In [ ]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Disable scientific notation for large numbers
pd.options.display.float_format = '{:.0f}'.format

# Setting display options for Pandas to show three decimal places for floati
pd.set_option('display.float_format', lambda x: '%.2f' % x)
```

Data Loading

In [ ]:
```
# import data
house_prices_df = pd.read_csv('/content/drive/MyDrive/House_prices.csv')
```

Data Exploration

In [ ]:
```
house_prices_df.info()  # Display information about the DataFrame, including
```

In [ ]:
```
house_prices_df.head()  # Display top 5 records
```

In [ ]:
```
house_prices_df[['Size','Bedrooms','Bathrooms','YearBuilt','Price']].describ
```

Data Cleaning

In [ ]:
```
house_prices_df.isna().sum()  # Find sum of missing values
```

Since sum of missing values is zero, so there is no need of replacing null values.

In [ ]:
```
print(house_prices_df.duplicated().sum()) # Find sum of duplicate records
```

Since, sum of duplicated values is zero, so there is no need to drop duplicates.

Data Preprocessing

Feature scaling

In [ ]:
```
# Select features to scale
features = ['Size', 'Bedrooms']

# Apply Min-Max Scaling
min_max_scaler = MinMaxScaler()
house_prices_df[features] = min_max_scaler.fit_transform(house_prices_df[fea
```

In [ ]:
```
house_prices_df[features].head()  # Print top 5 features
```

Encode Categorical Features

In [ ]:
```
# One-Hot Encoding for Location
one_hot_encoded = pd.get_dummies(house_prices_df['Location'], prefix='Locati

# Add the new columns to the original DataFrame
house_prices_df = pd.concat([house_prices_df, one_hot_encoded], axis=1)
```

In [ ]:
```
# Save cleaned data
house_prices_df.to_csv('/content/drive/MyDrive/House_prices_final.csv', inde
```

Analyze Predictors

In [ ]:
```
# Compute correlation matrix for all numerical features with Price
correlations = house_prices_df.corr(numeric_only=True)['Price'].sort_values(

# Display the correlations
print(correlations)
```

Model training and evaluation

Train a Linear Regression Model and evaluate model performance

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [ ]:
```
import numpy as np

# Select features and target variable
X = house_prices_df.drop(['Id','Location','Price', 'Condition', 'Garage'], a
y = house_prices_df['Price']

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mse)
```

```
# Print Evaluation Metrics
print(f'Root Mean Squared Error(RMSE): {rmse:.2f}')
print(f'R^2 Score: {r2:.4f}')
```

Insights

Predict outputs for the test data including predicted vs. actual prices

In [ ]:
```
# Create a DataFrame with actual and predicted prices
results = pd.DataFrame({'Actual': y_test.values, 'Predicted': y_pred})

# Display the first 10 rows
print(results.head(10))
```

Summary of the most important predictors influencing house prices.

In [ ]:
```
# Remove the target variable itself and sort by absolute correlation
correlations_filtered = correlations.drop('Price').abs().sort_values(ascendi

# Select predictors with absolute correlation above 0.05
important_predictors = correlations_filtered[correlations_filtered > 0.05]

print(important_predictors)
```