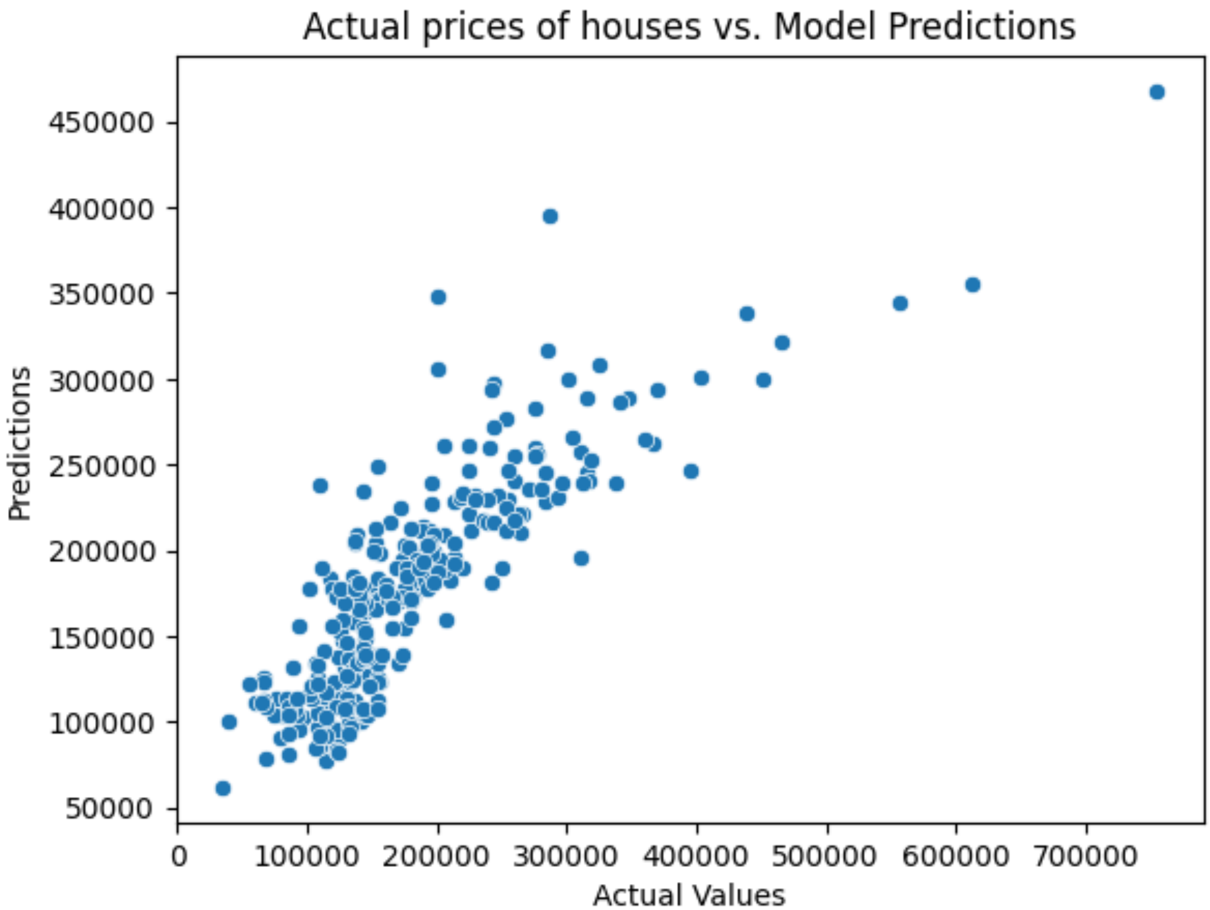
In [49]:
```
for feature, coef in zip(features, coefficients):
    print(f'{feature}: {coef}')
```
```
GrLivArea: 45.45358200866404
1stFlrSF: 70.93123244880826
2ndFlrSF: 19.004245633922277
LowQualFinSF: -44.481896073804585
BsmtFullBath: 22125.53078348177
BsmtHalfBath: 6538.890647874652
FullBath: 34869.22887902238
HalfBath: 22717.7813535538
BedroomAbvGr: -18379.60698218067
```

