In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df1=pd.read_csv('sales.csv', encoding='ISO-8859-1')
df2=pd.read_json('sales.json')
df3=pd.read_excel('sales.xlsx')
```

In [4]:
```
df1.head()
```

Out[4]:

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE |
|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 |

5 rows × 25 columns

In [5]:
```
df2.head()
```

Out[5]:

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE |
|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 |

5 rows × 25 columns

In [6]:
```
df3.head()
```

Out[6]:

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE |
|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 |

5 rows × 25 columns

In [7]:
```
df1.shape
```

Out[7]:
```
(2823, 25)
```

In [8]:
```
df2.shape
```

Out[8]:
```
(2823, 25)
```

In [9]:
```
df3.shape
```

Out[9]:
```
(2823, 25)
```

In [10]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

In [12]:
```
#Checks for NA values in columns
df1.isna().sum()
```

Out[12]:
```
ORDERNUMBER          0
QUANTITYORDERED      0
PRICEEACH            0
ORDERLINENUMBER      0
SALES                0
ORDERDATE            0
STATUS               0
QTR_ID               0
MONTH_ID             0
YEAR_ID              0
PRODUCTLINE          0
MSRP                 0
PRODUCTCODE          0
CUSTOMERNAME         0
PHONE                0
ADDRESSLINE1         0
ADDRESSLINE2      2521
CITY                 0
STATE             1486
POSTALCODE          76
COUNTRY              0
TERRITORY         1074
CONTACTLASTNAME      0
CONTACTFIRSTNAME     0
DEALSIZE             0
dtype: int64
```

In [11]:
```
#for calculating some statistical data like percentile, mean and std of the numeric
df1.describe()
```

Out[11]:

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | |
|---|---|---|---|---|---|---|
| **count** | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 282 |
| **mean** | 10258.725115 | 35.092809 | 83.658544 | 6.466171 | 3553.889072 | |
| **std** | 92.085478 | 9.741443 | 20.174277 | 4.225841 | 1841.865106 | |
| **min** | 10100.000000 | 6.000000 | 26.880000 | 1.000000 | 482.130000 | |
| **25%** | 10180.000000 | 27.000000 | 68.860000 | 3.000000 | 2203.430000 | |
| **50%** | 10262.000000 | 35.000000 | 95.700000 | 6.000000 | 3184.800000 | |
| **75%** | 10333.500000 | 43.000000 | 100.000000 | 9.000000 | 4508.000000 | |
| **max** | 10425.000000 | 97.000000 | 100.000000 | 18.000000 | 14082.800000 | |

In [12]:
```
#Dropping unnecessary columns
df1 = df1.drop(['ADDRESSLINE1','ADDRESSLINE2','CITY','STATE','TERRITORY'],axis = 1)
```

In [13]:
```
df1.isna().sum()
```

Out[13]:
```
ORDERNUMBER          0
QUANTITYORDERED      0
PRICEEACH            0
ORDERLINENUMBER      0
SALES                0
ORDERDATE            0
STATUS               0
QTR_ID               0
MONTH_ID             0
YEAR_ID              0
PRODUCTLINE          0
MSRP                 0
PRODUCTCODE          0
CUSTOMERNAME         0
PHONE                0
POSTALCODE          76
COUNTRY              0
CONTACTLASTNAME      0
CONTACTFIRSTNAME     0
DEALSIZE             0
dtype: int64
```

In [14]:
```
#Filling all NA values with mode of the POSTALCODE column
df1 = df1['POSTALCODE'].fillna(df1.POSTALCODE.mode(), inplace=True)
