Encoding:

```
In [44]: #Label encoding
```

```
In [77]: import pandas as pd
data = {'Animal': ['Cat', 'Dog', 'Dog', 'Fish', 'Cat'], 'Flower': ['rose','sunflower','lotus','lilly','jasmine'],'Price':[1,2
df = pd.DataFrame(data)
print(df)
```

```
  Animal     Flower  Price
0    Cat       rose      1
1    Dog  sunflower      2
2    Dog      lotus      2
3   Fish      lilly      3
4    Cat    jasmine      4
```

```
In [78]: from sklearn.preprocessing import LabelEncoder
#create instance of label encoder
encoder = LabelEncoder()
#perform label encoding on 'Animal' column
df['enc_Animal'] = encoder.fit_transform(df['Animal'])
print(df)
```

```
  Animal     Flower  Price  enc_Animal
0    Cat       rose      1           0
1    Dog  sunflower      2           1
2    Dog      lotus      2           1
3   Fish      lilly      3           2
4    Cat    jasmine      4           0
```

```
In [4]: df['Flower'] = encoder.fit_transform(df['Flower'])
print(df)
```

```
  Animal  Flower  Price  enc_Animal
0    Cat       3      1           0
1    Dog       4      2           1
2    Dog       2      2           1
3   Fish       1      3           2
4    Cat       0      4           0
```

```
In [ ]: #One hot encoding
```

```
In [80]: one_hot_encoded_data = pd.get_dummies(df)
print(one_hot_encoded_data)
```

```
   Flower  Price  enc_Animal  Animal_Cat  Animal_Dog  Animal_Fish
0       3      1           0        True       False        False
1       4      2           1       False        True        False
2       2      2           1       False        True        False
3       1      3           2       False       False         True
4       0      4           0        True       False        False
```

```
In [81]: one_hot_encoded_data = pd.get_dummies(df, dtype= 'int')
print(one_hot_encoded_data)
```

```
   Flower  Price  enc_Animal  Animal_Cat  Animal_Dog  Animal_Fish
0       3      1           0           1           0            0
1       4      2           1           0           1            0
2       2      2           1           0           1            0
3       1      3           2           0           0            1
4       0      4           0           1           0            0
```

Convert all true or false into 0 and 1

```
In [82]: one_hot_encoded_data = pd.get_dummies(df, dtype= 'int', columns=['Animal','Flower'])
         print(one_hot_encoded_data)
```

```
   Price  enc_Animal  Animal_Cat  Animal_Dog  Animal_Fish  Flower_0  Flower_1  \
0      1           0           1           0            0         0         0
1      2           1           0           1            0         0         0
2      2           1           0           1            0         0         0
3      3           2           0           0            1         0         1
4      4           0           1           0            0         1         0

   Flower_2  Flower_3  Flower_4
0         0         1         0
1         0         0         1
2         1         0         0
3         0         0         0
```

Normalization:

Min max

Z normalization

```
import pandas as pd

# create data
df = pd.DataFrame([
                   [180000, 110, 18.9, 1400],
                   [360000, 905, 23.4, 1800],
                   [230000, 230, 14.0, 1300],
                   [60000, 450, 13.5, 1500]],

                  columns=['Col A', 'Col B',
                           'Col C', 'Col D'])

# view data
display(df)
```

	Col A	Col B	Col C	Col D
0	180000	110	18.9	1400
1	360000	905	23.4	1800
2	230000	230	14.0	1300
3	60000	450	13.5	1500

```
In [85]: import matplotlib.pyplot as plt
         df.plot(kind = 'bar')
```

```
In [90]: # copy the data
         df_min_max_scaled = df.copy()

         # apply normalization techniques
         for column in df_min_max_scaled.columns:
             df_min_max_scaled[column] = (df_min_max_scaled[column] - df_min_max_scaled[column].min()) / (df_min_max_scaled[column].ma

         # view normalized data
         print(df_min_max_scaled)
```

```
      Col A     Col B     Col C  Col D
0  0.400000  0.000000  0.545455    0.2
1  1.000000  1.000000  1.000000    1.0
2  0.566667  0.150943
3  0.000000  0.427673
```