Predict results

In [50]:
```
predictions = model.predict(X_test)
```

In [51]:
```
np.set_printoptions(threshold=5)  # Print only 5 elements per array
print(predictions)
```
```
[112866.76155431 308640.3343831  120052.08531031 ... 193394.87353075
 122311.0076527   82585.36875711]
```

In [52]:
```
sns.scatterplot(x=y_test, y=predictions)
plt.xlabel('Actual Values')
plt.ylabel('Predictions')
plt.title('Actual prices of houses vs. Model Predictions')
plt.show()
```



Evaluation of the model

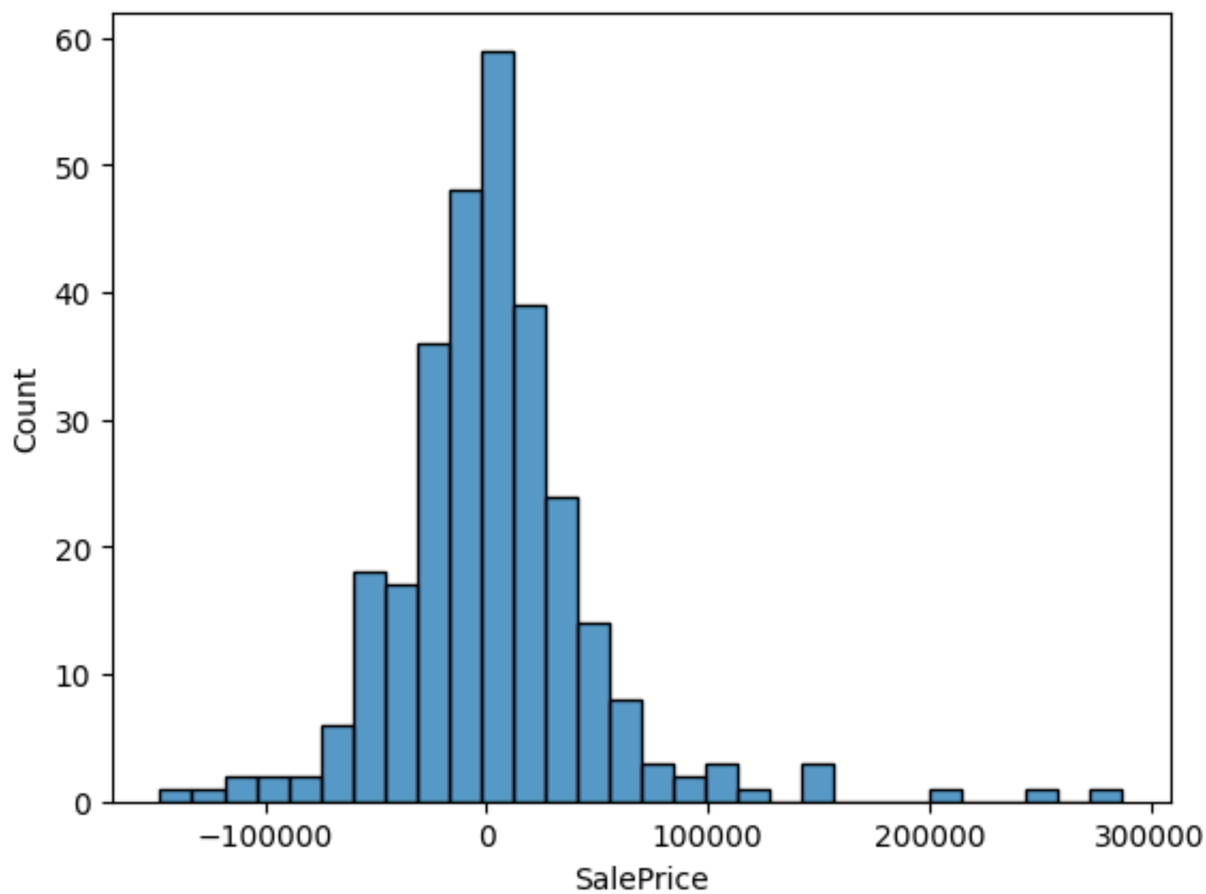
In [53]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
```

