Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, cross_val_score
import plotly.express as px
import plotly.graph_objects as go
```

```
houses_data = pd.read_csv("train.csv")
test_data = pd.read_csv("test.csv")
```

Data Cleaning and Preparation

```
# Display basic information about the dataset
print("Step 2: Data Collection")
print("\nDataset loaded successfully.")
print("\nBasic Information About the Dataset:")
print("Number of Rows:", len(houses_data))
print("Number of Columns:", len(houses_data.columns))
print("\nSample Data (first 5 rows):")
print(houses_data.head())
```

```
Step 2: Data Collection

Dataset loaded successfully.

Basic Information About the Dataset:
Number of Rows: 1460
Number of Columns: 81

Sample Data (first 5 rows):
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
0   1          60       RL         65.0     8450   Pave   NaN      Reg
1   2          20       RL         80.0     9600   Pave   NaN      Reg
2   3          60       RL         68.0    11250   Pave   NaN      IR1
3   4          70       RL         60.0     9550   Pave   NaN      IR1
4   5          60       RL         84.0    14260   Pave   NaN      IR1

  LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal MoSold  \
0         Lvl    AllPub  ...        0    NaN   NaN         NaN       0      2
1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0      5
2         Lvl    AllPub  ...        0    NaN   NaN         NaN       0      9
3         Lvl    AllPub  ...        0    NaN   NaN         NaN       0      2
4         Lvl    AllPub  ...        0    NaN   NaN         NaN       0     12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]
```

```
houses_data.info()
```

```
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 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
```

```
houses_data.describe()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	14
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	19
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	18
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	19
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	19
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	20
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	20

8 rows × 38 columns

```
houses_data.isnull().sum()
```

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

```
# Extract columns with null values
columns_with_null = houses_data.columns[houses_data.isnull().any()]

# Display columns with null values
print("Columns with null values:")
for col in columns_with_null:
    print(col)
```

```
Columns with null values:
LotFrontage
Alley
MasVnrType
MasVnrArea
BsmtQual
BsmtCond
BsmtExposure
BsmtFinType1
BsmtFinType2
Electrical
FireplaceQu
GarageType
GarageYrBlt
GarageFinish
GarageQual
GarageCond
PoolQC
Fence
MiscFeature
```

```
# Check for missing values in each column
missing_values = houses_data.isnull().sum()

# Print columns with missing values and their corresponding counts
columns_with_missing_values = missing_values[missing_values > 0]
print("\nColumns with Missing Values:")
print(columns_with_missing_values)
```

```

Columns with Missing Values:
LotFrontage      259
Alley           1369
MasVnrType       872
MasVnrArea         8
BsmtQual          37
BsmtCond          37
BsmtExposure      38
BsmtFinType1      37
BsmtFinType2      38
Electrical         1
FireplaceQu      690
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
GarageCond        81
PoolQC          1453
Fence           1179
MiscFeature     1406
dtype: int64
```

```
# Check for duplicate rows
duplicates_before = houses_data.duplicated().sum()

# Remove duplicate rows
houses_data.drop_duplicates(inplace=True)

# Check for duplicate rows after removal
duplicates_after = houses_data.duplicated().sum()

# Print the results
if duplicates_before > 0:
    print(f"Handling Duplicates\n{duplicates_before} duplicate row(s) were found and removed.")
else:
    print("Handling Duplicates\nNo duplicate rows found in the dataset.")
```

```
Handling Duplicates
No duplicate rows found in the dataset.
```

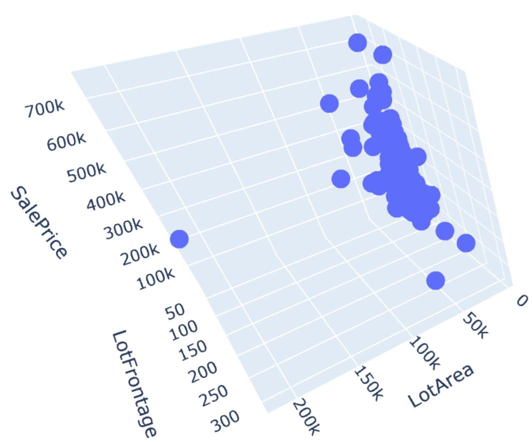
```
# Get the column names and data types
column_info = houses_data.dtypes

# Display column names and data types horizontally
for col_name, data_type in column_info.items():  # Use items() instead of iteritems()
    print(f"{col_name}: {data_type}\t", end='')
```

```
Id: int64	MSSubClass: int64	MSZoning: object	LotFrontage: float64	LotArea: int64	Street: object	Alley: object
```