```

In [15]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      2823 non-null   object
 17  CITY              2823 non-null   object
 18  STATE             2823 non-null   object
 19  POSTALCODE        2823 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         2823 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

In [16]:
```
df2.isna().sum()
```

Out[16]:
```
ORDERNUMBER         0
QUANTITYORDERED     0
PRICEEACH           0
ORDERLINENUMBER     0
SALES               0
ORDERDATE           0
STATUS              0
QTR_ID              0
MONTH_ID            0
YEAR_ID             0
PRODUCTLINE         0
MSRP                0
PRODUCTCODE         0
CUSTOMERNAME        0
PHONE               0
ADDRESSLINE1        0
ADDRESSLINE2        0
CITY                0
STATE               0
POSTALCODE          0
COUNTRY             0
TERRITORY           0
CONTACTLASTNAME     0
CONTACTFIRSTNAME    0
DEALSIZE            0
dtype: int64
```

In [17]:
```
df2 = df2.drop(['ADDRESSLINE1','ADDRESSLINE2','CITY','STATE','TERRITORY'],axis = 1)
```

In [18]:
```
df2.describe()
```

Out[18]:

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | |
|---|---|---|---|---|---|---|
| **count** | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 282 |
| **mean** | 10258.725115 | 35.092809 | 83.658544 | 6.466171 | 3553.889072 | |
| **std** | 92.085478 | 9.741443 | 20.174277 | 4.225841 | 1841.865106 | |
| **min** | 10100.000000 | 6.000000 | 26.880000 | 1.000000 | 482.130000 | |
| **25%** | 10180.000000 | 27.000000 | 68.860000 | 3.000000 | 2203.430000 | |
| **50%** | 10262.000000 | 35.000000 | 95.700000 | 6.000000 | 3184.800000 | |
| **75%** | 10333.500000 | 43.000000 | 100.000000 | 9.000000 | 4508.000000 | |
| **max** | 10425.000000 | 97.000000 | 100.000000 | 18.000000 | 14082.800000 | |

In [19]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

In [20]:
```
df2.isna().sum()
```

Out[20]:
```
ORDERNUMBER          0
QUANTITYORDERED      0
PRICEEACH            0
ORDERLINENUMBER      0
SALES                0
ORDERDATE            0
STATUS               0
QTR_ID               0
MONTH_ID             0
YEAR_ID              0
PRODUCTLINE          0
MSRP                 0
PRODUCTCODE          0
CUSTOMERNAME         0
PHONE                0
POSTALCODE           0
COUNTRY              0
CONTACTLASTNAME      0
CONTACTFIRSTNAME     0
DEALSIZE             0
dtype: int64
```

In [21]:
```
df3.describe()
```

Out[21]:

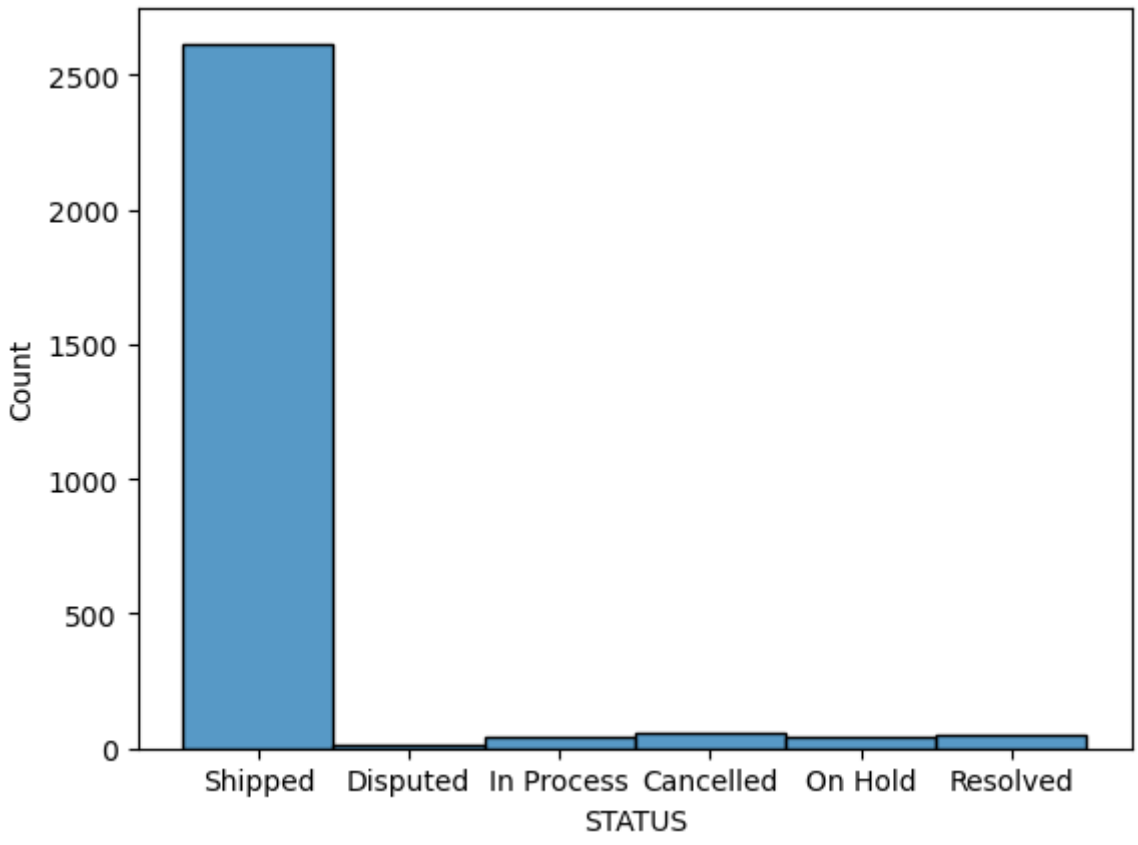
| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | |
|---|---|---|---|---|---|---|
| **count** | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 282 |
| **mean** | 10258.725115 | 35.092809 | 83.658544 | 6.466171 | 3553.889072 | |
| **std** | 92.085478 | 9.741443 | 20.174277 | 4.225841 | 1841.865106 | |
| **min** | 10100.000000 | 6.000000 | 26.880000 | 1.000000 | 482.130000 | |
| **25%** | 10180.000000 | 27.000000 | 68.860000 | 3.000000 | 2203.430000 | |
| **50%** | 10262.000000 | 35.000000 | 95.700000 | 6.000000 | 3184.800000 | |
| **75%** | 10333.500000 | 43.000000 | 100.000000 | 9.000000 | 4508.000000 | |
| **max** | 10425.000000 | 97.000000 | 100.000000 | 18.000000 | 14082.800000 | |

In [22]:
```
import pandas as pd