In [54]:
```
print('Mean Absolute Error:',mean_absolute_error(y_test, predictions))
print('Mean Squared Error:',mean_squared_error(y_test, predictions))
print('Root Mean Squared Error:',math.sqrt(mean_squared_error(y_test,
predictions)))
```

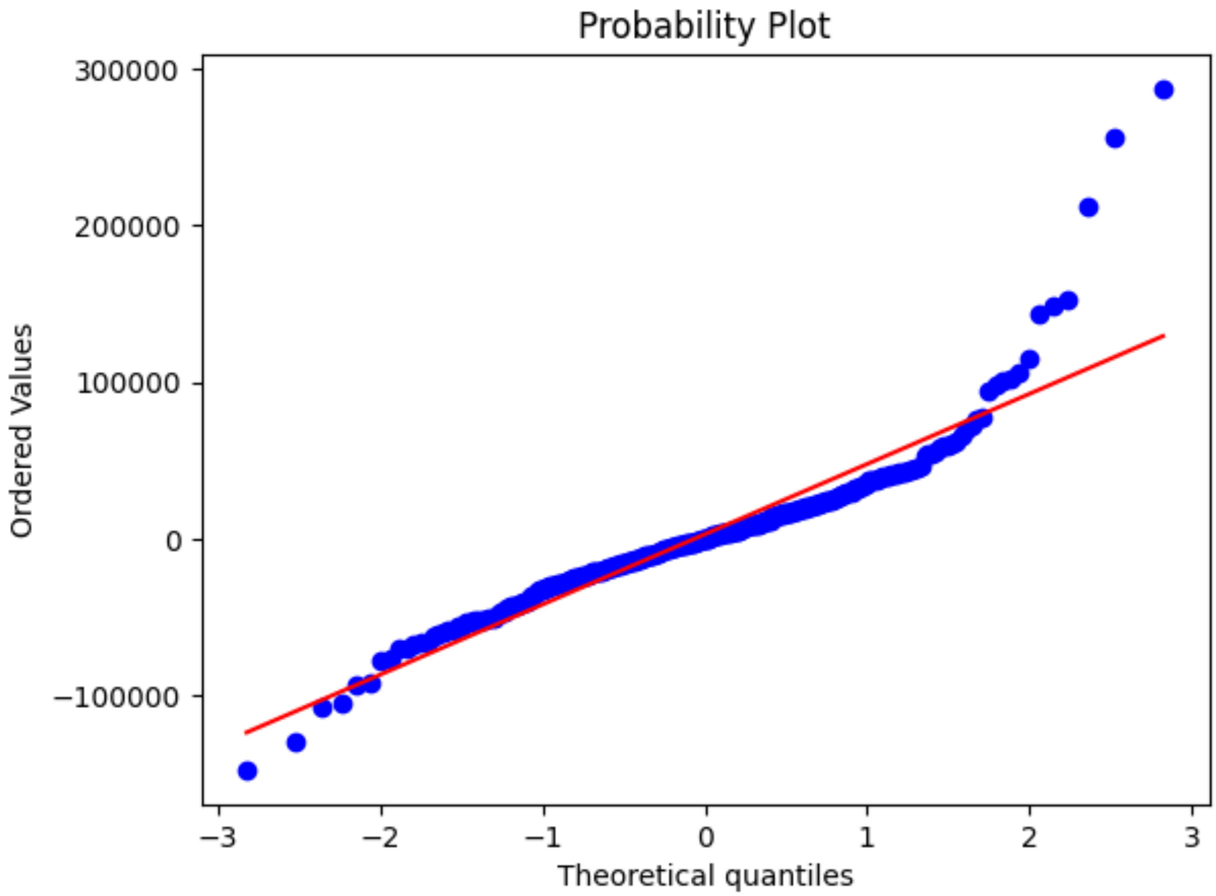
```
Mean Absolute Error: 31410.205502278408
Mean Squared Error: 2269069226.157571
Root Mean Squared Error: 47634.74809587609
```

In [55]:
```
residuals = y_test-predictions
sns.histplot(residuals, bins=30)
```

Out[55]: <Axes: xlabel='SalePrice', ylabel='Count'>



In [56]:
```
import pylab
import scipy.stats as stats