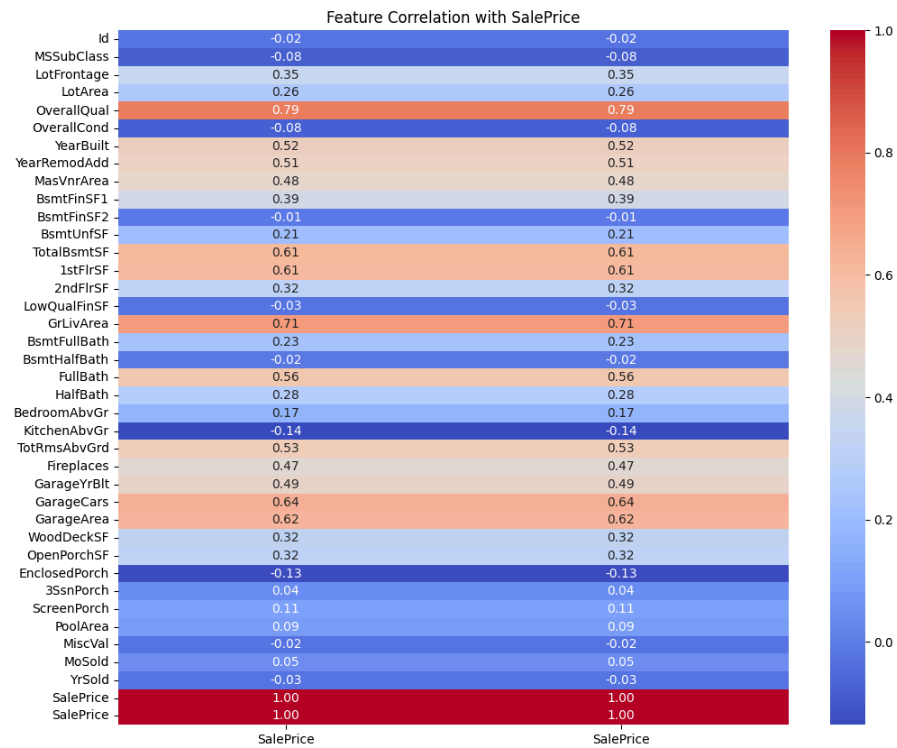
```
houses_data.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

```
fig = px.scatter_3d(houses_data,x = 'LotArea', y='LotFrontage',z = 'SalePrice')
fig.show()
```

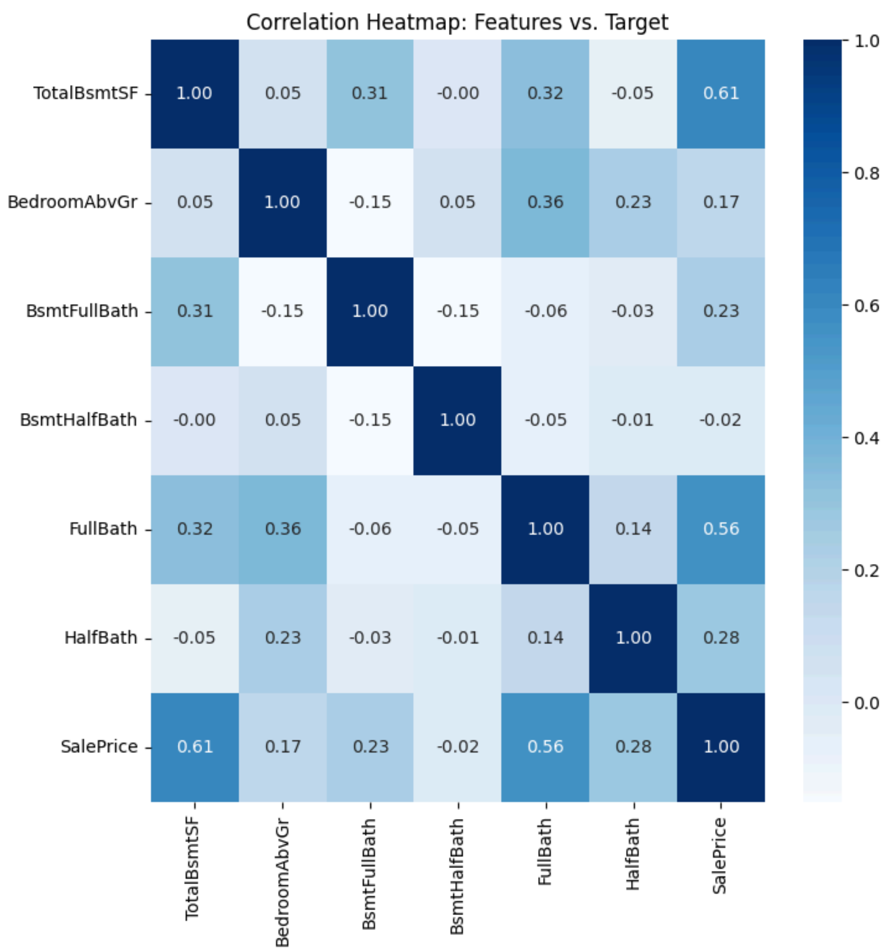