# Load the CSV file into a DataFrame
df1 = pd.read_csv('sales.csv',encoding='ISO-8859-1')
df2 = pd.read_json('sales.json')
df3 = pd.read_excel('sales.xlsx')
# Check the data types of columns
data_types = df1.dtypes
print(data_types)
```

```
ORDERNUMBER           int64
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
ORDERDATE            object
STATUS               object
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
PRODUCTLINE          object
MSRP                  int64
PRODUCTCODE          object
CUSTOMERNAME         object
PHONE                object
ADDRESSLINE1         object
ADDRESSLINE2         object
CITY                 object
STATE                object
POSTALCODE           object
COUNTRY              object
TERRITORY            object
CONTACTLASTNAME      object
CONTACTFIRSTNAME     object
DEALSIZE             object
dtype: object
```

In [23]:

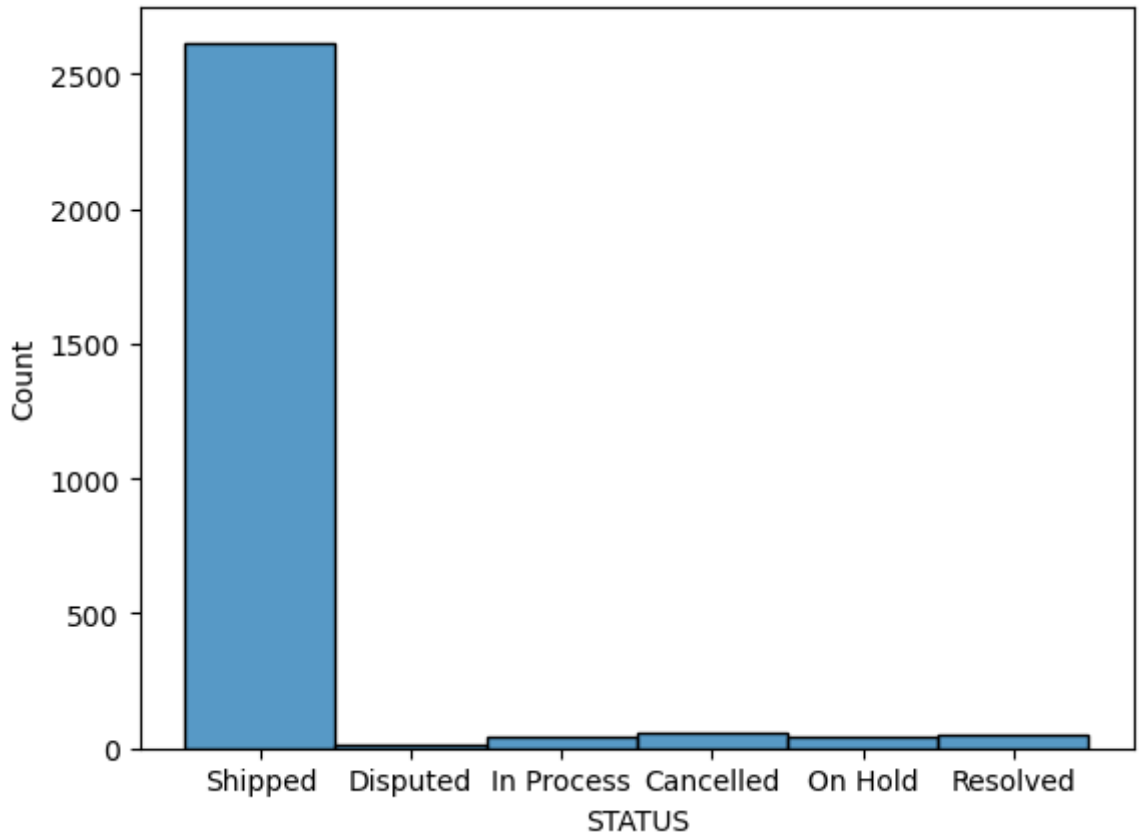
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming your DataFrame is named df1
sns.histplot(x='STATUS', data=df1)
plt.show()
```