stats.probplot(residuals, dist="norm", plot=pylab)
pylab.show()
```



Use the trained model on our test data

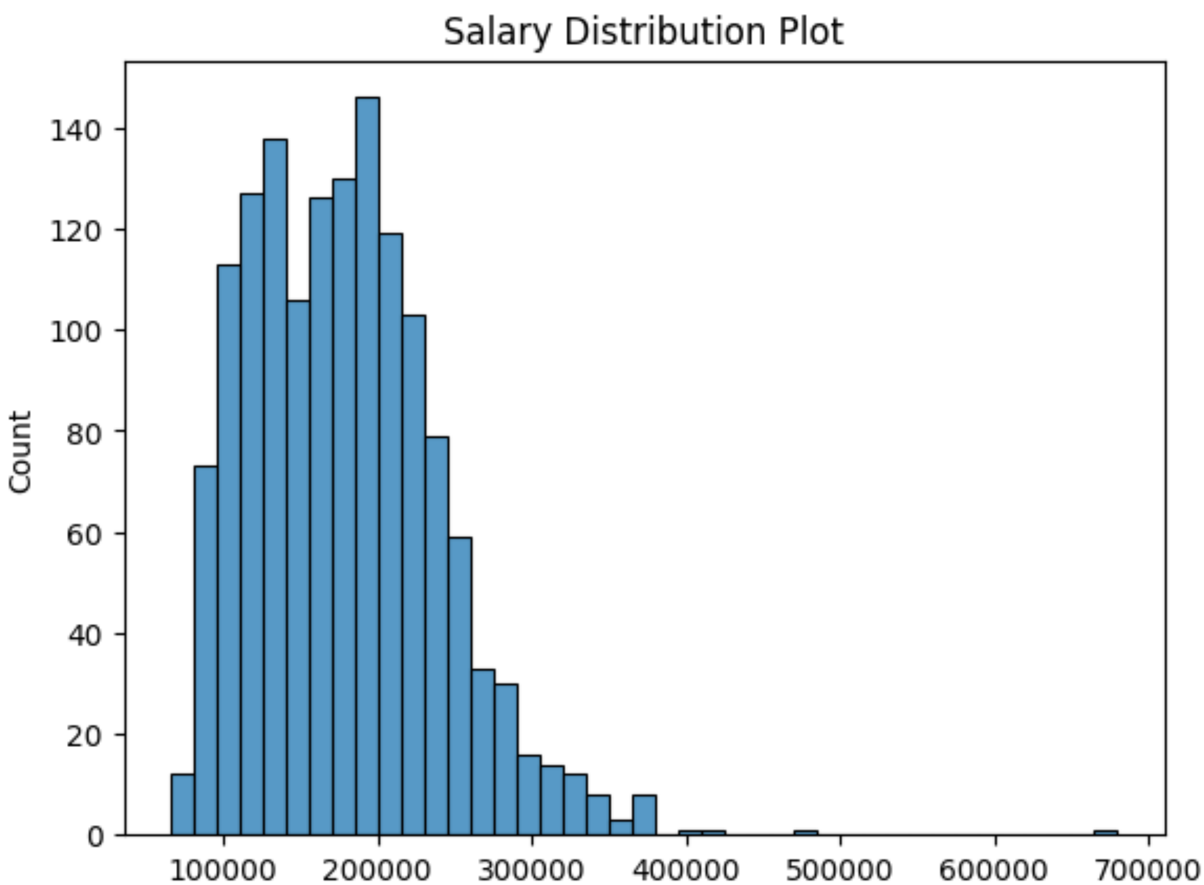
In [57]:
```
Xt = df_test_cleaned[features]
```

In [58]:
```
test_predictions = model.predict(Xt)
```

In [59]:
```
test_predictions
```

Out[59]:
```
array([104689.28980193, 159422.08883339, 192805.94429241, ...,
       128229.8257631 , 101461.04973747, 220250.83345123])
```

In [60]:
```
plt.title('Salary Distribution Plot')
sns.histplot(test_predictions)
plt.show()
```



In [61]:
```
submission_df = pd.DataFrame({
    'Id': df_test_cleaned['Id'],
    'SalePrice': test_predictions
})

# Save the predictions to a CSV file
submission_df.to_csv('submission.csv', index=False)
```

Conclusion

The current linear regression model provides a basic understanding of house price predictions, but the relatively high error metrics indicate room for improvement.