In [135]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings(action="ignore")
```

In [2]:

```
df = pd.read_csv("datasets_435_896_sales_data_sample.csv", encoding='Latin-1')
```

In [3]:

```
df.head()
```

Out[3]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERD
0	10107	30	95.70	2	2871.00	2/24/2
1	10121	34	81.35	5	2765.90	5/7/2003
2	10134	41	94.74	2	3884.34	7/1/2003
3	10145	45	83.26	6	3746.70	8/25/2
4	10159	49	100.00	14	5205.27	10/10/2

5 rows × 25 columns

In [4]:

```
print("No. of Rows in DataFrame : ", df.shape[0])
print("No. of Columns in DataFrame : ", df.shape[1])
```

```
No. of Rows in DataFrame :  2823
No. of Columns in DataFrame :  25
```

In [7]:

```
# Columns in DataFrame

list(data.columns)
```

Out[7]:

```
['ORDERNUMBER',
 'QUANTITYORDERED',
 'PRICEEACH',
 'ORDERLINENUMBER',
 'SALES',
 'ORDERDATE',
 'STATUS',
 'QTR_ID',
 'MONTH_ID',
 'YEAR_ID',
 'PRODUCTLINE',
 'MSRP',
 'PRODUCTCODE',
 'CUSTOMERNAME',
 'PHONE',
 'ADDRESSLINE1',
 'ADDRESSLINE2',
 'CITY',
 'STATE',
 'POSTALCODE',
 'COUNTRY',
 'TERRITORY',
 'CONTACTLASTNAME',
 'CONTACTFIRSTNAME',
 'DEALSIZE']
```

In [6]:

```
# Checking for Duplicate Rows in the training set

duplicate_rows = df[df.duplicated()]
print("Duplicate Rows :", duplicate_rows)
```

```
Duplicate Rows : Empty DataFrame
Columns: [ORDERNUMBER, QUANTITYORDERED, PRICEEACH, ORDERLINENUMBER, SA
LES, ORDERDATE, STATUS, QTR_ID, MONTH_ID, YEAR_ID, PRODUCTLINE, MSRP,
PRODUCTCODE, CUSTOMERNAME, PHONE, ADDRESSLINE1, ADDRESSLINE2, CITY, ST
ATE, POSTALCODE, COUNTRY, TERRITORY, CONTACTLASTNAME, CONTACTFIRSTNAM
E, DEALSIZE]
Index: []

[0 rows x 25 columns]
```

Observation: No Duplicate Row in Dataframe.

In [7]:

```
# Checking for duplicate columns

def getDuplicateColumns(df):

    '''
    Utility Function to get a list of duplicate columns.
    '''

    duplicateColumnNames = set()
    # Iterate over all the columns in dataframe
    for x in range(df.shape[1]):
        # Select column at xth index.
        col = df.iloc[:, x]
        # Iterate over all the columns in DataFrame from (x+1)th index till end
        for y in range(x + 1, df.shape[1]):
            # Select column at yth index.
            otherCol = df.iloc[:, y]
            # Check if two columns at x 7 y index are equal
            if col.equals(otherCol):
                duplicateColumnNames.add(df.columns.values[y])
    return list(duplicateColumnNames)

duplicate_columns = getDuplicateColumns(df)

print(duplicate_columns)
```

```
[]
```

Observation: No Duplicate Columns in Dataframe.

In [8]:

```
# Checking for Missing Values in the Dataframe

def missing_info(column, df):

    na = df[column].isna()
    count = na.sum()
    total_count = df.shape[0]
    miss_prcnt = np.round((count/total_count)*100,3)

    return (count, miss_prcnt)
```

In [9]:

```
def missing_train_info(df):

    columns_missing_info = []

    for column in df:

        count, miss_prcnt = missing_info(column, df);

        if(count):
            columns_missing_info.append([column, count, miss_prcnt])

    column_names = ['Feature_Name', 'Missing_Count', 'Missing_Percentage']

    missing_info_df = pd.DataFrame(data = columns_missing_info, columns = column_na

    return missing_info_df
```

In [10]:

```
missing_train_df = missing_train_info(df)
```

In [11]:

```
# Modifying the display setting of the pandas so as to view all the rows in a dataf

pd.set_option("display.max_rows", None, "display.max_columns", None)
# pd.reset_option("display.max_rows", "display.max_columns")
```

In [12]:

```
missing_train_df.head(df.shape[1])
```

Out[12]:

	Feature_Name	Missing_Count	Missing_Percentage
0	ADDRESSLINE2	2521	89.302
1	STATE	1486	52.639
2	POSTALCODE	76	2.692
3	TERRITORY	1074	38.045

Observation:

- ADDRESSLINE2 Feature has the most number of missing values, followed by STATE, TERRITORY and POSTALCODE.

Performing EDA on each Column

Univariate Analysis

DEALSIZE

In [75]:

```
df[['DEALSIZE']].describe()
```

Out[75]:

	DEALSIZE
count	2823
unique	3
top	Medium
freq	1384

In [76]:

```
df[["DEALSIZE"]].value_counts()
```

Out[76]:

```
DEALSIZE
Medium      1384
Small       1282
Large        157
dtype: int64
```

In [78]:

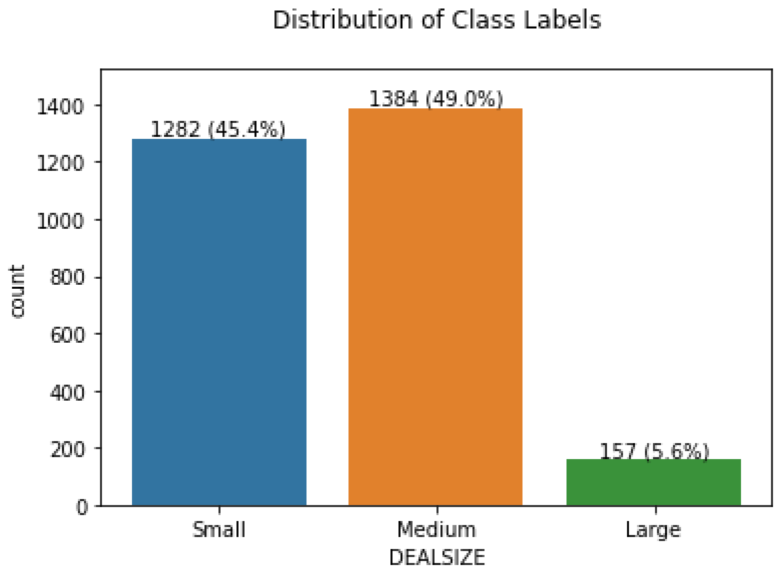
```
ax = sns.countplot(x='DEALSIZE', data = df)