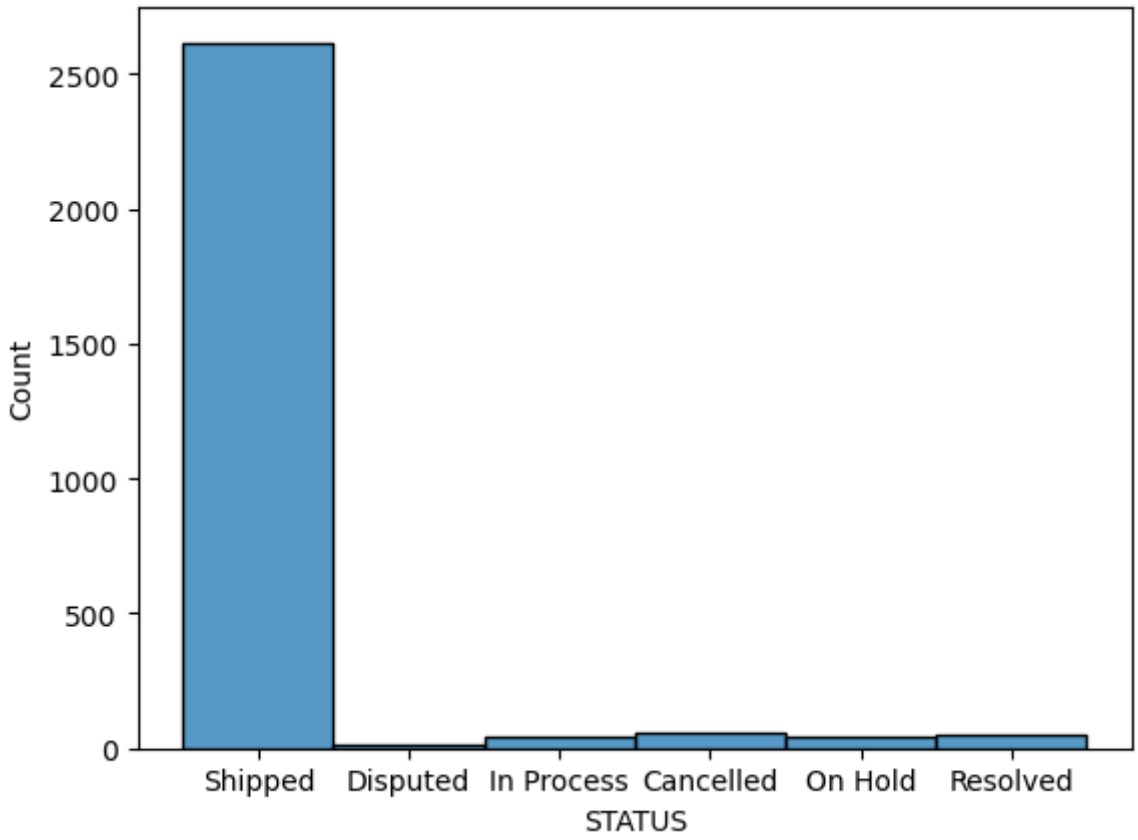

In [24]:

```
#Plotting histogram plot for STATUS column
```

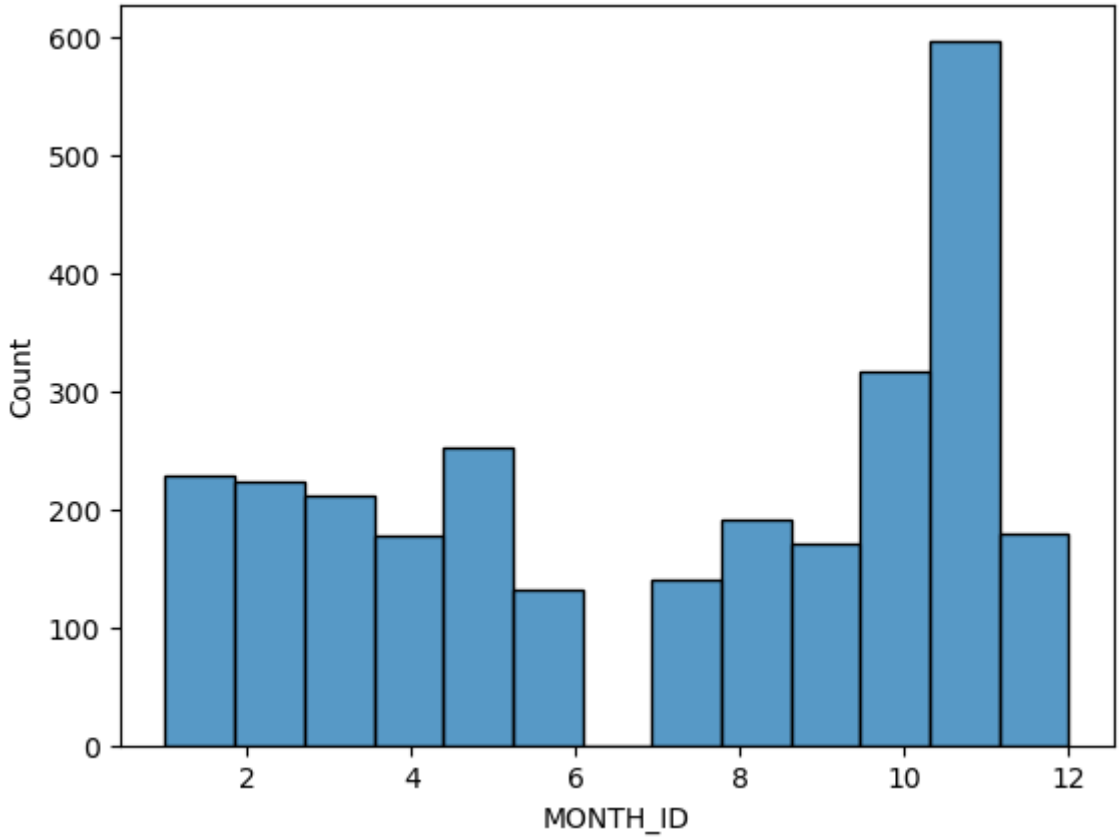
```
import seaborn as sns
import matplotlib.pyplot as plt


sns.histplot(x='STATUS', data=df2, )
plt.show()
```