```
#  list of categorical columns to exclude
categorical_columns = ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Cor

# Select numerical columns
numerical_columns = [col for col in houses_data.columns if col not in categorical_columns]

# Create a DataFrame with only numerical features and the target variable
numerical_data = houses_data[numerical_columns + ['SalePrice']]

# Calculate the correlation matrix
correlation_matrix = numerical_data.corr()

# Step 2: Generate a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix[['SalePrice']], annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Feature Correlation with SalePrice")
plt.show()
```




```
# Split the data into training and testing sets
X = houses_data[['TotalBsmtSF', 'BedroomAbvGr', 'BsmtFullBath', 'BsmtHalfBath',"FullBath", "HalfBath"]]
y = houses_data['SalePrice']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Select the columns of interest (features and target)
features = houses_data[['TotalBsmtSF', 'BedroomAbvGr', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath']]
target = houses_data[['SalePrice']]

# Create a new DataFrame with only the selected columns
data_subset = pd.concat([features, target], axis=1)  # Use square brackets and specify axis=1

# Calculate the correlation matrix
correlation_matrix = data_subset.corr()

# Create a heatmap for the correlation matrix
plt.figure(figsize=(8, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='Blues', fmt=".2f")
plt.title("Correlation Heatmap: Features vs. Target")
plt.show()
```




```
# Check for missing values in the training dataset
missing_train = houses_data.isnull().sum()
print("Missing Values in Training Data:")
print(missing_train)

# Check for missing values in the testing dataset
missing_test = test_data.isnull().sum()
print("\nMissing Values in Testing Data:")
print(missing_test)
```

```
Missing Values in Training Data:
Id               0
MSSubClass       0
MSZoning         0
LotFrontage    259
LotArea          0
              ...
MoSold           0
```

```
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
Length: 81, dtype: int64

Missing Values in Testing Data:
Id                0
MSSubClass        0
MSZoning          4
LotFrontage     227
LotArea           0
               ...
MiscVal           0
MoSold            0
YrSold            0
SaleType          1
SaleCondition     0
Length: 80, dtype: int64
```

```
# Missing values in selected features
features_missing_values = X.isnull().sum()
features_missing_values
```

```
TotalBsmtSF     0
BedroomAbvGr    0
BsmtFullBath    0
BsmtHalfBath    0
FullBath        0
HalfBath        0
dtype: int64
```

```
#create a linear regression model
model = LinearRegression()


# Fit the model to the training data
model.fit(X_train, y_train)
print(model)
```

```
LinearRegression()
```

```
# Make predictions on the test data
y_pred = model.predict(X_test)


# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)


print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Mean Squared Error: 2693215363.50
R-squared: 0.65
```

```
X.sample(5)
```

	TotalBsmtSF	BedroomAbvGr	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath
1024	1565	2	1	0	2	0
607	896	3	1	0	3	0
1176	876	3	1	0	1	0
1428	788	2	1	0	1	0
914	612	2	0	0	2	1

```
# Predict the price of a new house
new_house = np.array([[2500, 3, 1,0,2,1]])
predicted_price = model.predict(new_house)
print(f"Predicted Price for the New House: ${predicted_price[0]:.2f}")
```

```
Predicted Price for the New House: $354103.01
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning:
```

```
X does not have valid feature names, but LinearRegression was fitted with feature names
```

```
# Cross-validation to assess model performance
cv_scores = cross_val_score(model, X, y, cv=5)
print('Cross-Validation Scores:', cv_scores)
print('Mean CV Score:', cv_scores.mean())
```

```
Cross-Validation Scores: [0.60009438 0.65556307 0.63763301 0.61946238 0.44203474]
Mean CV Score: 0.5909575152454488
```

```
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs. Predicted Prices")
plt.show()
```