plt.title("\nDistribution of Class Labels\n")

plt.margins(0.05, 0.1)

for p in ax.patches:
  x=p.get_bbox().get_points()[:,0]
  y=p.get_bbox().get_points()[1,1]
  ax.annotate('{} ({:.1f}%)'.format(int(y),100.*y/len(df)), (x.mean(), y),
          ha='center', va='bottom')

plt.show()
```



Observation:

- This feature seems to be the predictor or the class label having 3 classes namely SMALL, MEDIUM, LARGE.
- This prblem can be considered as a MultiClass Classification Problem where given a data point we need to classify it having DEALSIZE as SMALL, MEDIUM or LARGE.

ORDERNUMBER

In [44]:

```
df[['ORDERNUMBER']].describe()
```

Out[44]:

	ORDERNUMBER
count	2823.000000
mean	10258.725115
std	92.085478
min	10100.000000
25%	10180.000000
50%	10262.000000
75%	10333.500000
max	10425.000000

In [67]:

```
df[['ORDERNUMBER']].nunique()
```

Out[67]:

```
ORDERNUMBER    307
dtype: int64
```

Observation:

- We have a total of 2823 rows but we only have 307 unique order numbers which means we have repeated Order Numbers in Dataframe.

QUANTITYORDERED

In [48]:

```
df[['QUANTITYORDERED']].describe()
```

Out[48]:

	QUANTITYORDERED
count	2823.000000
mean	35.092809
std	9.741443
min	6.000000
25%	27.000000
50%	35.000000
75%	43.000000
max	97.000000

In [66]:

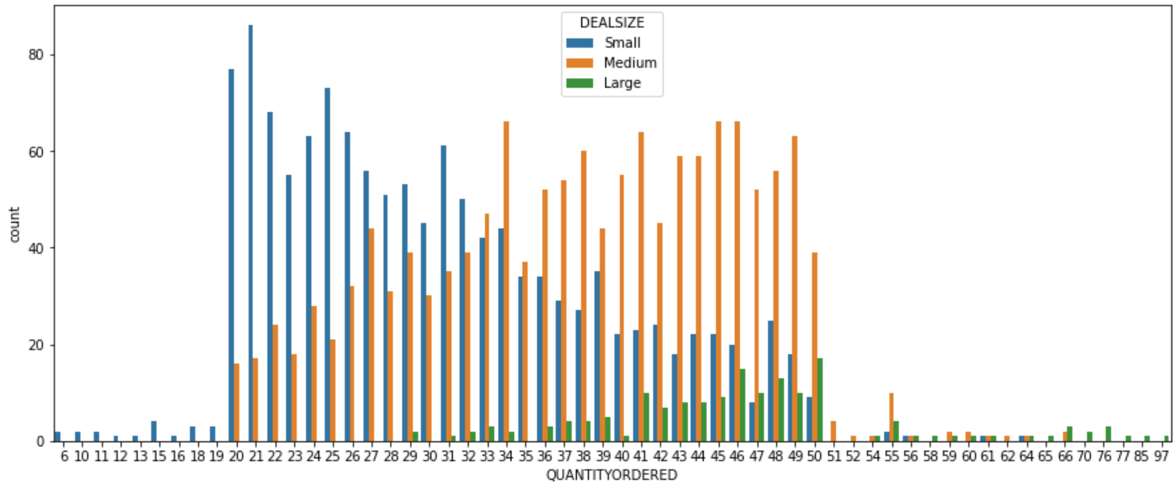
```
df[['QUANTITYORDERED']].nunique()
```

Out[66]:

```
QUANTITYORDERED    58
dtype: int64
```

In [59]:

```
plt.figure(figsize=(15,6))
sns.countplot(x="QUANTITYORDERED",hue='DEALSIZE', data=df)
plt.show()
```



Observation:

- Count per QUANTITYORDERED majorly lies between 20 and 50.

PRICEEACH

In [51]:

```
df[['PRICEEACH']].describe()
```

Out[51]:

	PRICEEACH
count	2823.000000
mean	83.658544
std	20.174277
min	26.880000
25%	68.860000
50%	95.700000
75%	100.000000
max	100.000000

In [65]:

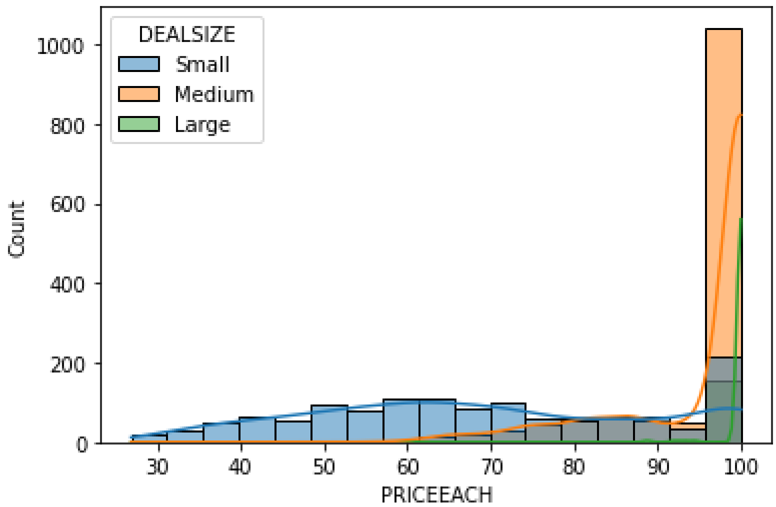
```
df[['PRICEEACH']].nunique()
```

Out[65]:

```
PRICEEACH    1016
dtype: int64
```

In [53]:

```
sns.histplot(x='PRICEEACH',hue='DEALSIZE', data=df,  kde=True)
plt.show()
```



Observation:

- Small Sized Deals have been distributed almost evenly on the PRICE.
- Medium Sized Deals majorly have high Price
- Large Deals also have somewhat high price but not as much as Medium Sized Deals.

ORDERLINENUMBER

In [62]:

```
df[['ORDERLINENUMBER']].describe()
```

Out[62]:

	ORDERLINENUMBER
count	2823.000000
mean	6.466171
std	4.225841
min	1.000000
25%	3.000000
50%	6.000000
75%	9.000000
max	18.000000

In [64]:

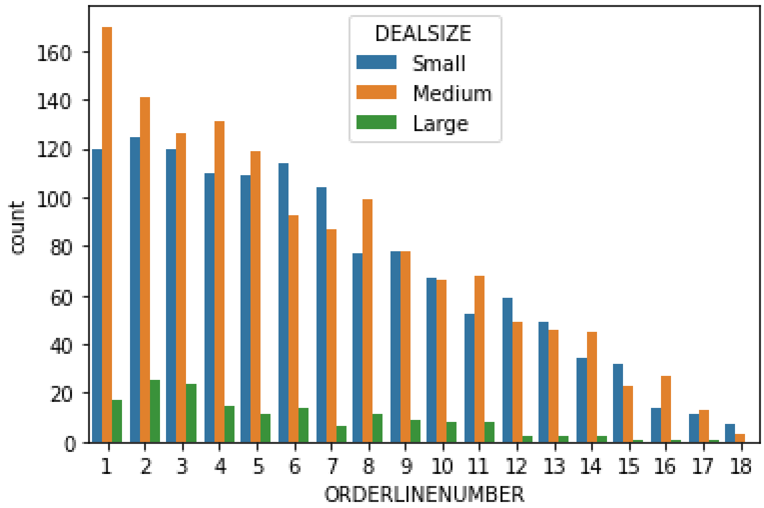
```
df[['ORDERLINENUMBER']].nunique()
```

Out[64]:

```
ORDERLINENUMBER    18
dtype: int64
```

In [68]:

```
sns.countplot(x="ORDERLINENUMBER",hue='DEALSIZE', data=df)
plt.show()
```



Observation:

- Majority of the Orders are proceeded using the initial Order Line Numbers.

SALES

In [70]:

```
df[['SALES']].describe()
```

Out[70]:

	SALES
count	2823.000000
mean	3553.889072
std	1841.865106
min	482.130000
25%	2203.430000
50%	3184.800000
75%	4508.000000
max	14082.800000

In [72]:

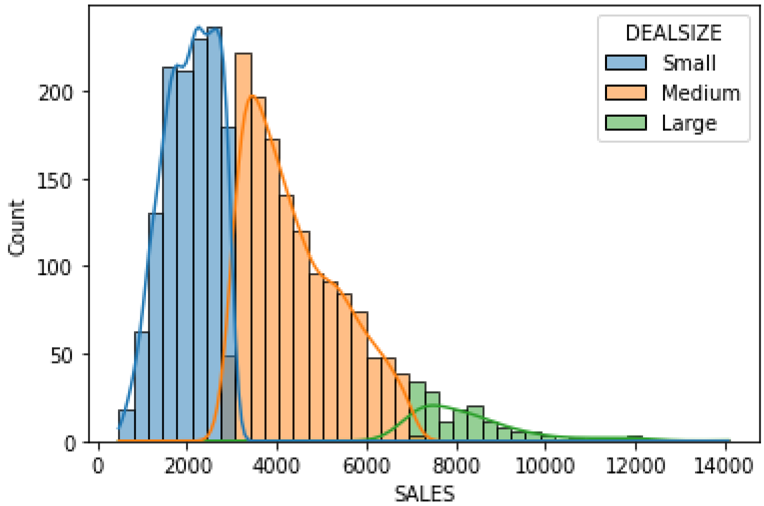
```
df[['SALES']].nunique()
```

Out[72]:

```
SALES    2763
dtype: int64
```

In [74]:

```
sns.histplot(x='SALES',hue='DEALSIZE', data=df,  kde=True)
plt.show()
```



Observation:

- SALES feature seem to be very good in predicting the DEALSIZE since from the above graph we can see a particular range of SALES in which each DEALSIZE lies.
- We can simply construct a simple If-Else based system to classify the DEALSIZE based on the SALES,

  A data point having SALES between,

```
<3000 - SMALL
3000-7000 - MEDIUM
>7000 - LARGE
```

This system will make some errors too but the errors will be very low in number.

ORDERDATE

In [81]:

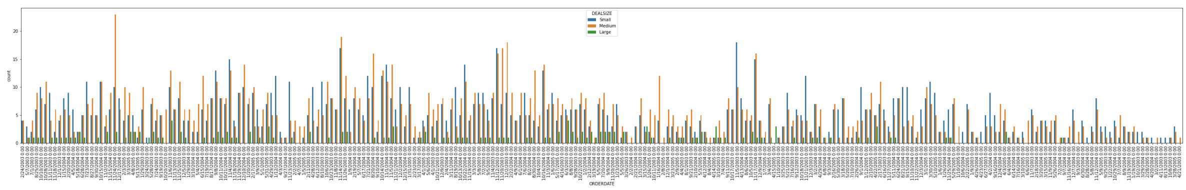
```
df[['ORDERDATE']].describe()
```

Out[81]:

	ORDERDATE
count	2823
unique	252
top	11/14/2003 0:00
freq	38

In [88]:

```
plt.figure(figsize=(50,6))
sns.countplot(x="ORDERDATE",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```



STATUS

In [79]:

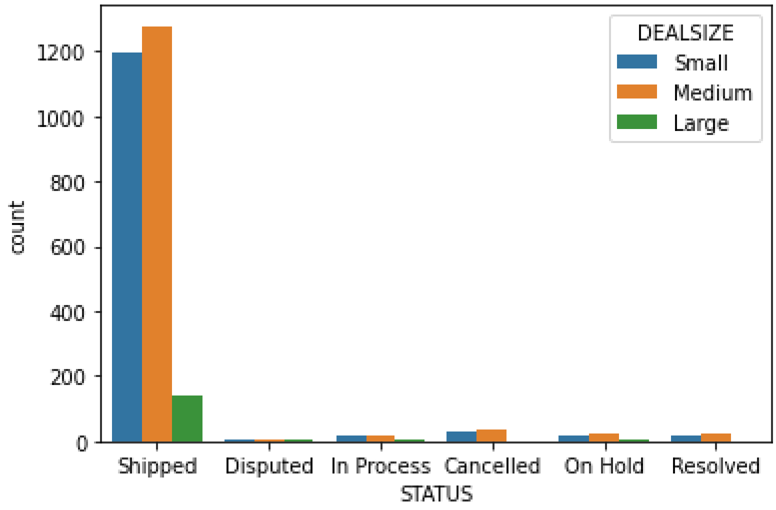
```
df[['STATUS']].describe()
```

Out[79]:

	STATUS
count	2823
unique	6
top	Shipped
freq	2617

In [80]:

```
sns.countplot(x="STATUS",hue='DEALSIZE', data=df)
plt.show()
```



Observation:

- Majority of the Orders have Shipped Status.