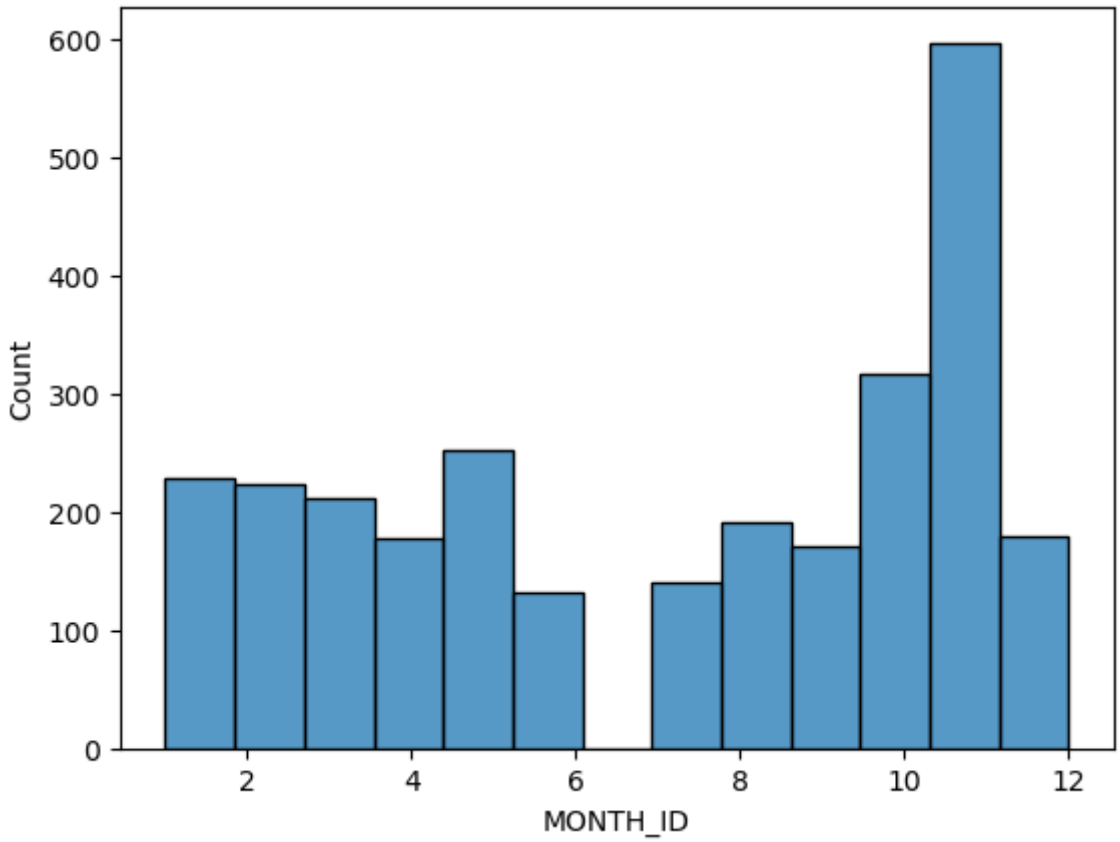

In [25]:
```
#Plotting histogram plot for STATUS column

import seaborn as sns
import matplotlib.pyplot as plt


sns.histplot(x='STATUS', data=df3, )
plt.show()
```



