In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
df = pd.read_csv("house_price_data.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	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 [5]:

```
df.tail()
```

Out[5]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCo
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	

5 rows × 81 columns

In [6]:

```
df.shape
```

Out[6]:

```
(1460, 81)
```

In [7]:

```
df.columns
```

Out[7]:

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

In [8]:

```
df.duplicated().sum()
```

Out[8]:

```
0
```

In [9]:

```
df = df.drop_duplicates()
```

In [10]:

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

Out[10]:

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

In [11]:

```
null_counts = df.isnull().sum()
```

In [12]:

```
features_with_null = null_counts[null_counts > 0].index
print(features_with_null)
```

```
Index(['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Electrical', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence',
       'MiscFeature'],
      dtype='object')
```

In [13]:

```
null_counts = df[['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual', 'BsmtC
                  'BsmtFinType1', 'BsmtFinType2', 'Electrical', 'FireplaceQu', 'GarageTy
                  'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence', 'MiscFe
print(null_counts)
```

```
LotFrontage      259
Alley           1369
MasVnrType         8
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
```

In [14]:

```
numeric_features = ['LotFrontage', 'MasVnrArea', 'GarageYrBlt']
for feature in numeric_features:
    df[feature].fillna(df[feature].mean(), inplace=True)

categorical_features = ['Alley', 'MasVnrType', 'BsmtQual', 'BsmtCond', 'BsmtExposure',
                        'BsmtFinType1', 'BsmtFinType2', 'Electrical', 'FireplaceQu',
                        'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond',
                        'PoolQC', 'Fence', 'MiscFeature']
for feature in categorical_features:
    df[feature].fillna('None', inplace=True)
```

In [15]:

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

Out[15]:

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

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1460 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          1460 non-null   object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     1460 non-null   object
 26  MasVnrArea     1460 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1460 non-null   object
 31  BsmtCond       1460 non-null   object
 32  BsmtExposure   1460 non-null   object
 33  BsmtFinType1   1460 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1460 non-null   object
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 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     1460 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    1460 non-null   object
 58  GarageType     1460 non-null   object
 59  GarageYrBlt    1460 non-null   float64
 60  GarageFinish   1460 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1460 non-null   object
 64  GarageCond     1460 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         1460 non-null   object
 73  Fence          1460 non-null   object
 74  MiscFeature    1460 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: 935.3+ KB
```

In [17]:

```
object_columns = df.select_dtypes(include='object').columns.tolist()
numerical_columns = df.select_dtypes(include=['int', 'float']).columns.tolist()

print("Object columns:", object_columns)
print('\n')
print("Numerical columns:", numerical_columns)
```

```
Object columns: ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour',
'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Cond
ition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'Bsmt
Qual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heatin
g', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional',
'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond',
'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleConditio
n']


Numerical columns: ['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'Overall
Qual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinS
F1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'Lo
wQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'Hal
fBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'Gar
ageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'Enclo
sedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'Y
rSold', 'SalePrice']
```

In [18]:

```
df.describe()
```

Out[18]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	146
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	197
std	421.610009	42.300571	22.024023	9981.264932	1.382997	1.112799	3
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	187
25%	365.750000	20.000000	60.000000	7553.500000	5.000000	5.000000	195
50%	730.500000	50.000000	70.049958	9478.500000	6.000000	5.000000	197
75%	1095.250000	70.000000	79.000000	11601.500000	7.000000	6.000000	200
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	201

8 rows × 38 columns

In [19]:

```
df.nunique()
```

Out[19]:

```
Id               1460
MSSubClass         15
MSZoning            5
LotFrontage       111
LotArea          1073
                 ... 
MoSold             12
YrSold              5
SaleType            9
SaleCondition       6
SalePrice         663
Length: 81, dtype: int64
```

In [20]:

```python
for i in object_columns:
    print(i)
    print(df[i].unique())
    print('\n')
```

```
MSZoning
['RL' 'RM' 'C (all)' 'FV' 'RH']


Street
['Pave' 'Grvl']


Alley
['None' 'Grvl' 'Pave']


LotShape
['Reg' 'IR1' 'IR2' 'IR3']


LandContour
['Lvl' 'Bnk' 'Low' 'HLS']
```

In [21]:

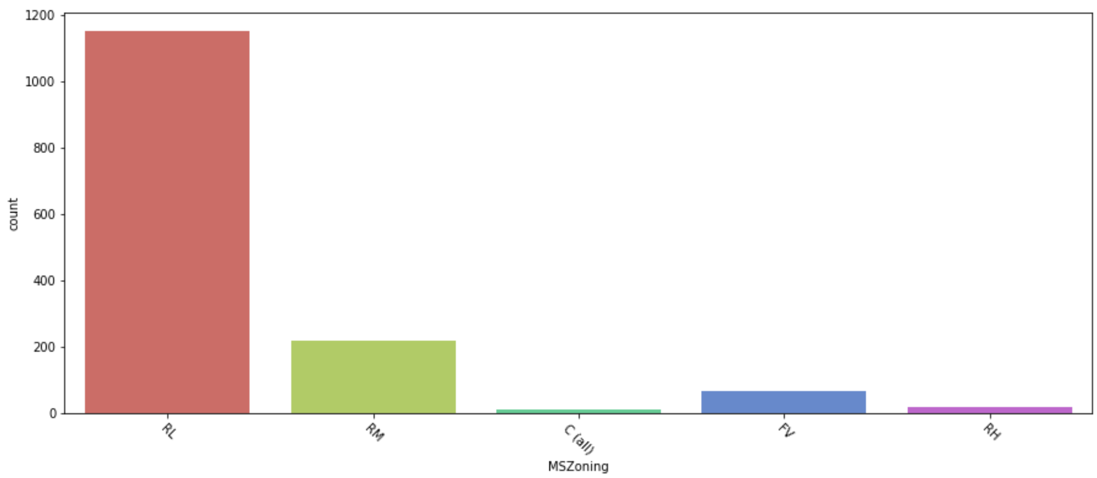
```python
for i in object_columns:
    print(i)
    print(df[i].value_counts())
    print('\n')
```

```
MSZoning
RL         1151
RM          218
FV           65
RH           16
C (all)      10
Name: MSZoning, dtype: int64


Street
Pave    1454
Grvl       6
Name: Street, dtype: int64


Alley
None    1369
Grvl      50
Pave      41
Name: Alley, dtype: int64
```

In [22]:

```
for i in object_columns:
    print('Countplot for:', i)
    plt.figure(figsize=(15,6))
    sns.countplot(df[i], data = df, palette = 'hls')
    plt.xticks(rotation = -45)
    plt.show()
    print('\n')
```

```
Countplot for: MSZoning
```

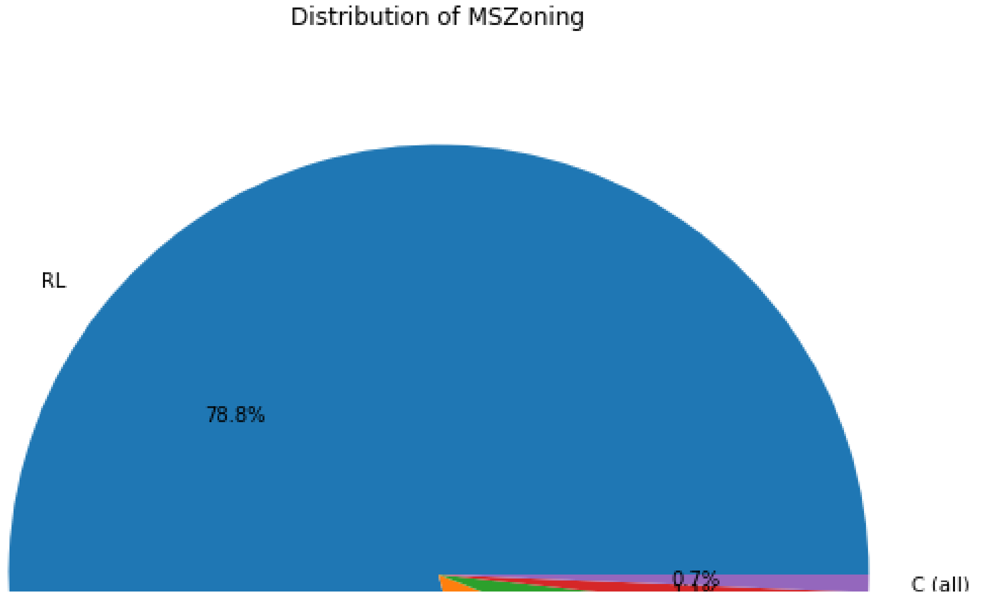



```
Countplot for: Street
```

In [23]:

```
for i in object_columns:
    print('Pie plot for:', i)
    plt.figure(figsize=(20, 10))
    df[i].value_counts().plot(kind='pie', autopct='%1.1f%%')
    plt.title('Distribution of ' + i)
    plt.ylabel('')
    plt.show()
    print('\n')
```

```
Pie plot for: MSZoning
```



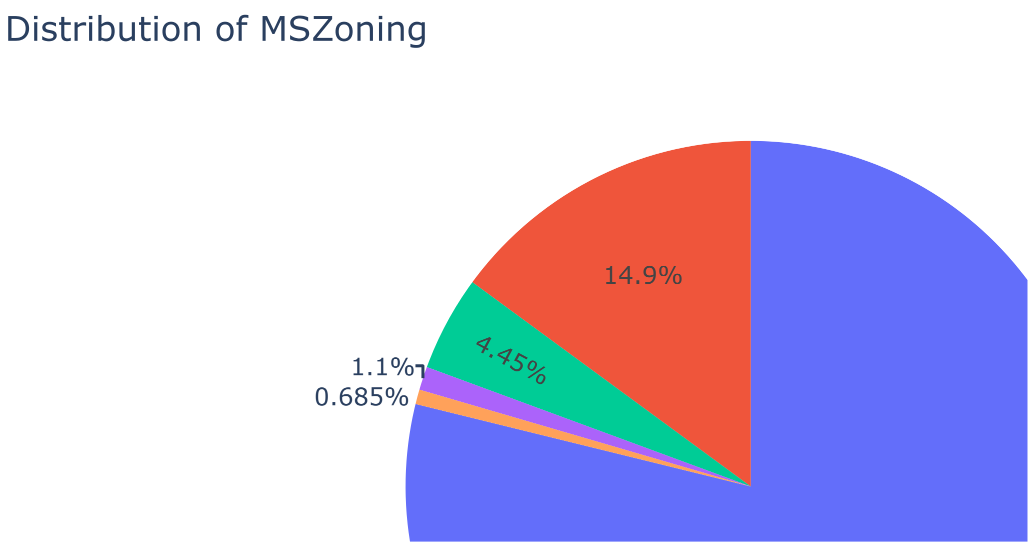
In [24]:

```
for i in object_columns:
    fig = go.Figure(data=[go.Bar(x=df[i].value_counts().index, y=df[i].value_counts())])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Count")
    fig.show()
```

In [25]:

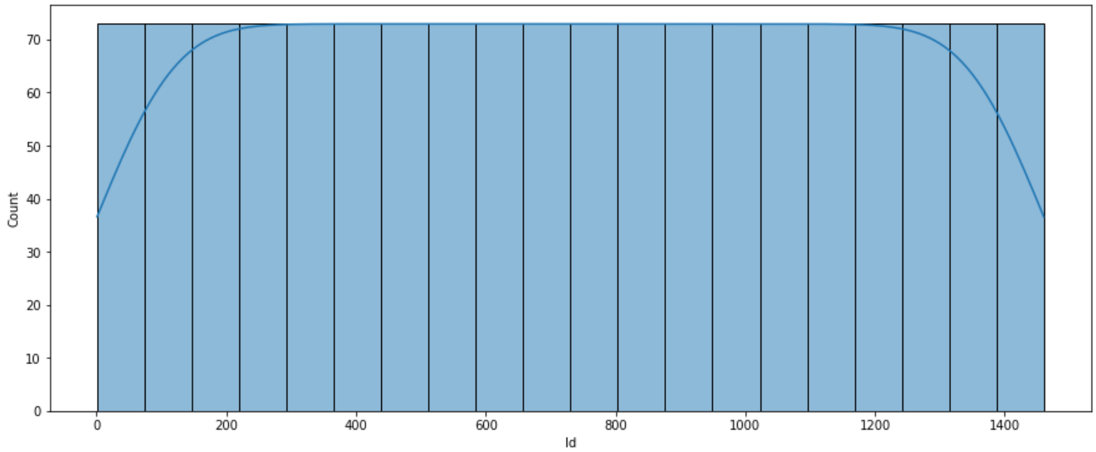
```
for i in object_columns:
    print('Pie plot for:', i)
    fig = px.pie(df, names=i, title='Distribution of ' + i)
    fig.show()
    print('\n')
```

```
Pie plot for: MSZoning
```



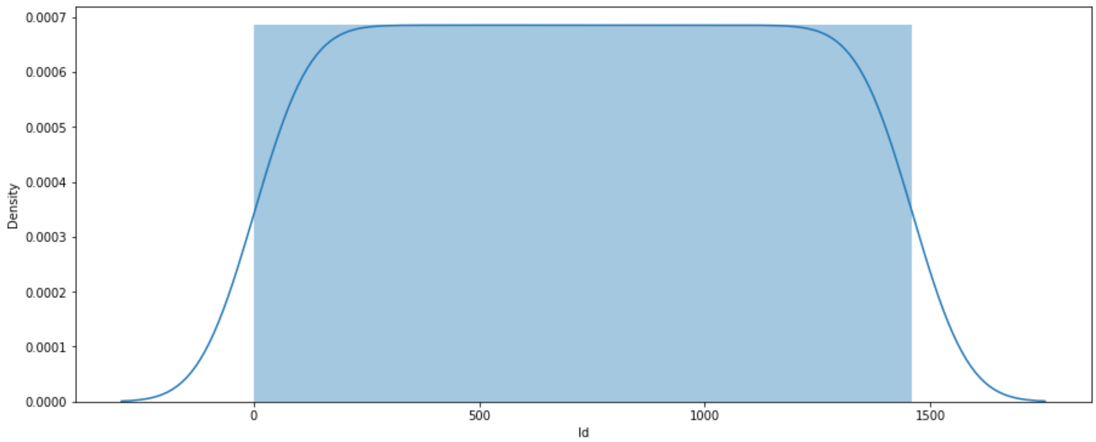
In [26]:

```
for i in numerical_columns:
    plt.figure(figsize=(15,6))
    sns.histplot(df[i], kde = True, bins = 20, palette = 'hls')
    plt.xticks(rotation = 0)
    plt.show()
```



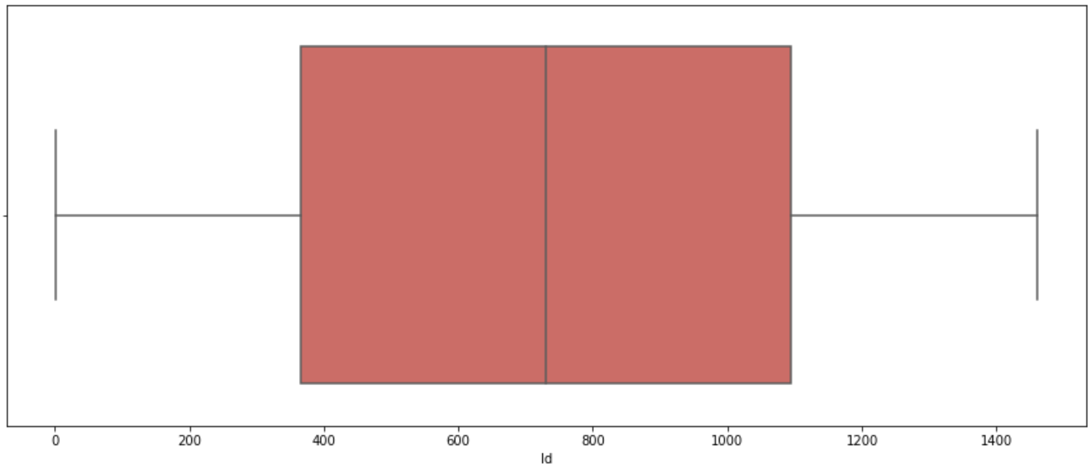
In [27]:

```
for i in numerical_columns:
    plt.figure(figsize=(15,6))
    sns.distplot(df[i], kde = True, bins = 20)
    plt.xticks(rotation = 0)
    plt.show()
```



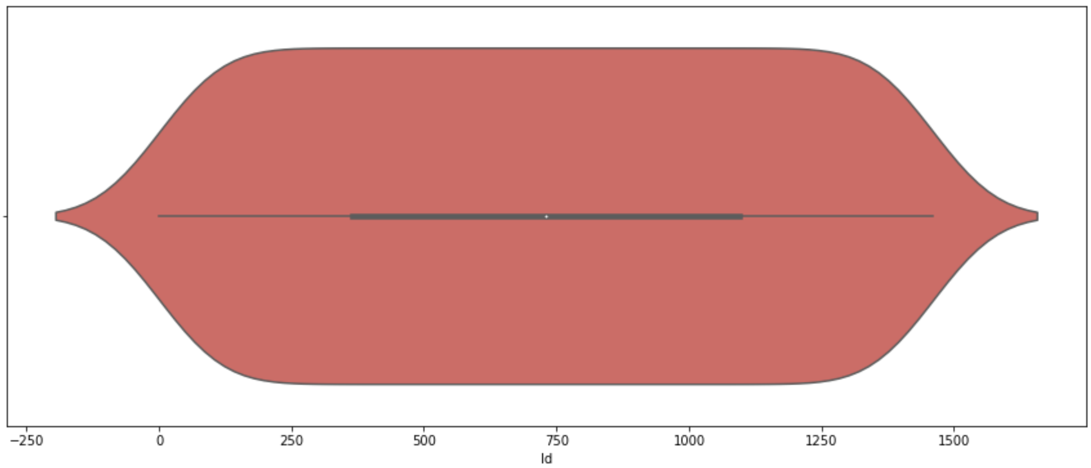
In [28]:

```python
for i in numerical_columns:
    plt.figure(figsize=(15,6))
    sns.boxplot(df[i], data=df, palette='hls')
    plt.xticks(rotation = 0)
    plt.show()
```



In [29]:

```python
for i in numerical_columns:
    plt.figure(figsize=(15,6))
    sns.violinplot(df[i], data=df, palette='hls')
    plt.xticks(rotation = 0)
    plt.show()
```

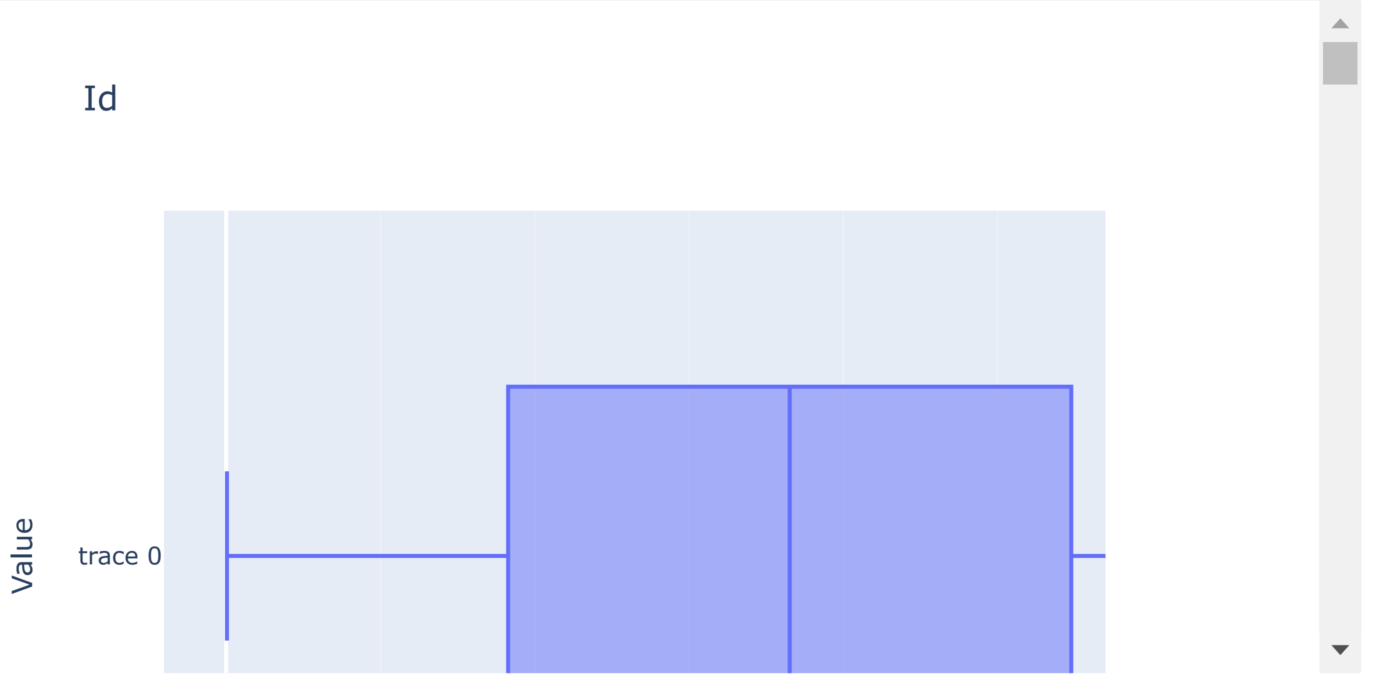


In [30]:

```
for i in numerical_columns:
    fig = go.Figure(data=[go.Histogram(x=df[i], nbinsx=20)])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Count")
    fig.show()
```

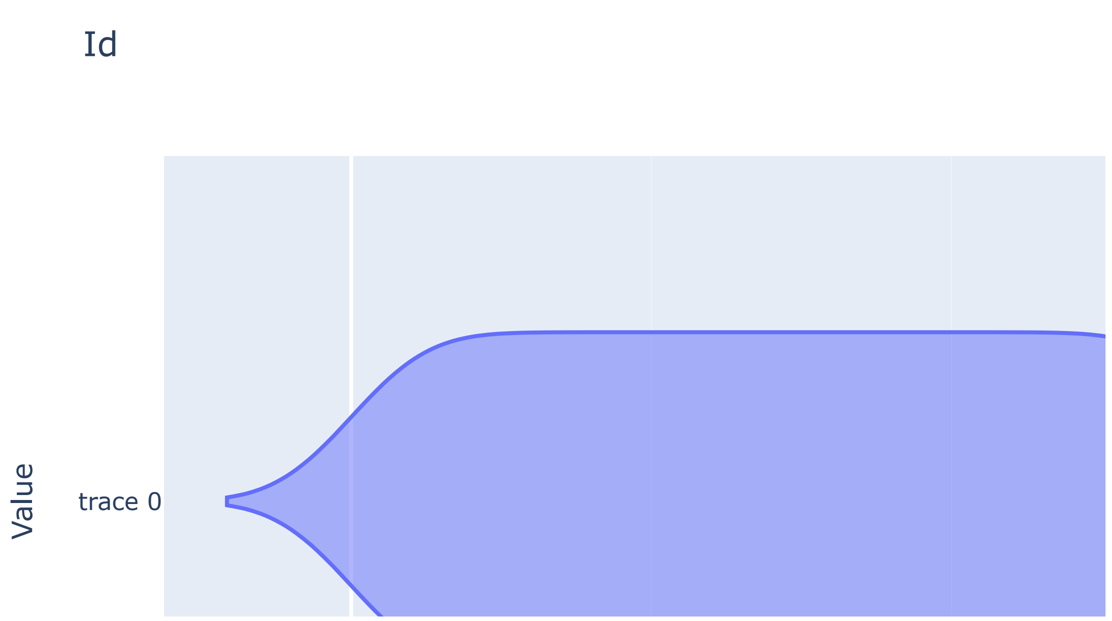
In [31]:

```
for i in numerical_columns:
    fig = go.Figure(data=[go.Box(x=df[i])])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Value")
    fig.show()
```



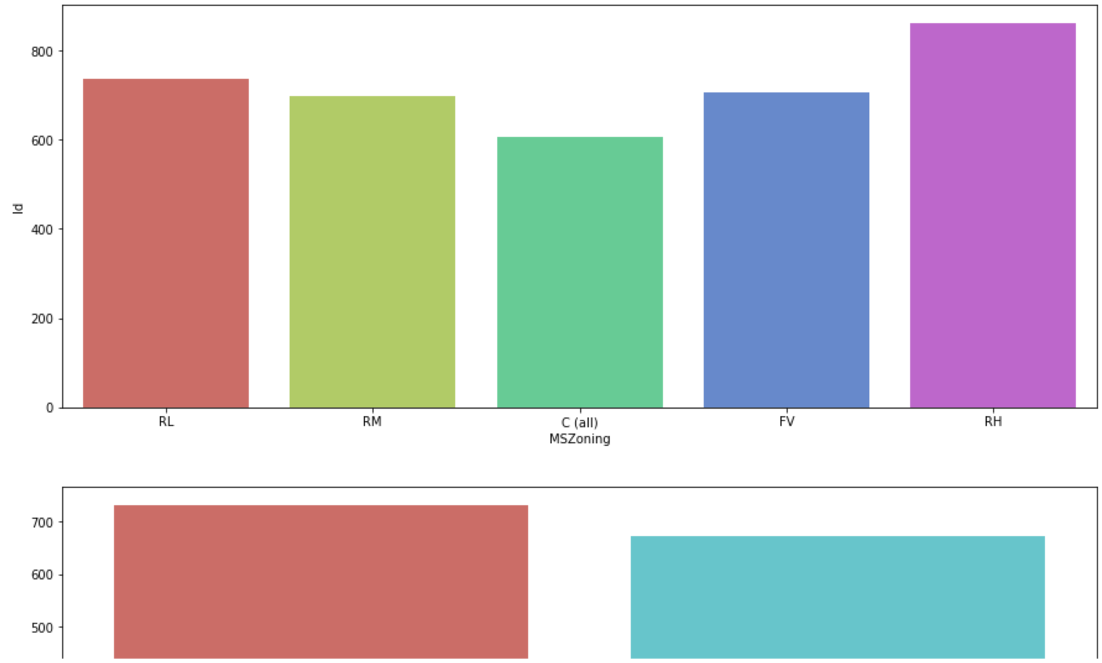
In [32]:

```
for i in numerical_columns:
    fig = go.Figure(data=[go.Violin(x=df[i])])
    fig.update_layout(
        title=i,
        xaxis_title=i,
        yaxis_title="Value")
    fig.show()
```



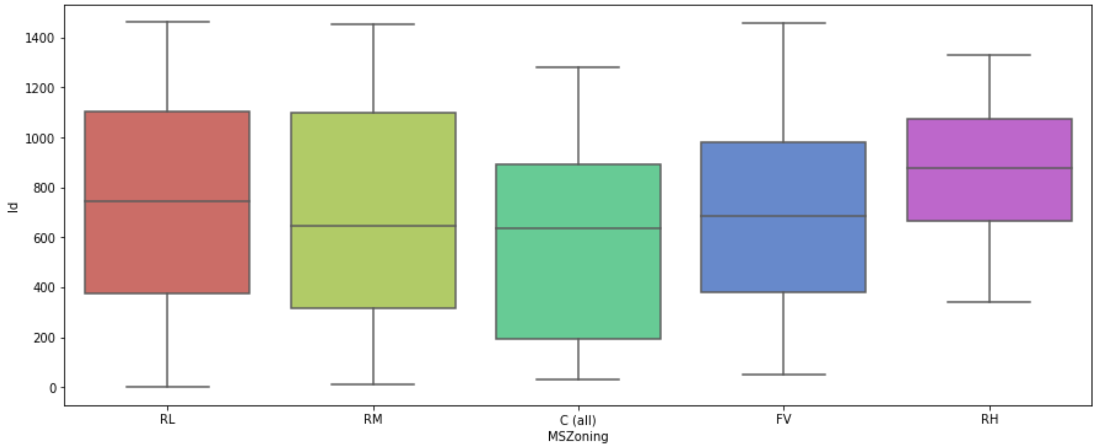
In [33]:

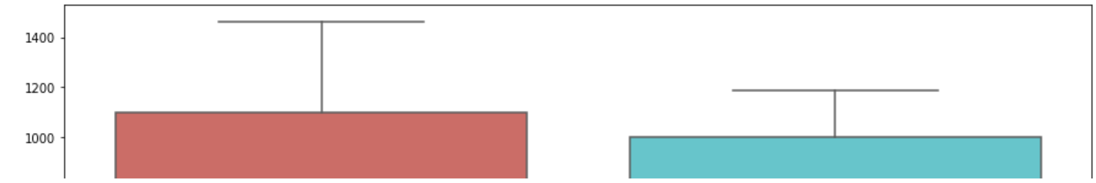
```
for i in numerical_columns:
    for j in object_columns:
        plt.figure(figsize=(15,6))
        sns.barplot(x = df[j], y = df[i], data = df, ci = None, palette = 'hls')
        plt.show()
```



In [34]:

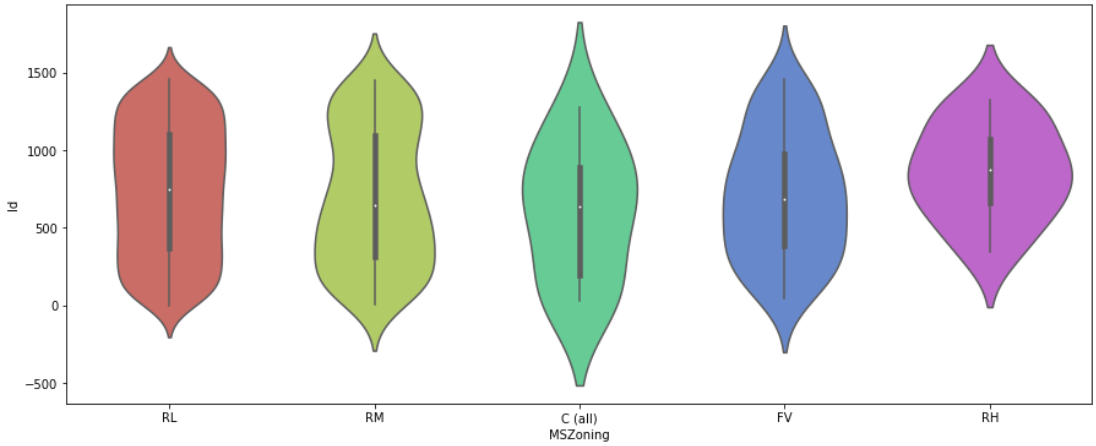
```
for i in numerical_columns:
    for j in object_columns:
        plt.figure(figsize=(15,6))
        sns.boxplot(x = df[j], y = df[i], data = df, palette = 'hls')
        plt.show()
```

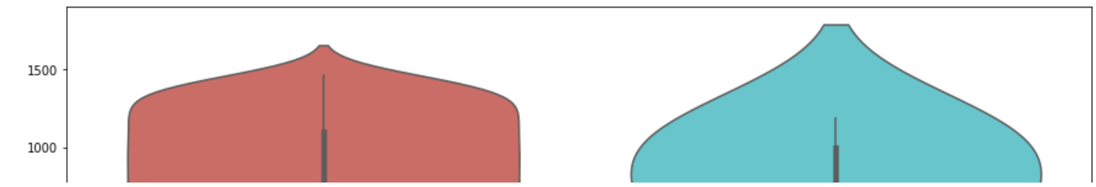




In [35]:

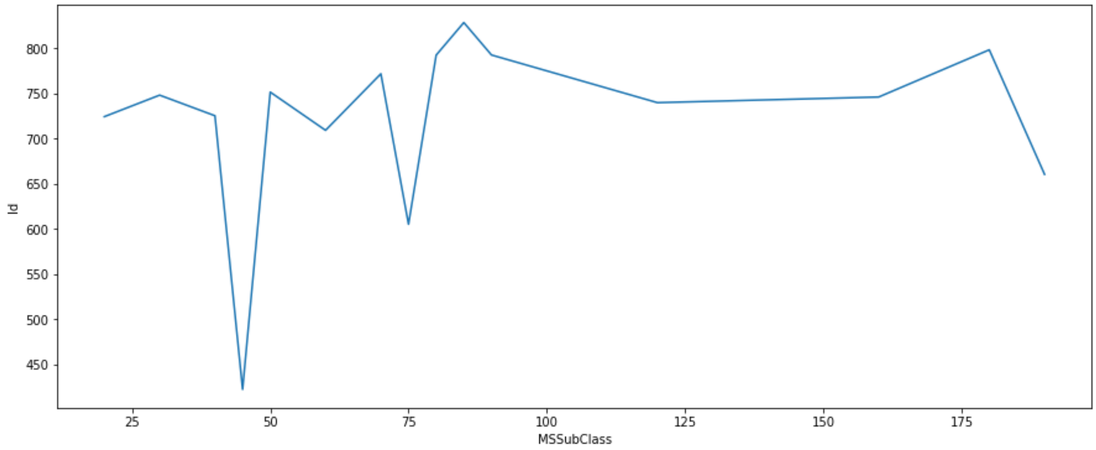
```
for i in numerical_columns:
    for j in object_columns:
        plt.figure(figsize=(15,6))
        sns.violinplot(x = df[j], y = df[i], data = df, palette = 'hls')
        plt.show()
```

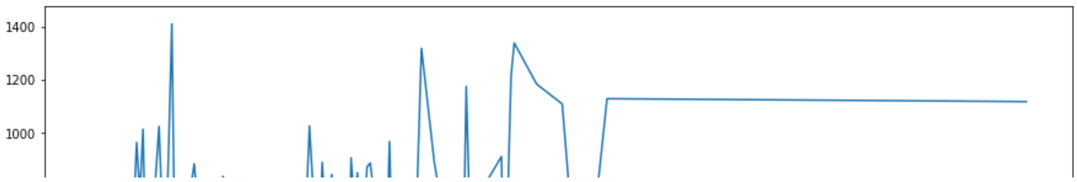




In [36]:

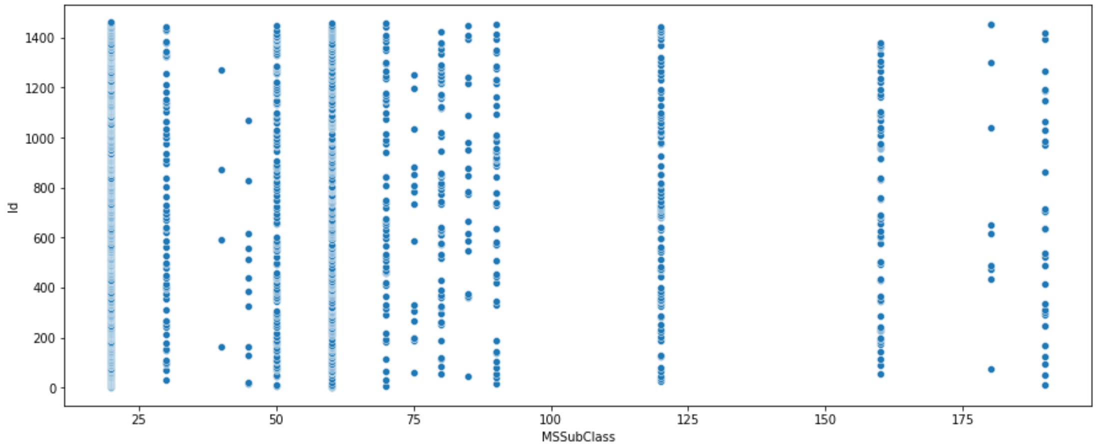
```
for i in numerical_columns:
    for j in numerical_columns:
        if i != j:
            plt.figure(figsize=(15,6))
            sns.lineplot(x = df[j], y = df[i], data = df, ci = None, palette = 'hls')
            plt.show()
```

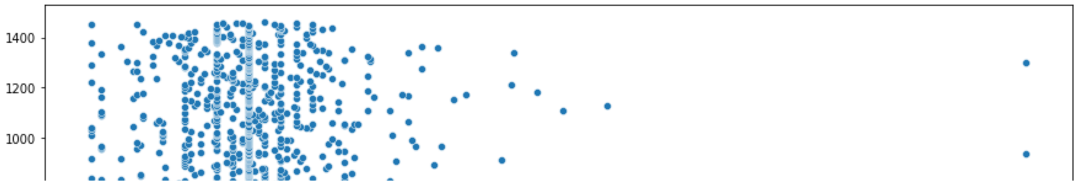




In [37]:

```
for i in numerical_columns:
    for j in numerical_columns:
        if i != j:
            plt.figure(figsize=(15,6))
            sns.scatterplot(x = df[j], y = df[i], data = df, palette = 'hls')
            plt.show()
```





In [38]:

```
corr = df.corr()
```

In [39]:

```
corr
```

Out[39]:

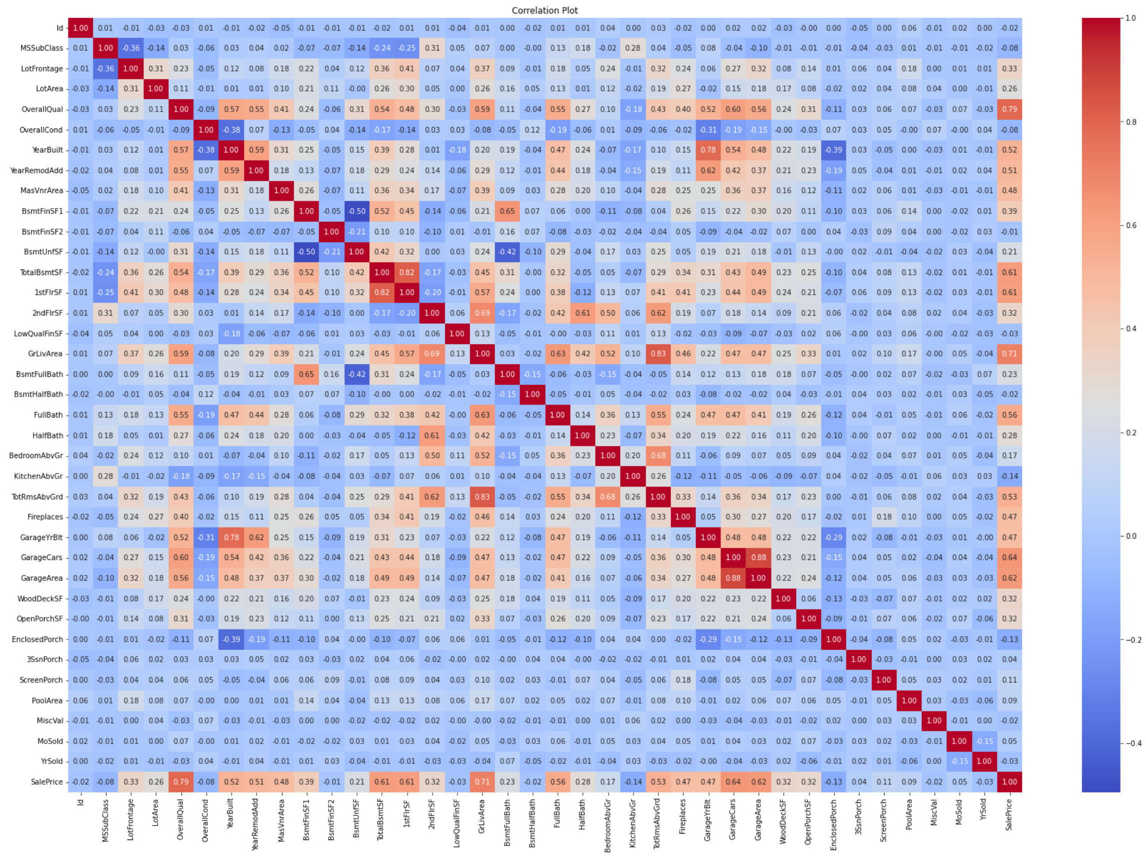
	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Y
Id	1.000000	0.011156	-0.009601	-0.033226	-0.028365	0.012609	-(
MSSubClass	0.011156	1.000000	-0.357056	-0.139781	0.032628	-0.059316	(
LotFrontage	-0.009601	-0.357056	1.000000	0.306795	0.234196	-0.052820	(
LotArea	-0.033226	-0.139781	0.306795	1.000000	0.105806	-0.005636	(
OverallQual	-0.028365	0.032628	0.234196	0.105806	1.000000	-0.091932	(
OverallCond	0.012609	-0.059316	-0.052820	-0.005636	-0.091932	1.000000	-(
YearBuilt	-0.012713	0.027850	0.117598	0.014228	0.572323	-0.375983	1
YearRemodAdd	-0.021998	0.040581	0.082746	0.013788	0.550684	0.073741	(
MasVnrArea	-0.050199	0.022895	0.179283	0.103960	0.410238	-0.127788	(
BsmtFinSF1	-0.005024	-0.069836	0.215828	0.214103	0.239666	-0.046231	(
BsmtFinSF2	-0.005968	-0.065649	0.043340	0.111170	-0.059119	0.040229	-(
BsmtUnfSF	-0.007940	-0.140759	0.122156	-0.002618	0.308159	-0.136841	(
TotalBsmtSF	-0.015415	-0.238518	0.363358	0.260833	0.537808	-0.171098	(
1stFlrSF	0.010496	-0.251758	0.414266	0.299475	0.476224	-0.144203	(
2ndFlrSF	0.005590	0.307886	0.072483	0.050986	0.295493	0.028942	(
LowQualFinSF	-0.044230	0.046474	0.036849	0.004779	-0.030429	0.025494	-(
GrLivArea	0.008273	0.074853	0.368392	0.263116	0.593007	-0.079686	(
BsmtFullBath	0.002289	0.003491	0.091481	0.158155	0.111098	-0.054942	(
BsmtHalfBath	-0.020155	-0.002333	-0.006419	0.048046	-0.040150	0.117821	-(
FullBath	0.005587	0.131608	0.180424	0.126031	0.550600	-0.194149	(
HalfBath	0.006784	0.177354	0.048258	0.014259	0.273458	-0.060769	(
BedroomAbvGr	0.037719	-0.023438	0.237023	0.119690	0.101676	0.012980	-(
KitchenAbvGr	0.002951	0.281721	-0.005805	-0.017784	-0.183882	-0.087001	-(
TotRmsAbvGrd	0.027239	0.040380	0.320146	0.190015	0.427452	-0.057583	(
Fireplaces	-0.019772	-0.045569	0.235755	0.271364	0.396765	-0.023820	(
GarageYrBlt	0.000070	0.080187	0.064324	-0.024812	0.518018	-0.306169	(
GarageCars	0.016570	-0.040110	0.269729	0.154871	0.600671	-0.185758	(
GarageArea	0.017634	-0.098672	0.323663	0.180403	0.562022	-0.151521	(
WoodDeckSF	-0.029643	-0.012579	0.077106	0.171698	0.238923	-0.003334	(
OpenPorchSF	-0.000477	-0.006100	0.137454	0.084774	0.308819	-0.032589	(
EnclosedPorch	0.002889	-0.012037	0.009790	-0.018340	-0.113937	0.070356	-(
3SsnPorch	-0.046635	-0.043825	0.062335	0.020423	0.030371	0.025504	(
ScreenPorch	0.001330	-0.026030	0.037684	0.043160	0.064886	0.054811	-(
PoolArea	0.057044	0.008283	0.180868	0.077672	0.065166	-0.001985	(
MiscVal	-0.006242	-0.007683	0.001168	0.038068	-0.031406	0.068777	-(
MoSold	0.021172	-0.013585	0.010158	0.001205	0.070815	-0.003511	(
YrSold	0.000712	-0.021407	0.006768	-0.014261	-0.027347	0.043950	-(

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Y
SalePrice	-0.021917	-0.084284	0.334901	0.263843	0.790982	-0.077856	(

38 rows × 38 columns

In [40]:

```
plt.figure(figsize=(30, 20))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Plot')
plt.show()
```

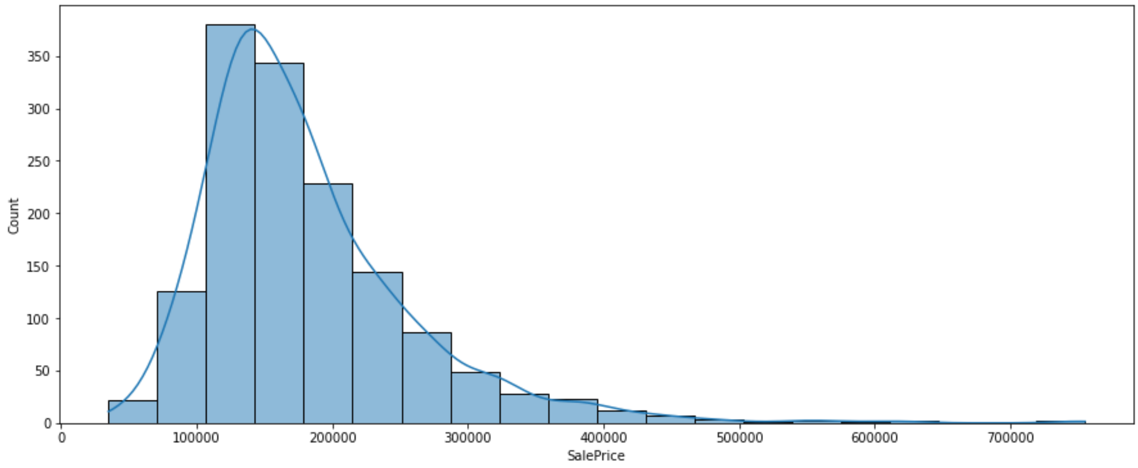


In [41]:

```
df1 = df.copy()
```

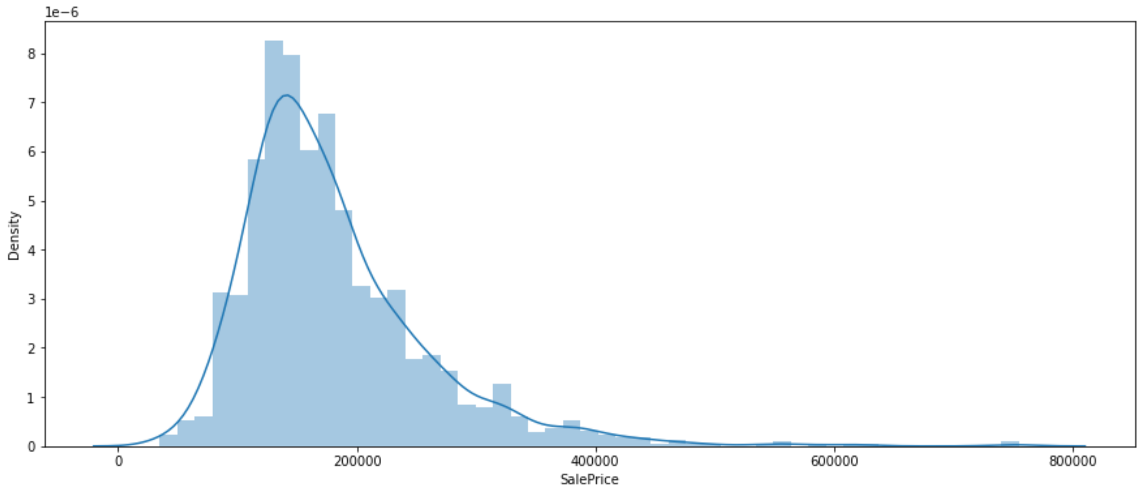
In [42]:

```
plt.figure(figsize=(15,6))
sns.histplot(df1['SalePrice'], kde = True, bins = 20, palette = 'hls')
plt.xticks(rotation = 0)
plt.show()
```



In [43]:

```
plt.figure(figsize=(15,6))
sns.distplot(df1['SalePrice'], kde = True)
plt.xticks(rotation = 0)
plt.show()
```

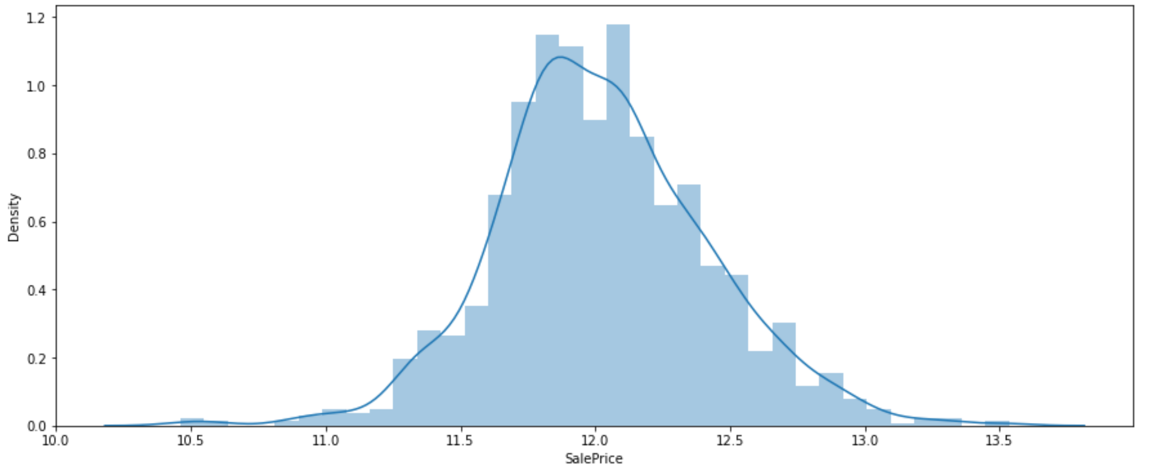


In [44]:

```
df1['SalePrice'] = np.log(df1['SalePrice'])
```

In [45]:

```
plt.figure(figsize=(15,6))
sns.distplot(df1['SalePrice'], kde = True)
plt.xticks(rotation = 0)
plt.show()
```

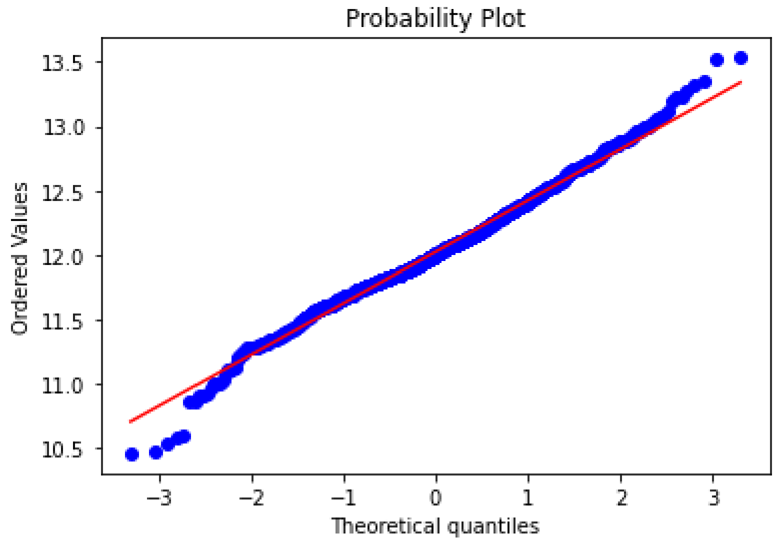


In [46]:

```
from sklearn.preprocessing import StandardScaler
from scipy import stats
```

In [47]:

```
fig = plt.figure()
res = stats.probplot(df1['SalePrice'], plot=plt)
```



In [48]:

```
df1 = df1.drop('Id', axis = 1)
```

In [49]:

```
numerical_columns = ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond
                     'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF
                     'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
                     'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath'
                     'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'Gara
                     'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedP
                     '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSol
```

In [50]:

```
skewness = df1[numerical_columns].skew()
skewed_columns = skewness[(skewness > 1) | (skewness < -1)]
print(skewed_columns)
```

```
MSSubClass        1.407657
LotFrontage       2.384950
LotArea          12.207688
MasVnrArea        2.676412
BsmtFinSF1        1.685503
BsmtFinSF2        4.255261
TotalBsmtSF       1.524255
1stFlrSF          1.376757
LowQualFinSF      9.011341
GrLivArea         1.366560
BsmtHalfBath      4.103403
KitchenAbvGr      4.488397
WoodDeckSF        1.541376
OpenPorchSF       2.364342
EnclosedPorch     3.089872
3SsnPorch        10.304342
ScreenPorch       4.122214
PoolArea         14.828374
MiscVal          24.476794
dtype: float64
```

In [51]:

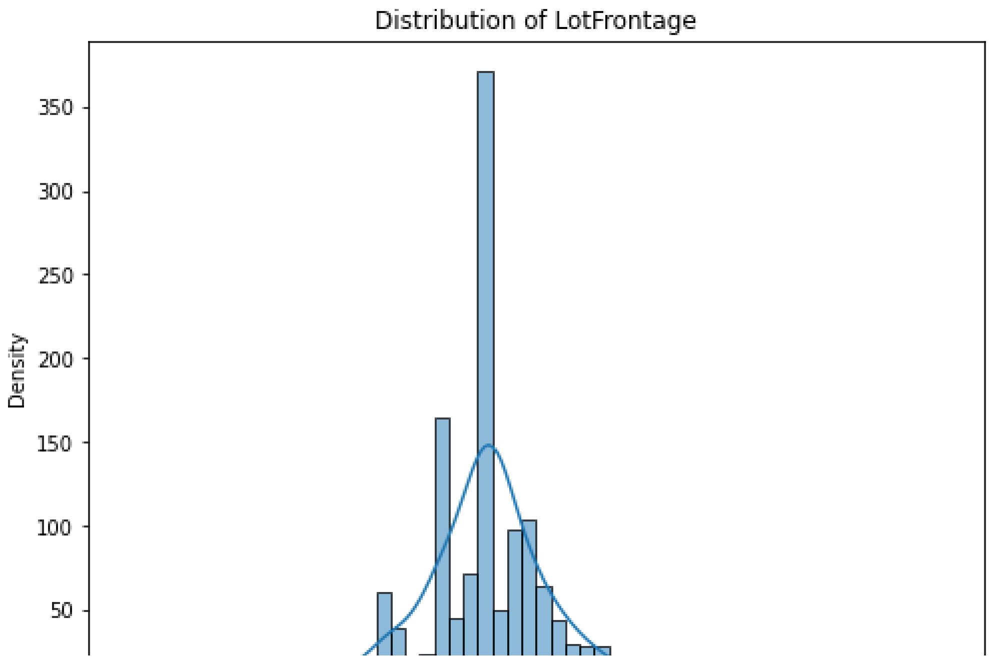
```
skewed_features = ['LotFrontage', 'LotArea', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', '
                   '1stFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtHalfBath', 'KitchenAbvG
                   'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea

for feature in skewed_features:
    df1[feature] = np.log1p(df1[feature])
```

In [52]:

```
transformed_features = ['LotFrontage', 'LotArea', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF
                        '1stFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtHalfBath', 'Kitche
                        'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'Poo
```

In [53]:

```
for feature in transformed_features:
    plt.figure(figsize=(8, 6))
    sns.histplot(df1[feature], kde=True)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Density')
    plt.show()
```



In [54]:

```
df1 = pd.get_dummies(df1, columns=object_columns, drop_first=True)
```

In [55]:

```
df1
```

Out[55]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodA
0	60	4.189655	9.042040	7	5	2003	20
1	20	4.394449	9.169623	6	8	1976	19
2	60	4.234107	9.328212	7	5	2001	20
3	70	4.110874	9.164401	7	5	1915	19
4	60	4.442651	9.565284	8	5	2000	20
...	...	...	...	...	...	...	
1455	60	4.143135	8.976894	6	5	1999	20
1456	20	4.454347	9.486152	6	6	1978	19
1457	70	4.204693	9.109746	7	9	1941	20
1458	20	4.234107	9.181735	5	6	1950	19
1459	20	4.330733	9.204121	5	6	1965	19

1460 rows × 261 columns

In [56]:

```
X = df1.drop(['SalePrice'], axis = 1)
y = df1['SalePrice']
```

In [57]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [58]:

```
from sklearn.linear_model import LinearRegression
regression_model = LinearRegression()
```

In [59]:

```
regression_model.fit(X_train, y_train)
```

Out[59]:

```
▾ LinearRegression
LinearRegression()
```

In [60]:

```
y_pred = regression_model.predict(X_test)
```

In [61]:

```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2_linear = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("R-squared Score:", r2_linear)
```

```
Mean Squared Error (MSE): 0.019397590404698888
Root Mean Squared Error (RMSE): 0.1392752325602039
R-squared Score: 0.8960547414419818
```

In [62]:

```
from sklearn.linear_model import Lasso, Ridge

lasso_model = Lasso()
ridge_model = Ridge()
```

In [63]:

```
lasso_model.fit(X_train, y_train)
```

Out[63]:

```
▾ Lasso
Lasso()
```

In [64]:

```
ridge_model.fit(X_train, y_train)
```

Out[64]:

```
▾ Ridge
Ridge()
```

In [65]:

```
lasso_y_pred = lasso_model.predict(X_test)
ridge_y_pred = ridge_model.predict(X_test)
```

In [66]:

```
from sklearn.metrics import mean_squared_error, r2_score

lasso_mse = mean_squared_error(y_test, lasso_y_pred)
lasso_rmse = np.sqrt(lasso_mse)
lasso_r2 = r2_score(y_test, lasso_y_pred)

ridge_mse = mean_squared_error(y_test, ridge_y_pred)
ridge_rmse = np.sqrt(ridge_mse)
ridge_r2 = r2_score(y_test, ridge_y_pred)

print("Lasso Regression - Mean Squared Error (MSE):", lasso_mse)
print("Lasso Regression - Root Mean Squared Error (RMSE):", lasso_rmse)
print("Lasso Regression - R-squared Score:", lasso_r2)

print('\n')

print("Ridge Regression - Mean Squared Error (MSE):", ridge_mse)
print("Ridge Regression - Root Mean Squared Error (RMSE):", ridge_rmse)
print("Ridge Regression - R-squared Score:", ridge_r2)
```

```
Lasso Regression - Mean Squared Error (MSE): 0.07508906390930213
Lasso Regression - Root Mean Squared Error (RMSE): 0.27402383821357973
Lasso Regression - R-squared Score: 0.5976225912553943


Ridge Regression - Mean Squared Error (MSE): 0.018095090829807328
Ridge Regression - Root Mean Squared Error (RMSE): 0.13451799444612356
Ridge Regression - R-squared Score: 0.9030344050114845
```

In [67]:

```
from sklearn.linear_model import ElasticNet
```

In [68]:

```
model = ElasticNet(alpha=1.0, l1_ratio=0.5)
```

In [69]:

```
model.fit(X_train, y_train)
```

Out[69]:

```
▾ ElasticNet
ElasticNet()
```

In [70]:

```
y_pred = model.predict(X_test)
```

In [71]:

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2_elastic = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("R-squared Score:", r2_elastic)
```

```
Mean Squared Error (MSE): 0.07107240352499863
Root Mean Squared Error (RMSE): 0.2665940800636778
R-squared Score: 0.6191465431213455
```

In [72]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [73]:

```
regressor = DecisionTreeRegressor(random_state=42)
```

In [74]:

```
regressor.fit(X_train, y_train)
```

Out[74]:

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

In [75]:

```
y_pred = regressor.predict(X_test)
```

In [76]:

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2_decision = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Absolute Error (RMSE):", rmse)
print("R-squared Score:", r2_decision)
```

```
Mean Squared Error (MSE): 0.040017724625345835
Root Mean Absolute Error (RMSE): 0.20004430665566525
R-squared Score: 0.7855582757290561
```

In [77]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [78]:

```
regressor = RandomForestRegressor(random_state=42)
```

In [79]:

```
regressor.fit(X_train, y_train)
```

Out[79]:

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [80]:

```
y_pred = regressor.predict(X_test)
```

In [81]:

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2_random = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Absolute Error (RMSE):", rmse)
print("R-squared Score:", r2_random)
```

```
Mean Squared Error (MSE): 0.02146666608916207
Root Mean Absolute Error (RMSE): 0.1465150712014367
R-squared Score: 0.884967250546951
```

In [82]:

```
from xgboost import XGBRegressor
```

In [83]:

```
regressor = XGBRegressor(random_state=42)
```

In [84]:

```
regressor.fit(X_train, y_train)
```

Out[84]:

```
XGBRegressor
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree
=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthw
ise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot
=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_we
```

In [85]:

```
y_pred = regressor.predict(X_test)
```

In [86]:

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2_xgboost = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("R-squared Score:", r2_xgboost)
```

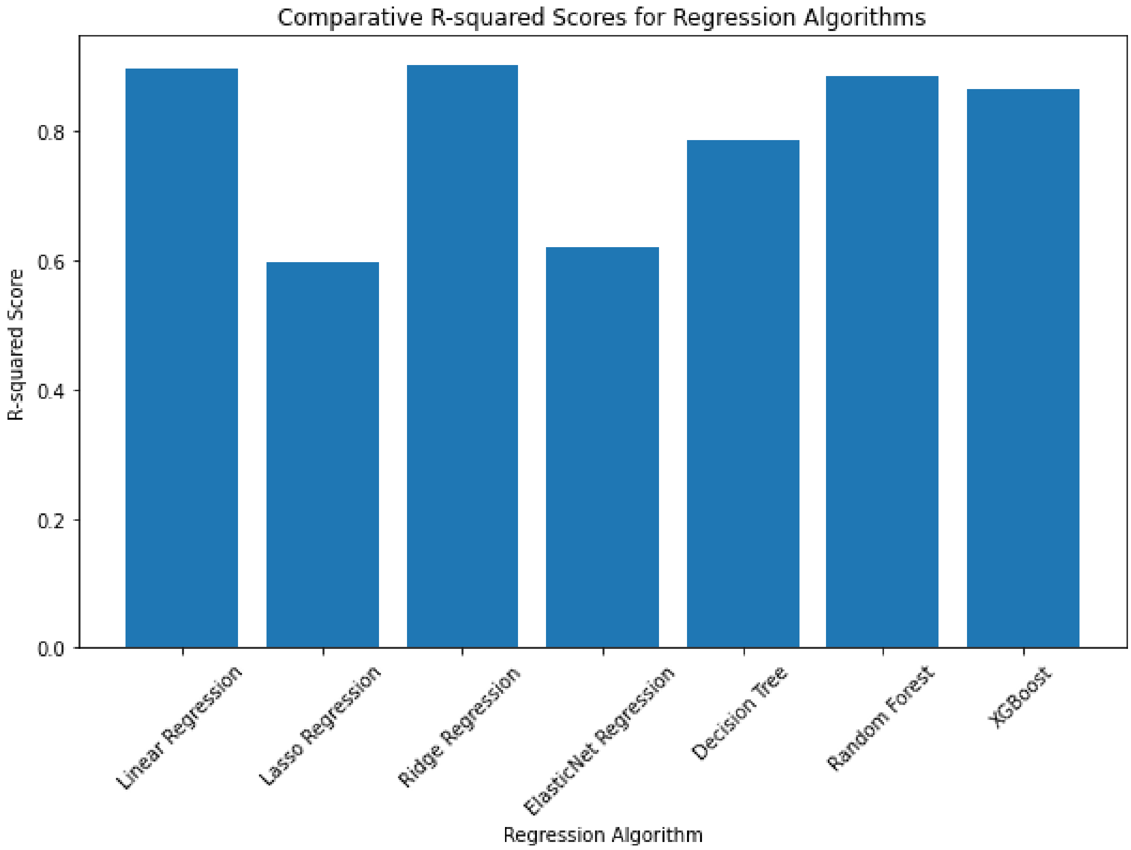
```
Mean Squared Error (MSE): 0.025051259332159053
Root Mean Squared Error (RMSE): 0.15827589624500332
R-squared Score: 0.8657586033028899
```

In [87]:

```
r2_scores = [r2_linear, lasso_r2, ridge_r2, r2_elastic, r2_decision, r2_random, r2_xgboo
algorithms = ['Linear Regression', 'Lasso Regression', 'Ridge Regression', 'ElasticNet R
```

In [88]:

```
plt.figure(figsize=(10, 6))
plt.bar(algorithms, r2_scores)
plt.xlabel('Regression Algorithm')
plt.ylabel('R-squared Score')
plt.title('Comparative R-squared Scores for Regression Algorithms')
plt.xticks(rotation=45)
plt.show()
```



In [89]:

```
fig = go.Figure(data=[go.Bar(x=algorithms, y=r2_scores)])
fig.update_layout(
    title='Comparative R-squared Scores for Regression Algorithms',
    xaxis_title='Regression Algorithm',
    yaxis_title='R-squared Score'
)
fig.show()
```