QTR_ID

In [96]:

```
df[['QTR_ID']].describe()
```

Out[96]:

	QTR_ID
count	2823.000000
mean	2.717676
std	1.203878
min	1.000000
25%	2.000000
50%	3.000000
75%	4.000000
max	4.000000

In [98]:

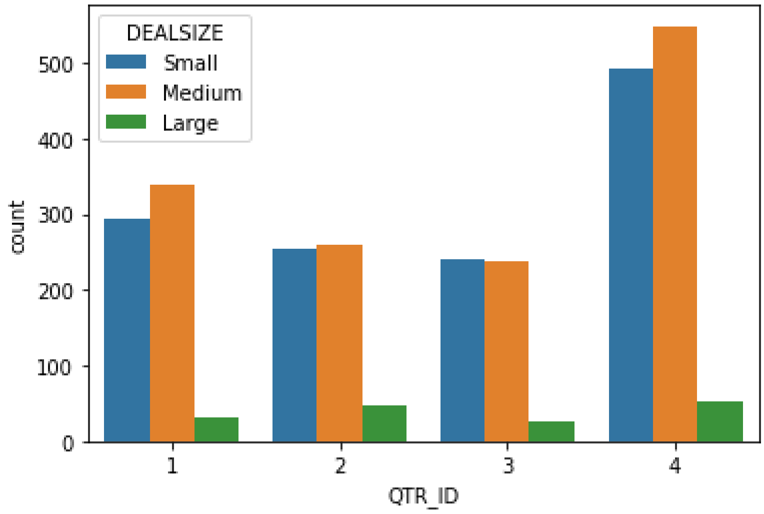
```
df[['QTR_ID']].nunique()
```

Out[98]:

```
QTR_ID    4
dtype: int64
```

In [99]:

```
sns.countplot(x="QTR_ID",hue='DEALSIZE', data=df)
plt.show()
```



Observation:

- Sales in the last Quarter is comparatively higher than the sales in other quarter.

MONTH_ID

In [100]:

```
df[['MONTH_ID']].describe()
```

Out[100]:

	MONTH_ID
count	2823.000000
mean	7.092455
std	3.656633
min	1.000000
25%	4.000000
50%	8.000000
75%	11.000000
max	12.000000

In [101]:

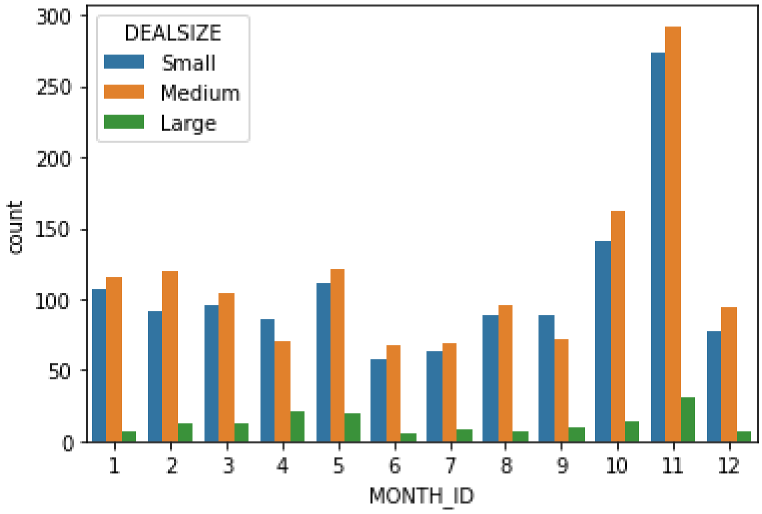
```
df[['MONTH_ID']].nunique()
```

Out[101]:

```
MONTH_ID    12
dtype: int64
```

In [103]:

```
sns.countplot(x="MONTH_ID",hue='DEALSIZE', data=df)
plt.show()
```



Observation:

- Much higher sale happened in the Month with MONT_ID 11 (possibly November).

YEAR_ID

In [104]:

```
df[['YEAR_ID']].describe()
```

Out[104]:

	YEAR_ID
count	2823.00000
mean	2003.81509
std	0.69967
min	2003.00000
25%	2003.00000
50%	2004.00000
75%	2004.00000
max	2005.00000

In [106]:

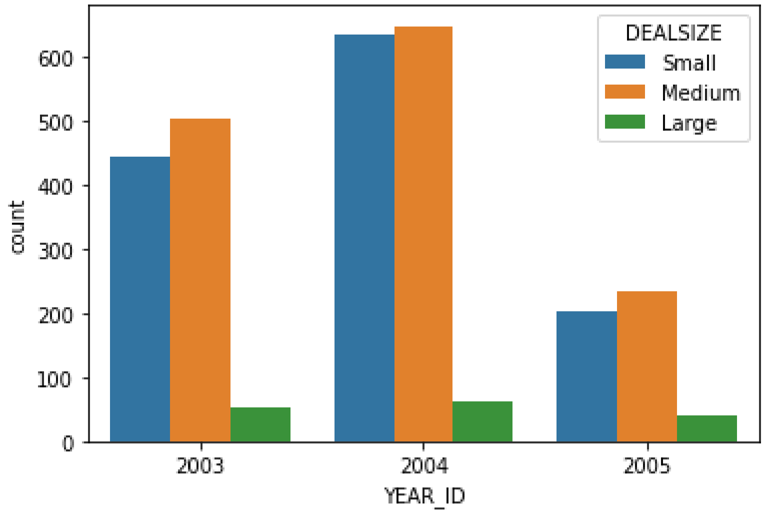
```
df[['YEAR_ID']].nunique()
```

Out[106]:

```
YEAR_ID    3
dtype: int64
```

In [107]:

```
sns.countplot(x="YEAR_ID",hue='DEALSIZE', data=df)
plt.show()
```



Observation:

- Sales in the YEAR 2004 was highest followed by the Sales in YEAR 2003 followed by YEAR 2005.