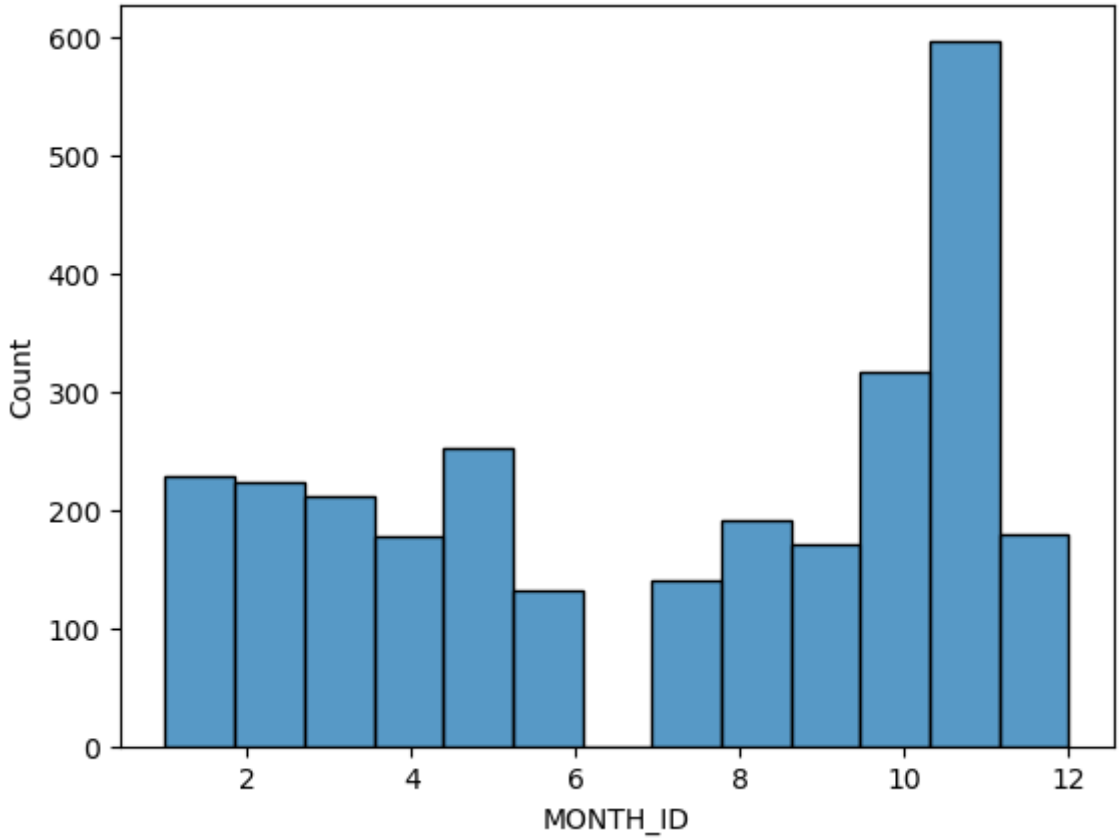

In [26]:
```
#Plotting histogram plot for MONTH_ID column
import seaborn as sns
import matplotlib.pyplot as plt


sns.histplot(x='MONTH_ID', data=df1, )
plt.show()
```



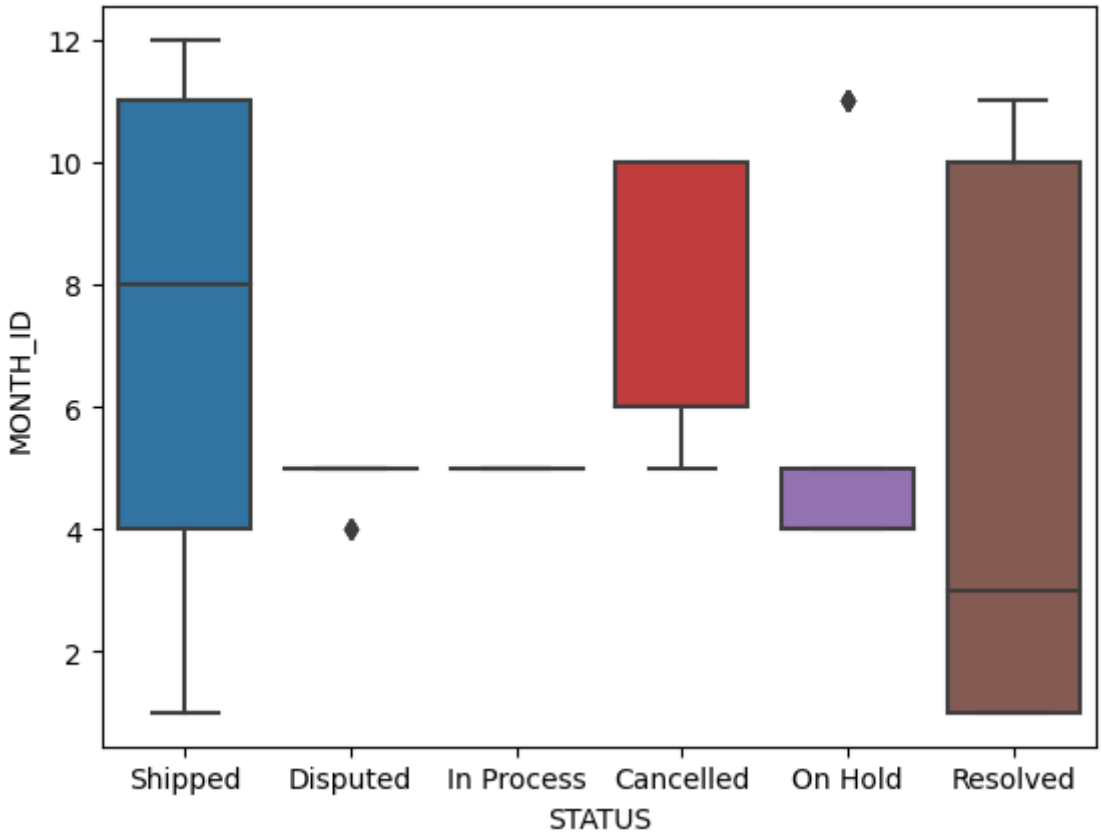
In [27]:
```
#Plotting histogram plot for MONTH_ID column
import seaborn as sns
import matplotlib.pyplot as plt


sns.histplot(x='MONTH_ID', data=df2, )
plt.show()
```



