In [1]:

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

In [2]:

```
mydata=pd.read_csv('C:Downloads/houseprice.csv')
```

In [3]:

```
mydata.head()
```

Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Ut
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 [4]:

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 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     1452 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
 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     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
```

In [5]:

```
mydata.describe()
```

Out[5]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Ye
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.(
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.:
std	421.610009	42.300571	24.284752	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	69.000000	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 [6]:

```
mydata.isnull().sum()
```

Out[6]:

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

Label Encoder

In [8]:

```
from sklearn.preprocessing import LabelEncoder
```

In [9]:

```
LE = LabelEncoder()
```

In [11]:

```
mydata_obj = mydata.select_dtypes(include = 'object').columns
print(mydata_obj)
```

```
Index(['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilitie
s',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',
       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType
2',
       'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
       'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQua
l',
       'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature',
       'SaleType', 'SaleCondition'],
      dtype='object')
```

In [12]:

```
for columns in mydata_obj:
    mydata[columns]=LE.fit_transform(mydata[columns].astype(str))
```

Checking the converted data types

In [14]:

```
1 mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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   int32  
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   int32  
 6   Alley          1460 non-null   int32  
 7   LotShape       1460 non-null   int32  
 8   LandContour    1460 non-null   int32  
 9   Utilities      1460 non-null   int32  
 10  LotConfig      1460 non-null   int32  
 11  LandSlope      1460 non-null   int32  
 12  Neighborhood   1460 non-null   int32  
 13  Condition1     1460 non-null   int32  
 14  Condition2     1460 non-null   int32  
 15  BldgType       1460 non-null   int32  
 16  HouseStyle     1460 non-null   int32  
 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   int32  
 22  RoofMatl       1460 non-null   int32  
 23  Exterior1st    1460 non-null   int32  
 24  Exterior2nd    1460 non-null   int32  
 25  MasVnrType     1460 non-null   int32  
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   int32  
 28  ExterCond      1460 non-null   int32  
 29  Foundation     1460 non-null   int32  
 30  BsmtQual       1460 non-null   int32  
 31  BsmtCond       1460 non-null   int32  
 32  BsmtExposure   1460 non-null   int32  
 33  BsmtFinType1   1460 non-null   int32  
 34  BsmtFinSF1     1460 non-null   int64  
 35  BsmtFinType2   1460 non-null   int32  
 36  BsmtFinSF2     1460 non-null   int64  
 37  BsmtUnfSF      1460 non-null   int64  
 38  TotalBsmtSF    1460 non-null   int64  
 39  Heating        1460 non-null   int32  
 40  HeatingQC      1460 non-null   int32  
 41  CentralAir     1460 non-null   int32  
 42  Electrical     1460 non-null   int32  
 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   int32
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   int32
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    1460 non-null   int32
 58  GarageType     1460 non-null   int32
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1460 non-null   int32
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1460 non-null   int32
 64  GarageCond     1460 non-null   int32
 65  PavedDrive     1460 non-null   int32
 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   int32
 73  Fence          1460 non-null   int32
 74  MiscFeature    1460 non-null   int32
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   int32
 79  SaleCondition  1460 non-null   int32
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int32(43), int64(35)
memory usage: 678.8 KB
```

Filling the null values

In [16]:

```
mydata['LotFrontage']=mydata['LotFrontage'].fillna(0)
mydata['BsmtQual']=mydata['BsmtQual'].fillna(0)
mydata['BsmtCond']=mydata['BsmtCond'].fillna(0)
mydata['BsmtExposure']=mydata['BsmtExposure'].fillna(0)
mydata['BsmtFinType1']=mydata['BsmtFinType1'].fillna(0)
mydata['BsmtFinType2']=mydata['BsmtFinType2'].fillna(0)
mydata['GarageType']=mydata['GarageType'].fillna(0)
mydata['GarageYrBlt']=mydata['GarageYrBlt'].fillna(0)
mydata['GarageFinish']=mydata['GarageFinish'].fillna(0)
mydata['GarageQual']=mydata['GarageQual'].fillna(0)
mydata['Electrical']=mydata['Electrical'].fillna(mydata.Electrical.mean())
mydata['GarageCond']=mydata['GarageCond'].fillna(0)
mydata['MasVnrType']=mydata['MasVnrType'].fillna(0)
mydata['MasVnrArea']=mydata['MasVnrArea'].fillna(0)
mydata['FireplaceQu']=mydata['FireplaceQu'].fillna(0)
mydata['FireplaceQu']=mydata['FireplaceQu'].fillna(0)
```

In [17]:

```
mydata.isnull().sum().head(41)
```

Out[17]:

```
Id              0
MSSubClass      0
MSZoning        0
LotFrontage     0
LotArea         0
Street          0
Alley           0
LotShape        0
LandContour     0
Utilities       0
LotConfig       0
LandSlope       0
Neighborhood    0
Condition1      0
Condition2      0
BldgType        0
HouseStyle      0
OverallQual     0
OverallCond     0
YearBuilt       0
YearRemodAdd    0
RoofStyle       0
RoofMatl        0
Exterior1st     0
Exterior2nd     0
MasVnrType      0
MasVnrArea      0
ExterQual       0
ExterCond       0
Foundation      0
BsmtQual        0
BsmtCond        0
BsmtExposure    0
BsmtFinType1    0
BsmtFinSF1      0
BsmtFinType2    0
BsmtFinSF2      0
BsmtUnfSF       0
TotalBsmtSF     0
Heating         0
HeatingQC       0
dtype: int64
```

In [18]:

```
mydata.isnull().sum().tail(40)
```

Out[18]:

```
CentralAir       0
Electrical       0
1stFlrSF         0
2ndFlrSF         0
LowQualFinSF     0
GrLivArea        0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
HalfBath         0
BedroomAbvGr     0
KitchenAbvGr     0
KitchenQual      0
TotRmsAbvGrd     0
Functional       0
Fireplaces       0
FireplaceQu      0
GarageType       0
GarageYrBlt      0
GarageFinish     0
GarageCars       0
GarageArea       0
GarageQual       0
GarageCond       0
PavedDrive       0
WoodDeckSF       0
OpenPorchSF      0
EnclosedPorch    0
3SsnPorch        0
ScreenPorch      0
PoolArea         0
PoolQC           0
Fence            0
MiscFeature      0
MiscVal          0
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
dtype: int64
```

Correlation

In [20]:

```
mydata_corr = mydata.corr()
```

In [21]:

```
mydata_corr
```

Out[21]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley
Id	1.000000	0.011156	-0.006096	-0.019761	-0.033226	0.008916	-0.001530
MSSubClass	0.011156	1.000000	0.035900	-0.215023	-0.139781	-0.024969	-0.105995
MSZoning	-0.006096	0.035900	1.000000	-0.051250	-0.034452	0.087654	-0.052039
LotFrontage	-0.019761	-0.215023	-0.051250	1.000000	0.100739	-0.025107	0.025087
LotArea	-0.033226	-0.139781	-0.034452	0.100739	1.000000	-0.197131	0.060105
...	...	...	...	...	...	...	...
MoSold	0.021172	-0.013585	-0.031496	0.018942	0.001205	0.003690	0.013094
YrSold	0.000712	-0.021407	-0.020628	-0.012094	-0.014261	-0.025043	0.020944
SaleType	0.019773	0.012464	0.097437	-0.043808	0.012292	0.014339	0.008205
SaleCondition	-0.005806	-0.024940	0.009494	0.054221	0.034169	0.006064	0.035717
SalePrice	-0.021917	-0.084284	-0.166872	0.209624	0.263843	0.041036	0.139868

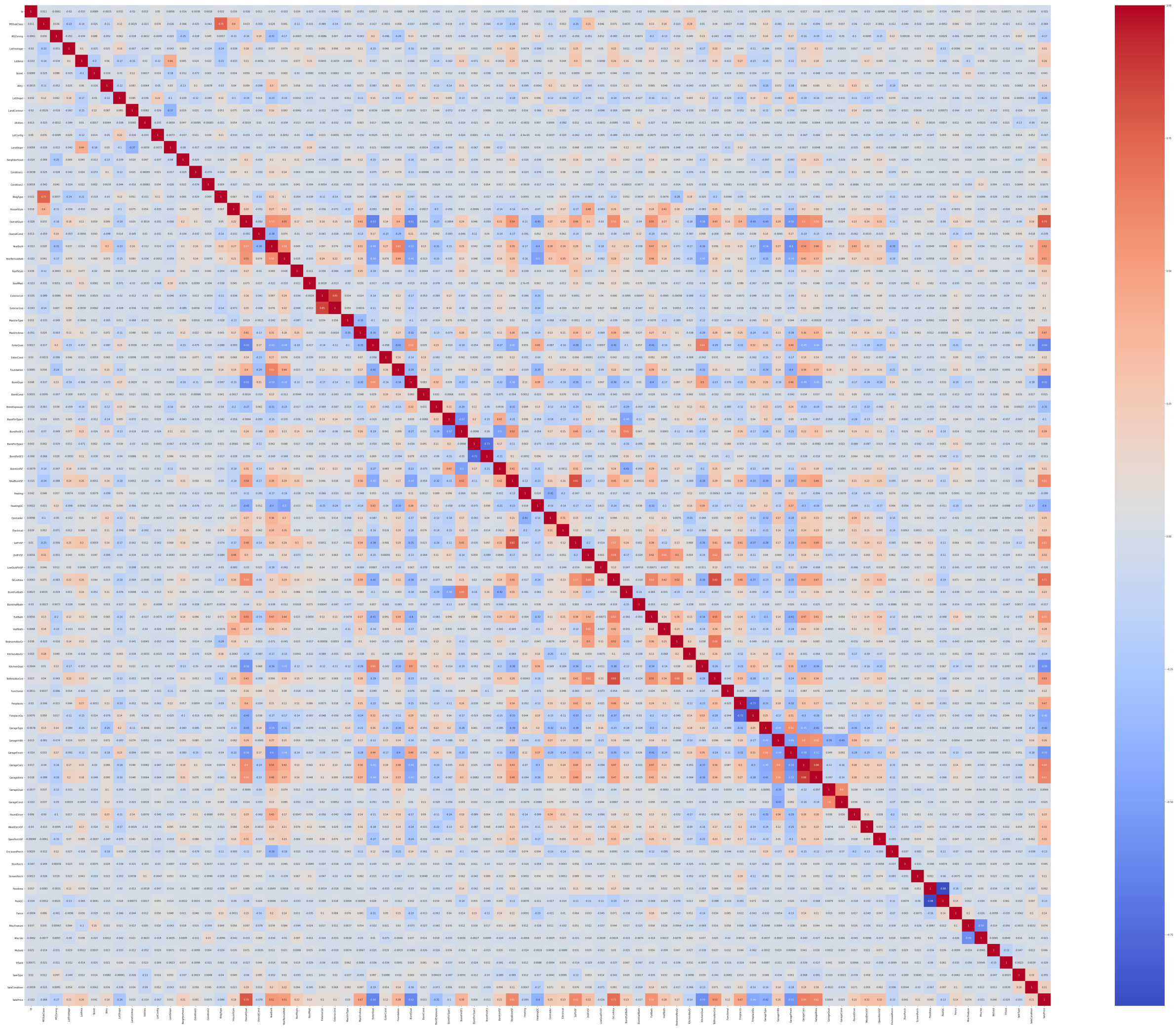
81 rows × 81 columns

In [22]:

```
plt.figure(figsize=(100,80))
sns.heatmap(mydata_corr, annot = True, cmap = 'coolwarm')
```

Out[22]:

<AxesSubplot:>



In [23]:

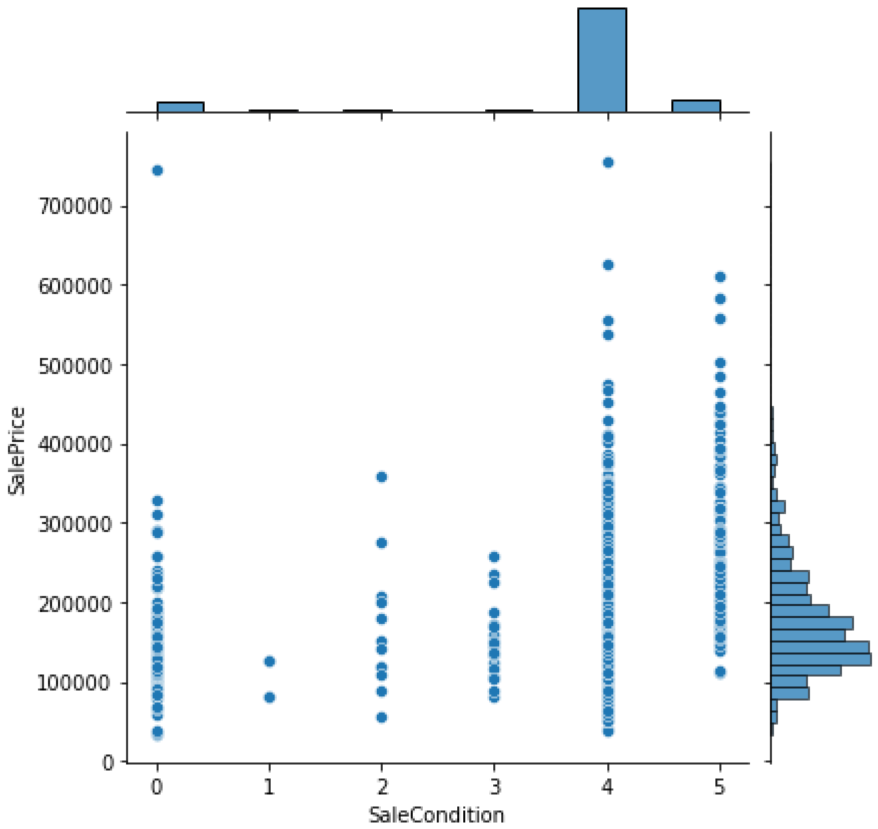
```
#Visualizing the data
```

In [24]:

```
sns.jointplot(x='SaleCondition', y='SalePrice', data=mydata)
```

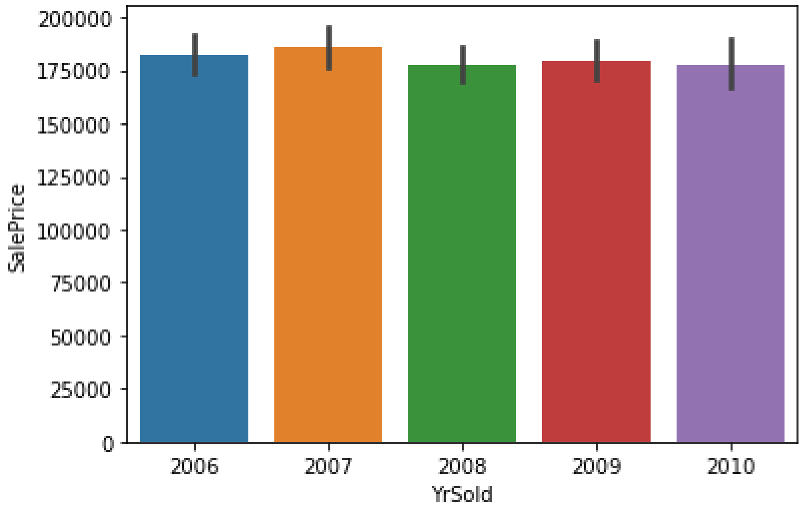
Out[24]:

```
<seaborn.axisgrid.JointGrid at 0x20539ead8e0>
```



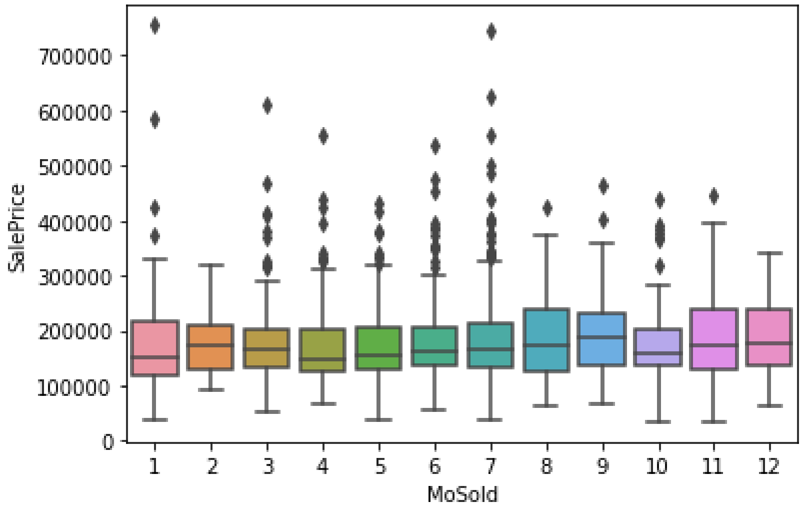
In [25]:

```
sns.barplot(x='YrSold', y='SalePrice', data=mydata);
```



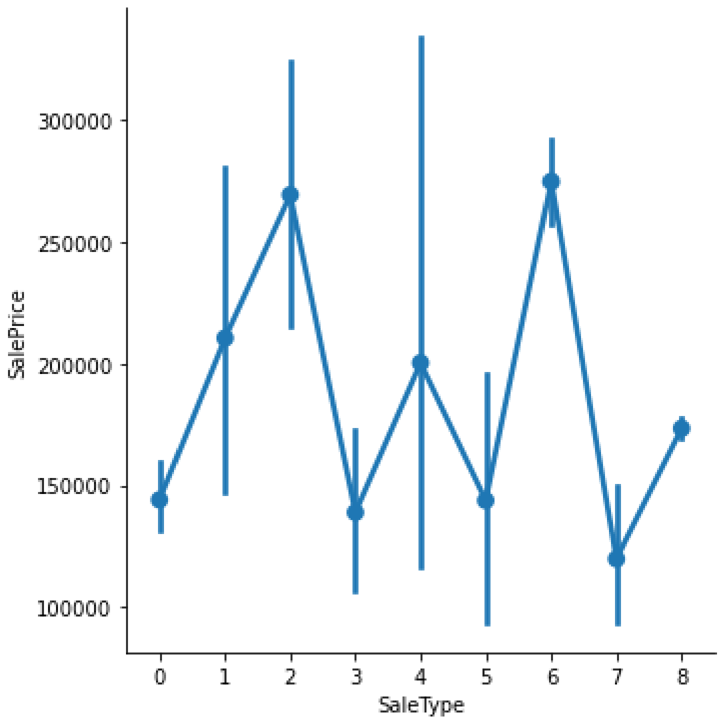
In [26]:

```
sns.boxplot(x='MoSold', y='SalePrice', data=mydata);
```



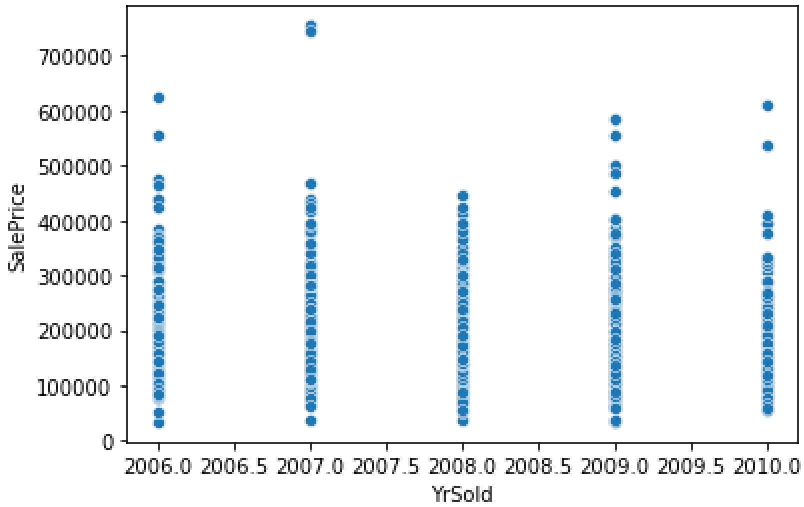
In [27]:

```
sns.catplot(x='SaleType', y='SalePrice',kind='point', data=mydata);
```



In [28]:

```
sns.scatterplot(x='YrSold',y='SalePrice', data=mydata);
```



In [29]:

```
x_ind = mydata.drop('SalePrice',axis=1)
```

In [30]:

```
x_ind
```

Out[30]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
0	1	60	3	65.0	8450	1	2	3	
1	2	20	3	80.0	9600	1	2	3	
2	3	60	3	68.0	11250	1	2	0	
3	4	70	3	60.0	9550	1	2	0	
4	5	60	3	84.0	14260	1	2	0	
...	...	...	...	...	...	...	...	...	
1455	1456	60	3	62.0	7917	1	2	3	
1456	1457	20	3	85.0	13175	1	2	3	
1457	1458	70	3	66.0	9042	1	2	3	
1458	1459	20	3	68.0	9717	1	2	3	
1459	1460	20	3	75.0	9937	1	2	3	

1460 rows × 80 columns

In [31]:

```
y_dep = mydata.SalePrice
```

Before Normalization

In [33]:

```
x_ind = mydata.drop('SalePrice', axis = 1)
```

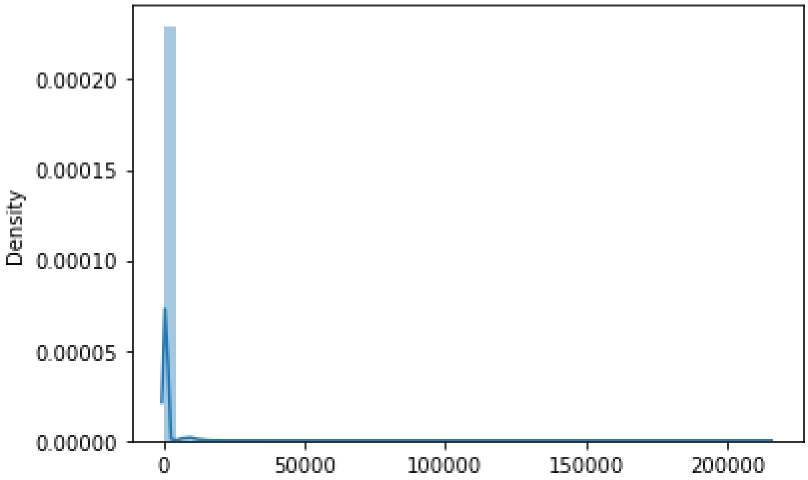
In [34]:

```
sns.distplot(x_ind)
```

```
C:\Users\aneef\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[34]:

```
<AxesSubplot:ylabel='Density'>
```



Standardizing the data

In [36]:

```
from sklearn.preprocessing import StandardScaler
```

In [37]:

```
norm = StandardScaler()
```

In [38]:

```
x_norm = norm.fit_transform(x_ind)
```

After Normalization

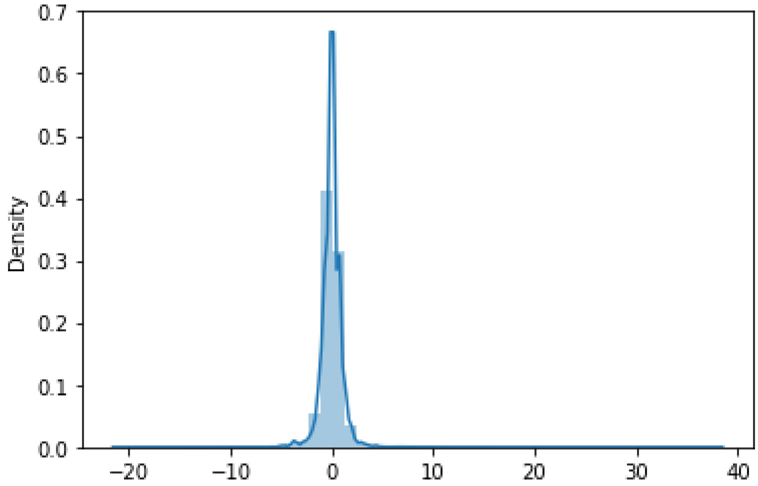
In [40]:

```
sns.distplot(x_norm)
```

```
C:\Users\aneef\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[40]:

```
<AxesSubplot:ylabel='Density'>
```



Perform using PCA

In [42]:

```
from sklearn.decomposition import PCA
```

In [43]:

```
PCA_Red = PCA()
```

In [44]:

```
x_new = PCA_Red.fit_transform(x_norm)
```

In [45]:

```
x_new
```

Out[45]:

```
array([[ 1.89763883e+00,  4.85627859e-01, -1.83851418e+00, ...,
        -7.56457486e-02, -1.19803169e-15,  5.31008807e-17],
       [ 3.61410333e-01, -2.07014506e+00,  1.03480822e+00, ...,
         1.19603234e-02,  1.09470203e-14, -1.00614673e-15],
       [ 2.46368699e+00,  2.36421566e-01, -1.42909507e+00, ...,
        -2.86038522e-01, -8.16277200e-16,  4.97418885e-16],
       ...,
       [ 1.65623646e+00,  2.35781195e+00,  1.39842345e+00, ...,
        -3.02532487e-01, -2.88403616e-16, -2.76430223e-16],
       [-2.64296338e+00, -2.99174153e+00,  1.50298924e+00, ...,
        -2.91106478e-02, -1.42970653e-15, -8.58861519e-17],
       [-6.39579847e-01, -2.85748153e+00,  1.30219476e+00, ...,
         6.69109547e-02, -5.63349486e-16,  1.94277387e-16]])
```

Covariance Matrix

In [47]:

```
cov_mat = np.cov(x_norm.T)
```

In [48]:

```
cov_mat
```

Out[48]:

```
array([[ 1.00068540e+00,  1.11641249e-02, -6.10029226e-03, ...,
         7.12281881e-04,  1.97869611e-02, -5.80978086e-03],
       [ 1.11641249e-02,  1.00068540e+00,  3.59246662e-02, ...,
        -2.14217103e-02,  1.24727778e-02, -2.49566714e-02],
       [-6.10029226e-03,  3.59246662e-02,  1.00068540e+00, ...,
        -2.06417083e-02,  9.75040488e-02,  9.50002094e-03],
       ...,
       [ 7.12281881e-04, -2.14217103e-02, -2.06417083e-02, ...,
         1.00068540e+00, -2.32888954e-03,  3.88308594e-03],
       [ 1.97869611e-02,  1.24727778e-02,  9.75040488e-02, ...,
        -2.32888954e-03,  1.00068540e+00,  1.84192716e-01],
       [-5.80978086e-03, -2.49566714e-02,  9.50002094e-03, ...,
         3.88308594e-03,  1.84192716e-01,  1.00068540e+00]])
```

In [49]:

```
eigen_vals,eigen_vecs=np.linalg.eig(cov_mat)
```

In [50]:

```
1 eigen_vals
```

Out[50]:

```
array([ 1.03757329e+01,  4.09658578e+00,  3.53972317e+00,  2.99064240e+00,
        2.51045091e+00,  2.21509055e+00,  2.06147584e+00,  1.96343414e+00,
        1.74781555e+00,  1.74105361e+00,  1.68844926e+00,  1.60883397e+00,
        1.53053749e+00,  1.43857915e+00,  1.36917018e+00,  1.31619820e+00,
        1.26039559e+00,  1.22084408e+00,  1.16857879e+00,  1.16127888e+00,
        1.12265006e+00,  1.11748859e+00,  1.09547489e+00,  1.06229245e+00,
        1.04290051e+00,  1.02147613e+00,  9.95231266e-01,  9.63671413e-01,
        9.42010269e-01,  9.36497863e-01,  9.13436997e-01,  8.89980560e-01,
        8.70609143e-01,  8.52353859e-02,  9.22938892e-02,  1.02178053e-01,
        1.08869919e-01,  1.20943792e-01,  1.34865637e-01,  8.48673357e-01,
        8.37233936e-01,  7.95763304e-01,  7.94145660e-01,  1.64828088e-01,
        7.74649006e-01,  2.04170463e-01,  2.10365541e-01,  2.16480428e-01,
        2.26677344e-01,  7.48746262e-01,  7.28191438e-01,  7.21971249e-01,
        7.05003515e-01,  6.78549212e-01,  6.61743313e-01,  6.51897435e-01,
        6.22724155e-01,  6.11218017e-01,  5.98339143e-01,  2.52321808e-01,
        2.58570000e-01,  5.72502582e-01,  5.46911757e-01,  5.34237438e-01,
        5.22921738e-01,  2.98114139e-01,  4.73492459e-01,  3.15280804e-01,
        3.25241599e-01,  3.34617291e-01,  3.57324607e-01,  3.63345644e-01,
        3.74536214e-01,  3.89215067e-01,  4.09902227e-01,  4.43835692e-01,
        4.25705953e-01,  4.32406966e-01,  4.56990613e-16, -1.19025200e-16])
```

In [51]:

```
1 eigen_vecs
```

Out[51]:

```
array([[ 1.60397423e-03, -8.21556757e-03, -1.52141219e-03, ...,
        -2.59941281e-03, -1.36802163e-17,  5.17668257e-17],
       [ 1.35033094e-02, -9.95792167e-02, -1.70350440e-01, ...,
        -4.10969164e-02, -4.45513068e-16,  1.86463804e-16],
       [ 7.64075276e-02, -6.32197546e-03,  9.15562253e-02, ...,
        -2.32997158e-02,  1.18915833e-17,  7.46703019e-17],
       ...,
       [ 8.46853346e-03,  3.53687555e-02, -1.37606833e-02, ...,
        -5.83108403e-02, -1.03224292e-16,  2.70554910e-17],
       [ 2.01082635e-02,  1.10200748e-03, -1.07807074e-03, ...,
         6.27508140e-02, -1.27331082e-16, -6.38745929e-17],
       [-7.60624410e-02,  9.00457286e-04, -8.00080128e-02, ...,
        -7.79934286e-02,  9.79184071e-17,  7.17849763e-17]])
```

Explained Variance

In [53]:

```
1 explained_variance = PCA_Red.explained_variance_ratio_
```

In [54]:

```
explained_variance
```

Out[54]:

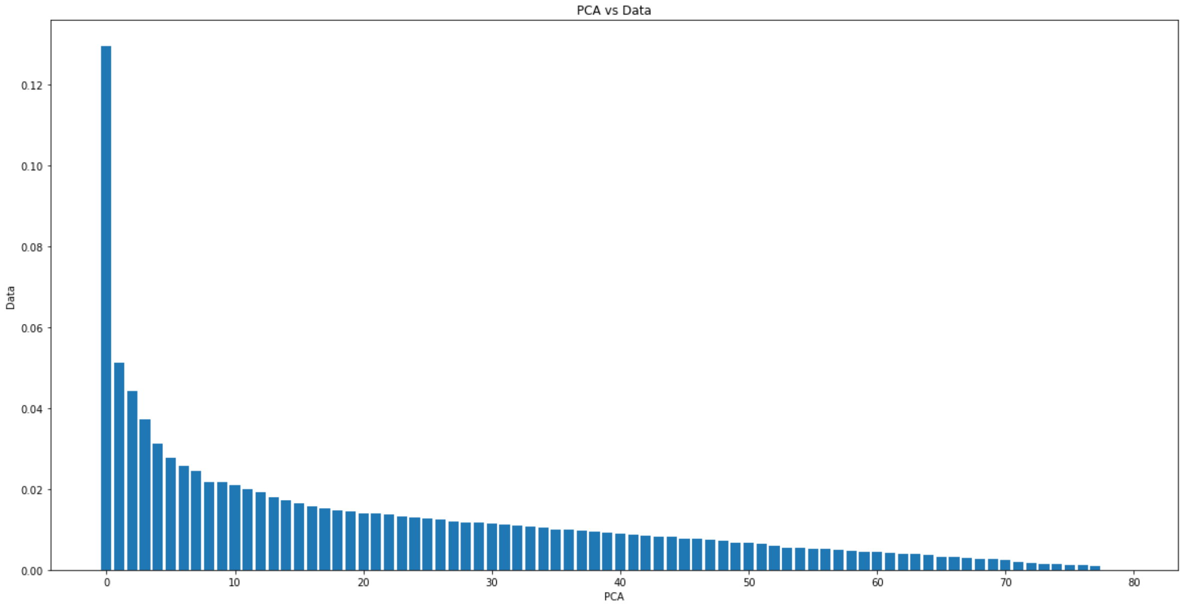
```
array([1.29607827e-01, 5.11722488e-02, 4.42162337e-02, 3.73574252e-02,
       3.13591428e-02, 2.76696671e-02, 2.57507983e-02, 2.45261165e-02,
       2.18327302e-02, 2.17482639e-02, 2.10911598e-02, 2.00966503e-02,
       1.91186147e-02, 1.79699228e-02, 1.71029050e-02, 1.64412087e-02,
       1.57441538e-02, 1.52500985e-02, 1.45972299e-02, 1.45060436e-02,
       1.40235140e-02, 1.39590399e-02, 1.36840571e-02, 1.32695606e-02,
       1.30273274e-02, 1.27597061e-02, 1.24318700e-02, 1.20376421e-02,
       1.17670632e-02, 1.16982053e-02, 1.14101419e-02, 1.11171373e-02,
       1.08751604e-02, 1.06011509e-02, 1.04582561e-02, 9.94022826e-03,
       9.92002156e-03, 9.67648031e-03, 9.35291778e-03, 9.09615846e-03,
       9.01845935e-03, 8.80650794e-03, 8.47605566e-03, 8.26612580e-03,
       8.14313662e-03, 7.77872040e-03, 7.63499218e-03, 7.47411651e-03,
       7.15138071e-03, 6.83171450e-03, 6.67339402e-03, 6.53204465e-03,
       5.91460186e-03, 5.54414618e-03, 5.40138496e-03, 5.31767966e-03,
       5.12026840e-03, 4.86185601e-03, 4.67849603e-03, 4.53870972e-03,
       4.46349830e-03, 4.17985126e-03, 4.06273538e-03, 3.93831073e-03,
       3.72387438e-03, 3.22991122e-03, 3.15186232e-03, 2.83152607e-03,
       2.70415193e-03, 2.62776819e-03, 2.55038275e-03, 2.05893991e-03,
       1.68466579e-03, 1.51076192e-03, 1.35994188e-03, 1.27635085e-03,
       1.15288343e-03, 1.06471257e-03, 2.55606370e-32, 1.13956272e-33])
```

In [55]:

```
plt.figure(figsize=(20,10))
plt.bar(range(80),explained_variance,label='Info gained by each PCA')
plt.xlabel('PCA')
plt.ylabel('Data')
plt.title('PCA vs Data')
```

Out[55]:

```
Text(0.5, 1.0, 'PCA vs Data')
```




```
We can take PCA n_components = 78 because by adding the 78 values from the above
array output we get 95.010256% approx by checking with the explained variance ratio.
For considering n_components, the PCA percentage should be around 95%
```

In [58]:

```
pca = PCA(n_components = 78)
```

In [59]:

```
x_new_info=pca.fit_transform(x_norm)
```

In [60]:

```
x_new_info
```

Out[60]:

```
array([[ 1.89763883,  0.48562786, -1.83851418, ..., -0.08217073,
        -0.0911645 , -0.07564575],
       [ 0.36141033, -2.07014506,  1.03480822, ...,  0.01345269,
         0.05654119,  0.01196032],
       [ 2.46368699,  0.23642157, -1.42909507, ...,  0.28245711,
        -0.05897354, -0.28603852],
       ...,
       [ 1.65623646,  2.35781195,  1.39842345, ..., -0.515532  ,
         0.20210621, -0.30253249],
       [-2.64296338, -2.99174153,  1.50298924, ..., -0.2169645 ,
         0.08330945, -0.02911065],
       [-0.63957985, -2.85748153,  1.30219476, ..., -0.28326542,
        -0.06648037,  0.06691095]])
```

Performing Train Test split using x_new_info

In [61]:

```
from sklearn.model_selection import train_test_split
```

In [62]:

```
x_train,x_test,y_train,y_test=train_test_split(x_new_info,y_dep,train_size=0.8,random_
```

After compressing the dataset, we can now further perform machine learning by training our data using Random Forest

Random Forest

In [64]:

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

In [65]:

```
model_RF = RandomForestRegressor()
```

In [66]:

```
model_RF.fit(x_train,y_train)
```

Out[66]:

```
RandomForestRegressor()
```

In [67]:

```
y_pred = model_RF.predict(x_test)
```

In [68]:

```
1 y_pred
```

Out[68]:

```
array([210228.93, 164991.96, 115111.16,  91583.5 , 154239.59, 326404.1 ,
       280639.17, 152666.12, 199659.66, 260622.69, 174379.35,  62318.22,
       225188.99, 317035.25, 243540.91, 110974.91, 118027.24, 133944.74,
       203614.3 , 140341.66, 114734.73, 133180.01, 230250.86, 346067.04,
       102930.85, 194953.04, 161196.1 , 171553.5 , 454035.09, 132063.85,
       116431.33, 118300.08, 119741.65,  90548.22, 165665.79, 355504.83,
       135791.64,  92309.5 , 267066.96, 110470.62, 142519.73, 145577.33,
        97688.41, 126681.51, 174214.32, 171283.5 , 116666.58, 187586.62,
       251158.39, 208244.26,  94800.22, 239533.19, 129077.02, 231422.39,
       192267.87, 115501.34, 117675.  , 173954.25, 118787.92, 189997.22,
       154056.01, 244629.65, 112319.38, 134689.21, 153447.41, 130548.78,
       120356.  , 201975.99, 158139.52, 156310.87, 163624.53,  91332.65,
       363191.83, 149471.23, 161255.  , 219057.25, 176467.07, 143897.75,
       440355.99, 232092.76, 212752.7 , 118005.59, 136680.31, 157197.1 ,
       166875.  , 148794.14, 152599.11, 170178.5 , 200905.55, 170675.35,
       243496.52, 214628.95,  99422.5 , 101823.63, 124943.74, 149051.5 ,
       117010.71, 119332.48, 143053.17, 151617.5 , 179755.57, 140459.  ,
       118794.24, 122487.75, 137782.5 , 152456.24, 183168.16, 158750.53,
       119537.97, 318808.73, 146578.48, 178682.57, 145108.1 , 197242.63,
       243623.05, 182181.04, 239143.05, 133662.2 , 182818.  , 260753.54,
       134001.53, 246340.15, 327194.68, 158329.25, 210537.53, 157741.73,
       385087.55, 116410.23, 206993.62, 225976.42, 268750.82,  89217.1 ,
       129291.64, 117477.87,  91944.5 , 202367.79, 459419.63, 382823.61,
       232921.35, 139187.75,  87385.72, 281944.87, 209202.49, 200958.15,
        73832.88, 198651.84,  92210.2 , 196840.02, 207596.92, 122418.08,
       167993.5 , 158949.02, 119729.87, 182709.33, 191013.75, 368577.22,
        69954.71, 135345.53,  77032.04, 139188.7 ,  68286.72, 111169.12,
       159788.5 , 143073.44, 134702.6 , 126404.25, 164153.5 , 120544.08,
       151485.4 , 101158.95, 242670.45, 144145.96, 214496.84, 273770.22,
       187799.28, 137980.79, 164775.59, 203557.87, 137397.14, 164722.02,
       134629.5 , 211758.03, 142270.13, 145310.59, 315298.44, 127260.11,
       374830.7 , 300611.89, 160596.25, 132219.39, 115004.7 , 142555.5 ,
        99081.  , 199803.04, 136867.01, 251772.72, 196961.26, 154906.1 ,
       142111.75,  98129.5 , 197663.25, 252187.47, 160272.  , 169765.  ,
       250437.27,  99849.76, 199347.9 , 307177.83, 198342.72, 327467.63,
       193437.4 , 111568.  , 191303.77, 127745.22, 308037.65, 252922.74,
       117413.58,  88342.72, 219286.62,  98522.  , 412771.5 , 124407.79,
       164010.02, 193002.34, 115306.46, 123227.25, 203593.99, 160684.9 ,
       187558.99, 173079.85, 125741.08, 191623.45,  96498.72, 120742.08,
       349381.4 , 130918.21, 296824.8 , 126287.16, 139664.31, 291401.64,
       355118.96, 155322.88, 141660.96, 146752.6 , 143271.75, 113241.97,
       142885.13, 139390.32, 168612.6 , 167014.02, 142929.13, 132323.76,
       189082.84, 161362.87,  90534.61, 125950.77, 170053.29,  83835.5 ,
       356338.5 , 134438.25, 223418.02, 159922.4 , 210624.9 , 189600.44,
       136526.92, 164831.  , 118667.84, 200539.1 , 146920.62, 102099.37,
       166319.02, 159866.  ,  80215.72, 109463.92, 192059.25,  79308.72,
       123233.73, 120249.92, 158695.28, 142609.1 , 116308.5 , 136046.32,
       117078.65, 148827.27, 233877.  , 239391.89, 126243.59,  91278.  ,
       250753.42, 118400.24,  85877.65, 306389.1 ])
```

In [69]:

```
model_RF.score(x_test,y_test)
```

Out[69]:

```
0.8488148068531223
```

In [70]:

```
#Mean Square error
```

In [71]:

```
from sklearn.metrics import mean_squared_error
```

In [72]:

```
MSE = mean_squared_error(y_test,y_pred)
```

In [73]:

```
MSE
```

Out[73]:

```
1078244236.892512
```

In [74]:

```
#Root mean square
```

In [75]:

```
root_mean_sqr = np.sqrt(MSE)
```

In [76]:

```
root_mean_sqr
```

Out[76]:

```
32836.62949957733
```

In [77]:

```
f_comp = pd.DataFrame({'Actual':y_test,'Machine_Pred':y_pred})
```

In [79]:

```
1 f_comp
```

Out[79]:

	Actual	Machine_Pred
258	231500	210228.93
267	179500	164991.96
288	122000	115111.16
649	84500	91583.50
1233	142000	154239.59
...	...	...
163	103200	91278.00
47	249700	250753.42
1432	64500	118400.24
98	83000	85877.65
409	339750	306389.10

292 rows × 2 columns

Residual

In [81]:

```
1 Residual = y_test-y_pred
```

In [82]:

```
1 Residual    #Difference of error between actual and machine predicted values
```

Out[82]:

```
258      21271.07
267      14508.04
288       6888.84
649      -7083.50
1233    -12239.59
           ...
163      11922.00
47       -1053.42
1432    -53900.24
98       -2877.65
409      33360.90
Name: SalePrice, Length: 292, dtype: float64
```

In [84]:

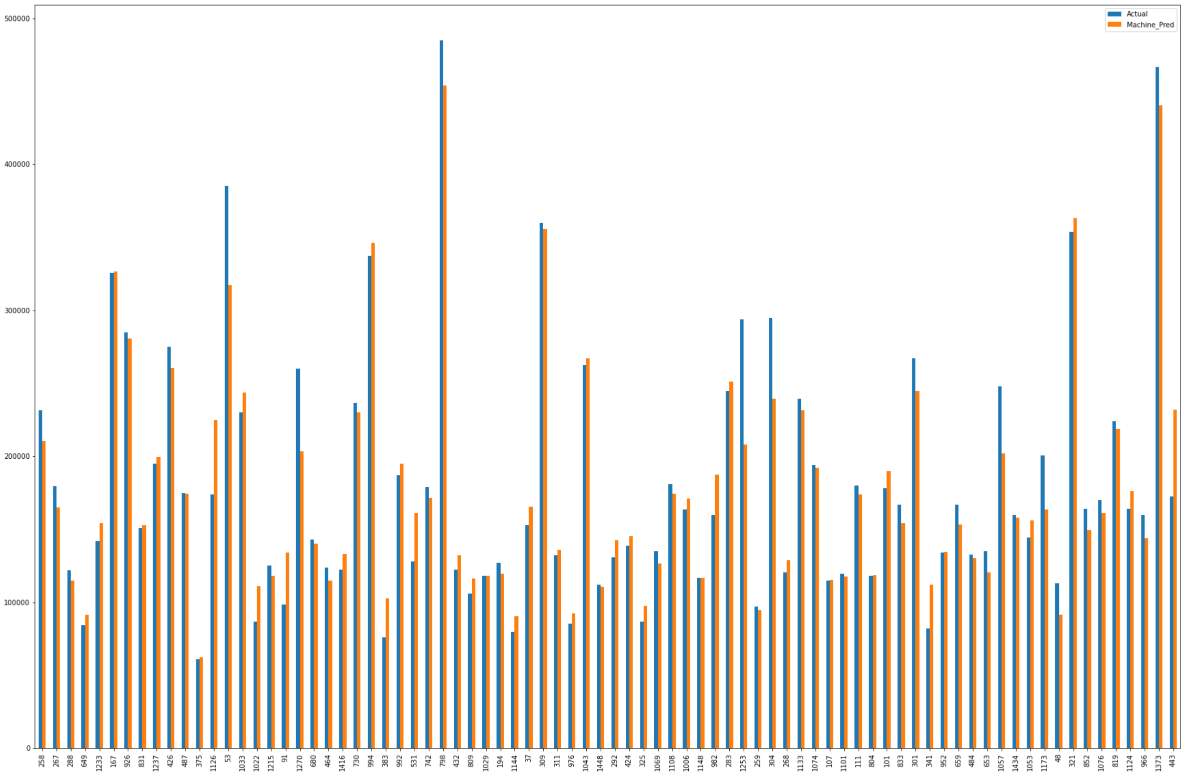
```
comp_graph = f_comp.head(80)
```

In [86]:

```
comp_graph.plot(kind='bar',figsize=(30,20));
```

Out[86]:

```
<AxesSubplot:>
```



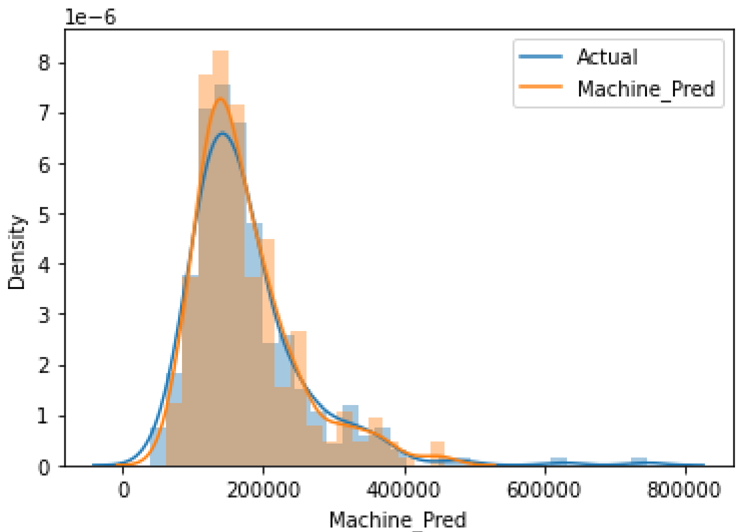
In [87]:

```
sns.distplot(f_comp['Actual'])
sns.distplot(f_comp['Machine_Pred'])
plt.legend(['Actual','Machine_Pred'])
```

```
C:\Users\aneef\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\aneef\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[87]:

```
<matplotlib.legend.Legend at 0x2053098b550>
```



Split without using PCA

In [89]:

```
from sklearn.model_selection import train_test_split
```

In [90]:

```
X_train,X_test,Y_train,Y_test=train_test_split(x_norm,y_dep,train_size=0.2,random_state
```

In [91]:

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

In [93]:

```
model_rf = RandomForestRegressor()
```

In [94]:

```
model_rf.fit(X_train,Y_train)
```

Out[94]:

```
RandomForestRegressor()
```

In [95]:

```
Y_pred_rf = model_rf.predict(X_test)
```

In [96]:

```
Y_pred_rf
```

Out[96]:

```
array([207735.1 , 169960.09, 117147.94, ..., 138603.25, 190443.07,
       305025.8 ])
```

In [97]:

```
model_rf.score(X_test,Y_test)
```

Out[97]:

```
0.8175697853303188
```

In [98]:

```
#Mean Squared error
```

In [99]:

```
from sklearn.metrics import mean_squared_error
```

In [100]:

```
mse = mean_squared_error(Y_test,Y_pred_rf)
mse
```

Out[100]:

```
1181803761.1816185
```

In [101]:

```
#Root mean square
```

In [102]:

```
Root_mean_SQR = np.sqrt(mse)
```

In [103]:

```
Root_mean_SQR
```

Out[103]:

34377.37280802037

In [105]:

```
F_comp = pd.DataFrame({'Actual':Y_test,'Machine_Pred':Y_pred_rf})
```

In [106]:

```
F_comp
```

Out[106]:

	Actual	Machine_Pred
258	231500	207735.10
267	179500	169960.09
288	122000	117147.94
649	84500	92477.00
1233	142000	143800.25
...	...	...
1317	208900	187983.38
1107	274725	216897.12
230	148000	138603.25
652	191000	190443.07
70	244000	305025.80

1168 rows × 2 columns

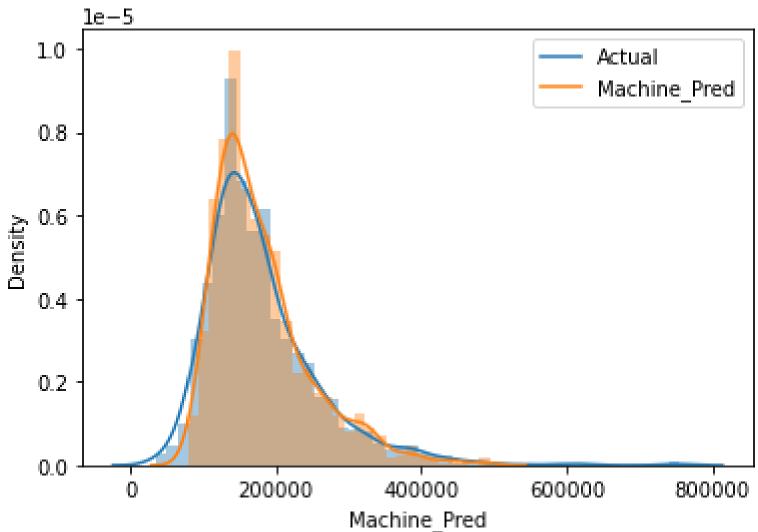
In [107]:

```
sns.distplot(F_comp['Actual'])
sns.distplot(F_comp['Machine_Pred'])
plt.legend(['Actual','Machine_Pred'])
```

```
C:\Users\aneef\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\aneef\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[107]:

```
<matplotlib.legend.Legend at 0x205311c3670>
```



Conclusion

```
In this project, we have compared the score and also used a comparison graph to show the difference between using PCA and without using PCA.
The score we got after using PCA is 84%
And the score before using PCA is 81%
Thus we can clearly see the PCA boosts the performance of the model by removing the outliers and performing high dimension reduction.
```