PRODUCTLINE

In [108]:

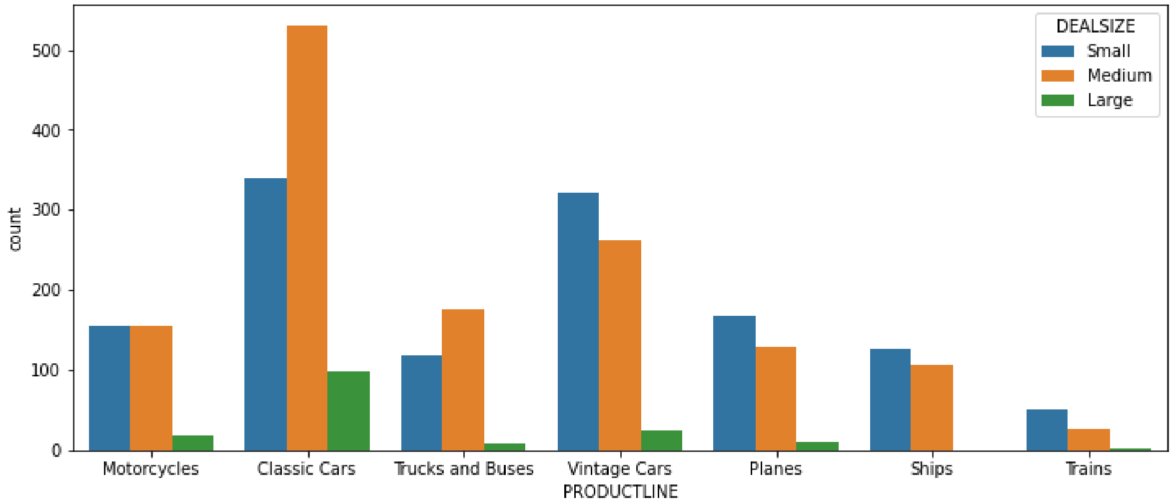
```
df[['PRODUCTLINE']].describe()
```

Out[108]:

	PRODUCTLINE
count	2823
unique	7
top	Classic Cars
freq	967

In [114]:

```
plt.figure(figsize=(12,5))
sns.countplot(x="PRODUCTLINE",hue='DEALSIZE', data=df)
plt.show()
```



Observation:

- Sales for the Cars was highest and lowest for Trains.

MSRP

In [115]:

```
df[['MSRP']].describe()
```

Out[115]:

	MSRP
count	2823.000000
mean	100.715551
std	40.187912
min	33.000000
25%	68.000000
50%	99.000000
75%	124.000000
max	214.000000

In [116]:

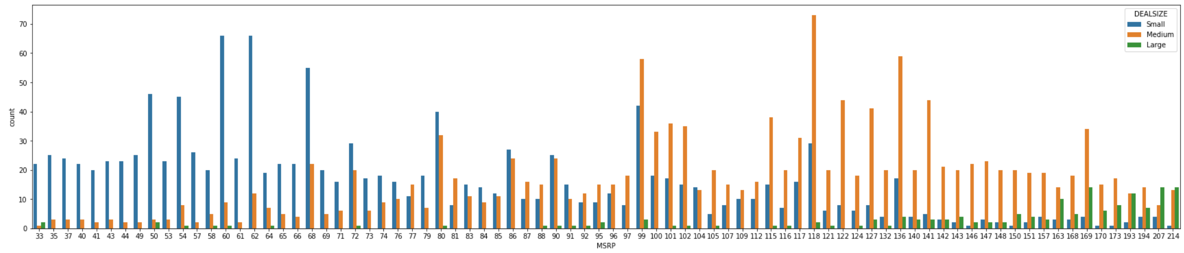
```
df[['MSRP']].nunique()
```

Out[116]:

```
MSRP    80
dtype: int64
```

In [120]:

```
plt.figure(figsize=(30,6))
sns.countplot(x="MSRP",hue='DEALSIZE', data=df)
plt.show()
```



Observation:

- MSRP also seems to be quite an important feature in classifying the DEALSIZE since majority of SMALL sized delas have a low MSRP whereas LARGE deals have mostly high MSRP.

> **PRODUCTCODE**

In [121]:

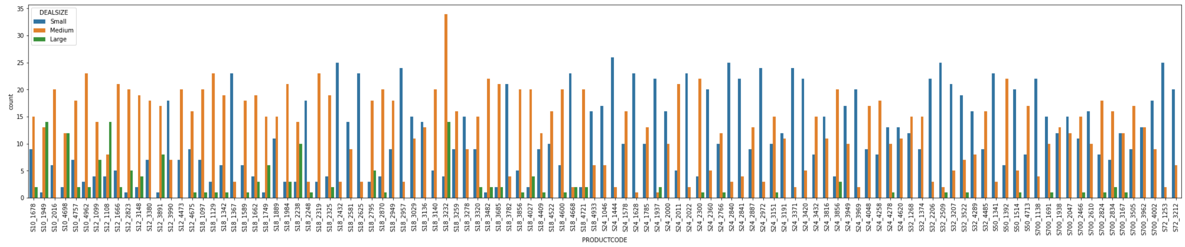
```
df[['PRODUCTCODE']].describe()
```

Out[121]:

	PRODUCTCODE
count	2823
unique	109
top	S18_3232
freq	52

In [126]:

```
plt.figure(figsize=(35, 6))
sns.countplot(x="PRODUCTCODE",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```



CUSTOMERNAME

In [127]:

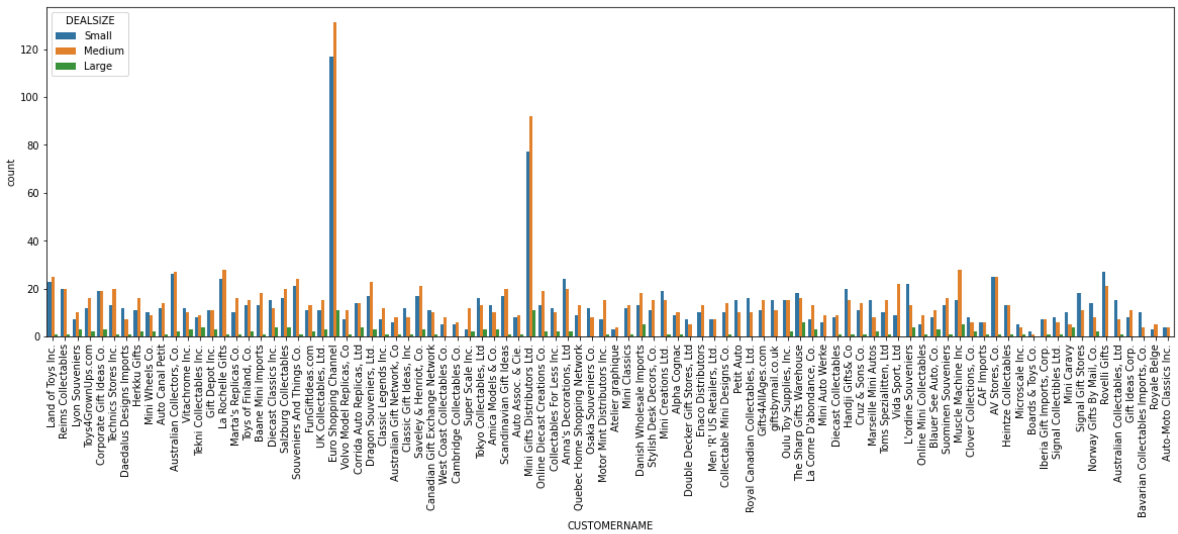
```
df[['CUSTOMERNAME']].describe()
```

Out[127]:

	CUSTOMERNAME
count	2823
unique	92
top	Euro Shopping Channel
freq	259

In [129]:

```
plt.figure(figsize=(20, 6))
sns.countplot(x="CUSTOMERNAME",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```



Observation:

- Most number of Purchase is made by Euro Shopping Channel and Mini Gifts Distribution Ltd.

PHONE

In [130]:

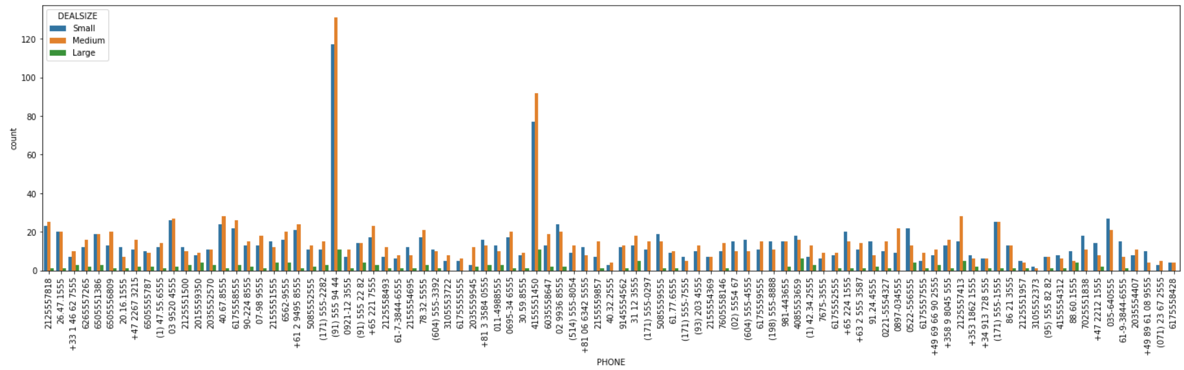
```
df[['PHONE']].describe()
```

Out[130]:

	PHONE
count	2823
unique	91
top	(91) 555 94 44
freq	259

In [131]:

```
plt.figure(figsize=(25, 6))
sns.countplot(x="PHONE",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```



ADDRESSLINE1

In [132]:

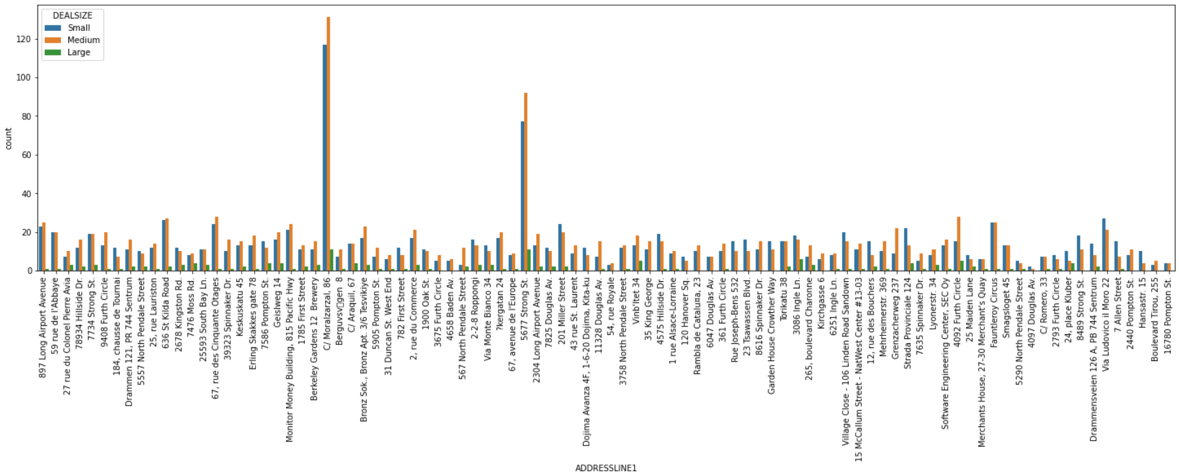
```
df[['ADDRESSLINE1']].describe()
```

Out[132]:

	ADDRESSLINE1
count	2823
unique	92
top	C/ Moralzarzal, 86
freq	259

In [136]:

```
plt.figure(figsize=(25, 6))
sns.countplot(x="ADDRESSLINE1",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```



ADDRESSLINE2

In [137]:

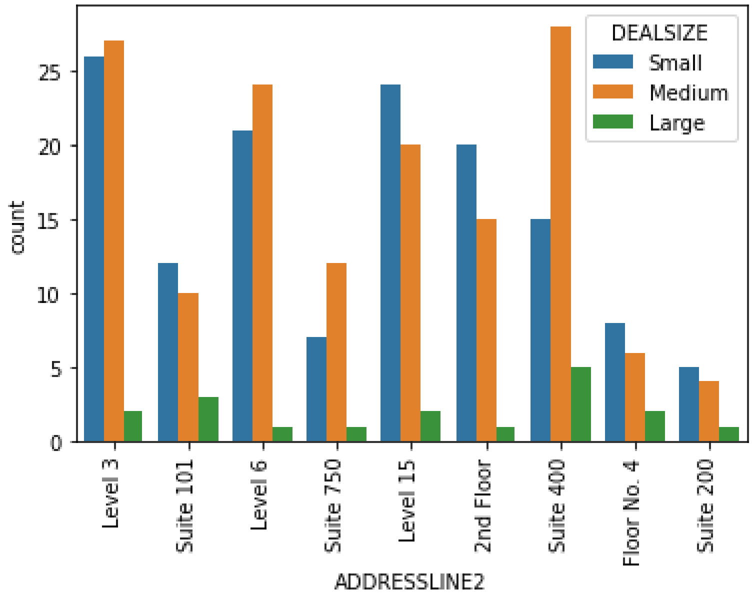
```
df[['ADDRESSLINE2']].describe()
```

Out[137]:

	ADDRESSLINE2
count	302
unique	9
top	Level 3
freq	55

In [138]:

```
sns.countplot(x="ADDRESSLINE2",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```



Observation:

- We have very few values for ADDRESSLINE2 as compared to ADDRESSLINE1 since the majority of the values in ADDRESSLINE2 are missing.

CITY

In [139]:

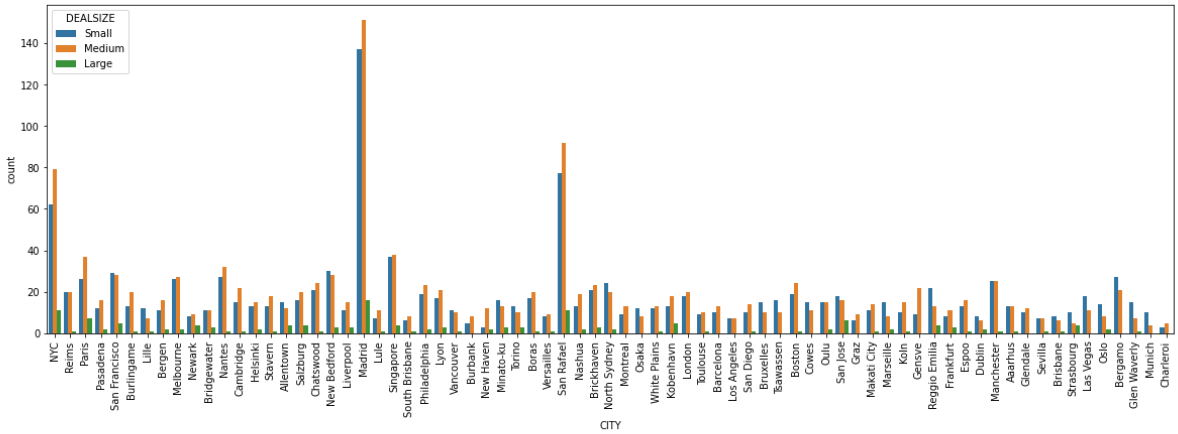
```
df[['CITY']].describe()
```

Out[139]:

	CITY
count	2823
unique	73
top	Madrid
freq	304

In [140]:

```
plt.figure(figsize=(20, 6))
sns.countplot(x="CITY",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```



STATE

In [143]:

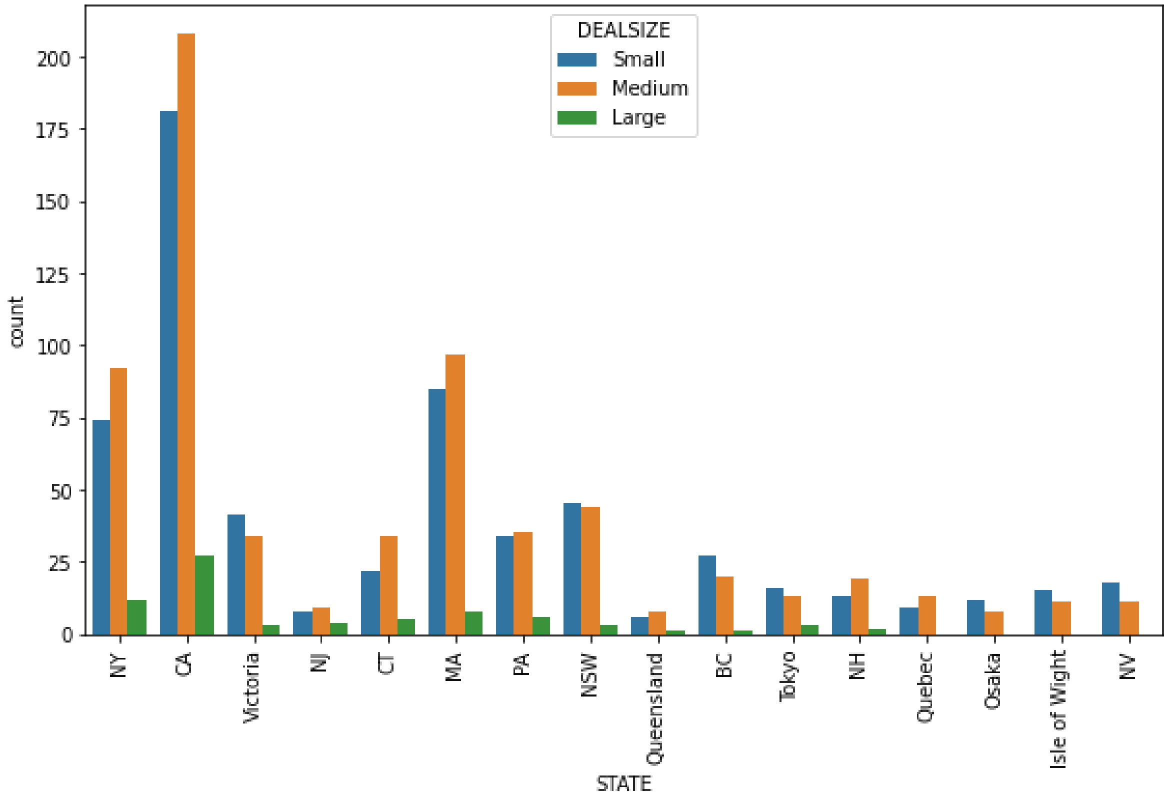
```
df[['STATE']].describe()
```

Out[143]:

	STATE
count	1337
unique	16
top	CA
freq	416

In [144]:

```
plt.figure(figsize=(10, 6))
sns.countplot(x="STATE",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```



Observatin:

- Majority of the sales corresponds to NY, CA and MA.

POSTALCODE

In [145]:

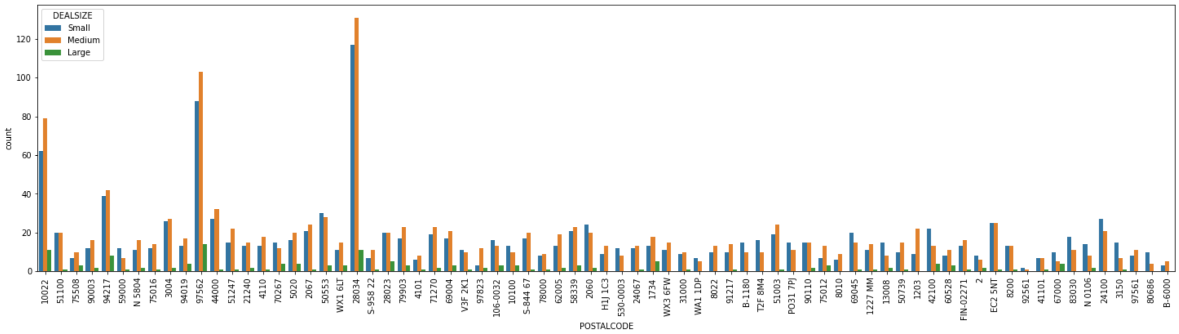
```
df[['POSTALCODE']].describe()
```

Out[145]:

	POSTALCODE
count	2747
unique	73
top	28034
freq	259

In [146]:

```
plt.figure(figsize=(25, 6))
sns.countplot(x="POSTALCODE",hue='DEALSIZE', data=df)
plt.xticks(rotation=90)
plt.show()
```

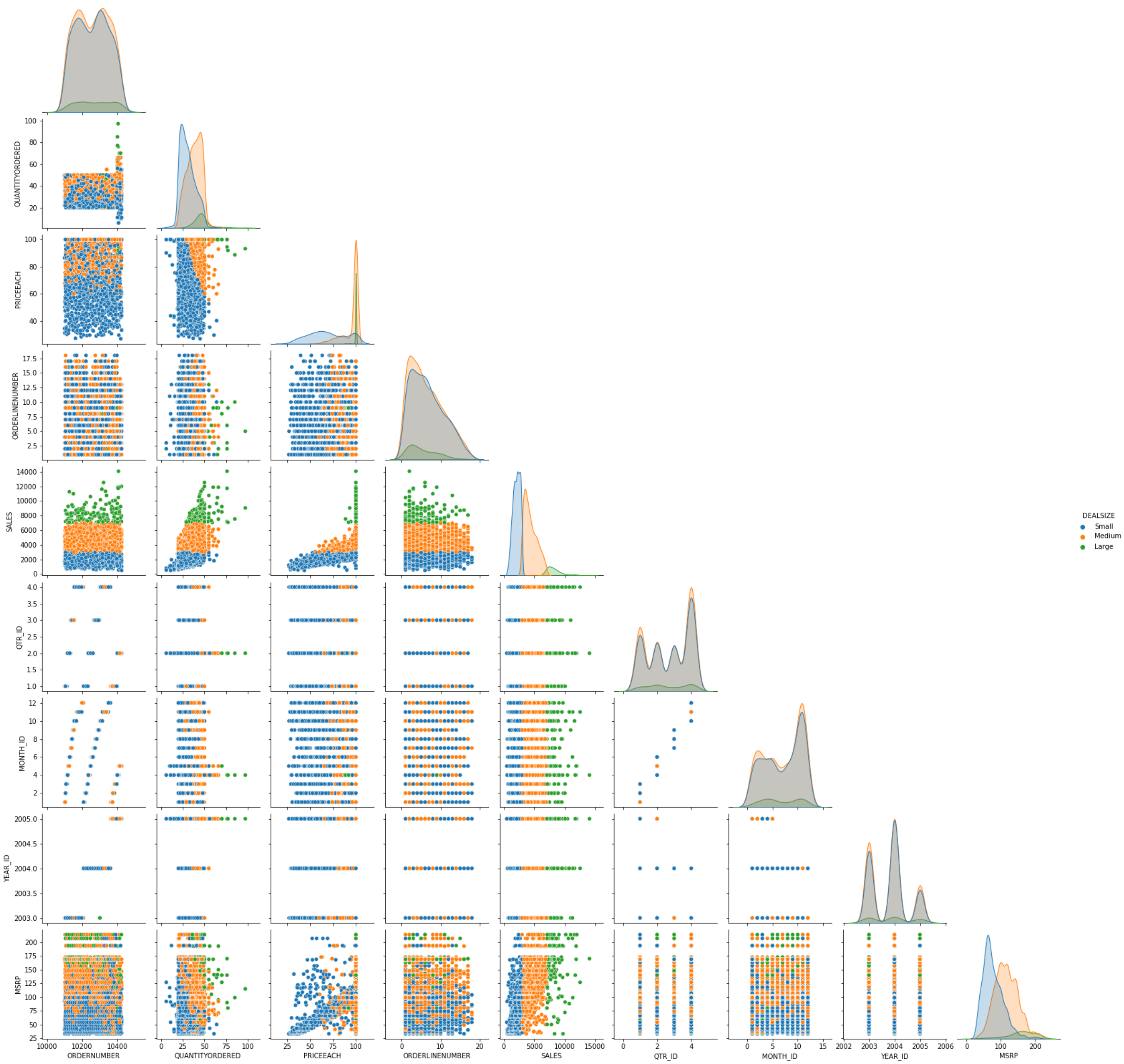


Bivariate Analysis

PairPlots

In [148]:

```
sns.pairplot(data=df, hue='DEALSIZE', corner=True)
plt.show()
```



Observation:

- SALES seems to be the most critical feature in deciding the DEALSIZE.