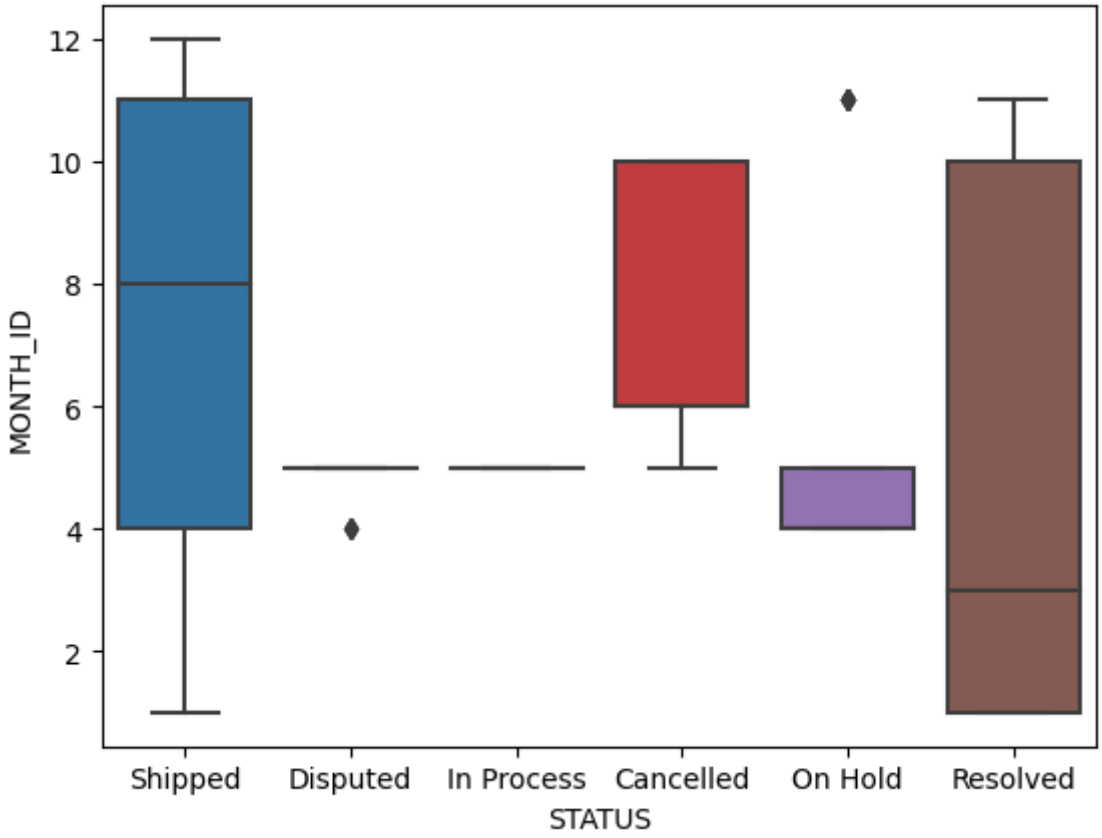

In [28]:
```
#Plotting histogram plot for MONTH_ID column
import seaborn as sns
import matplotlib.pyplot as plt


sns.histplot(x='MONTH_ID', data=df3, )
plt.show()
```