In [90]:
```
# copy the data
df_min_max_scaled = df.copy()

# apply normalization techniques
for column in df_min_max_scaled.columns:
    df_min_max_scaled[column] = (df_min_max_scaled[column] - df_min_max_scaled[column].min()) / (df_min_max_scaled[column].ma

# view normalized data
print(df_min_max_scaled)
```

Pop out this

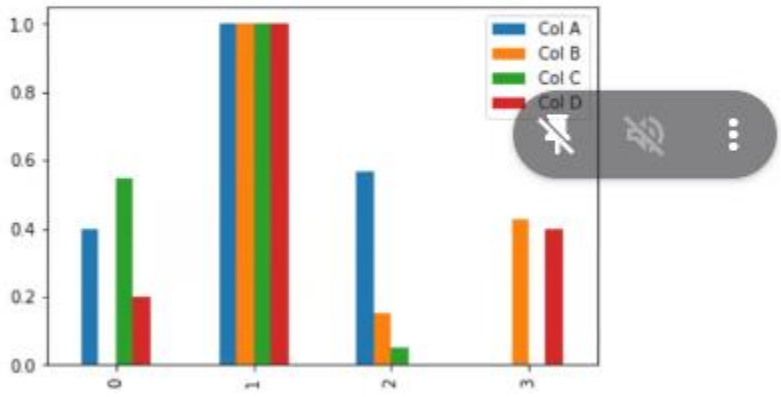
```
      Col A     Col B     Col C  Col D
0  0.400000  0.000000  0.545455    0.2
1  1.000000  1.000000  1.000000    1.0
2  0.566667  0.150943  0.050505    0.0
3  0.000000  0.427673  0.000000    0.4
```

```
_scaled[column] - df_min_max_scaled[column].min()) / (df_min_max_scaled[column].max() - df_min_max_scaled[column].min())
```

```
      Col A     Col B     Col C  Col D
0  0.400000  0.000000  0.545455    0.2
1  1.000000  1.000000  1.000000    1.0
2  0.566667  0.150943  0.050505    0.0
3  0.000000  0.427673  0.000000    0.4
```

In [13]:
```
import matplotlib.pyplot as plt
df_min_max_scaled.plot(kind = 'bar')
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x2ac8d73d148>



In [14]:
```
# copy the data
df_min_max_scaled = df.copy()

# apply normalization techniques
for column in df_min_max_scaled.columns:
    df_min_max_scaled.loc[2:4,column] = (df_min_max_scaled.loc[2:5,column] - df_min_max_scaled[column].min()) / (df_min_max_s

# view normalized data
print(df_min_max_scaled)
```

```
          Col A       Col B      Col C   Col D
0  180000.000000  110.000000  18.900000  1400.0
1  360000.000000  925.000000  23.400000  1800.0
2       0.566667    0.150943   0.050505     0.0
3       0.000000    0.427673   0.000000     0.4
```

In [93]:
```
#Z-Score normalization
```

In [94]:
```
# copy the data
df_z_scaled = df.copy()

# apply normalization techniques
for column in df_z_scaled.columns:
    df_z_scaled[column] = (df_z_scaled[column] -
                           df_z_scaled[column].mean()) / df_z_scaled[column].std()

# view normalized data
display(df_z_scaled)
```

	Col A	Col B	Col C	Col D
0	-0.221422	-0.895492	0.311486	-0.46291
1	1.227884	1.373564	1.278167	1.38873
2	0.181163	-0.552993	-0.741122	-0.92582
3	-1.187625	0.074922	-0.848531	0.00000

In [15]:
```
import matplotlib.pyplot as plt
df_z_scaled.plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ac8d825b88>
```