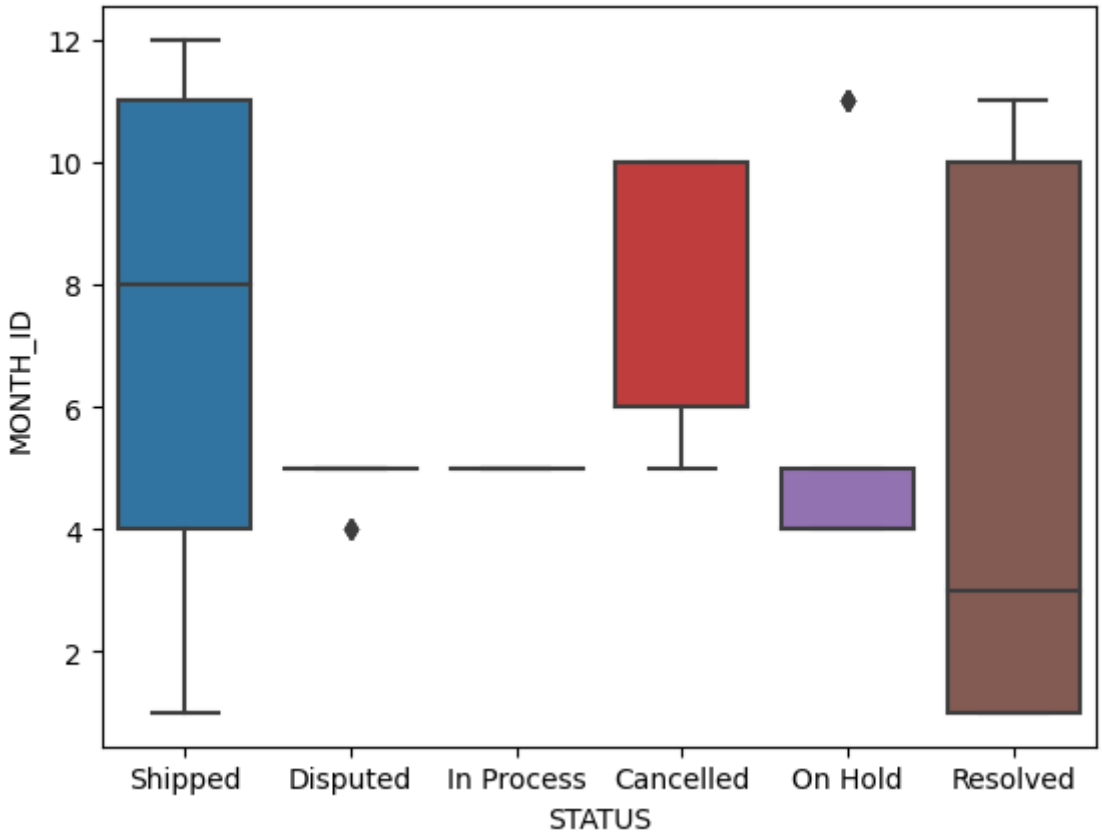

In [29]:
```
#Plotting boxplot for STATUS column against MONTH_ID column
sns.boxplot( x="STATUS", y= "MONTH_ID", data=df1, )
plt.show()
```




In [30]:
```
#Plotting boxplot for STATUS column against MONTH_ID column
sns.boxplot( x="STATUS", y= "MONTH_ID", data=df2, )
plt.show()
```

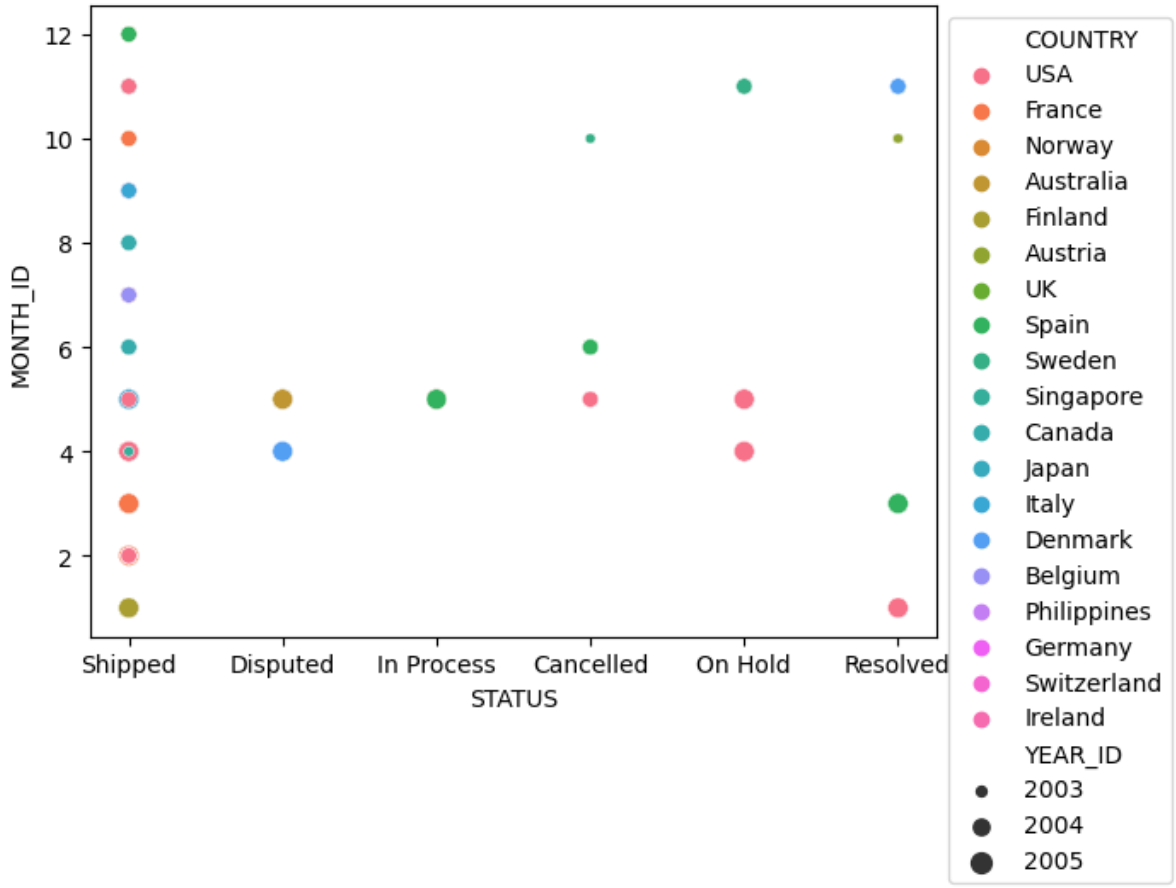



In [31]:
```
#Plotting boxplot for STATUS column against MONTH_ID column
sns.boxplot( x="STATUS", y= "MONTH_ID", data=df3, )
plt.show()
```

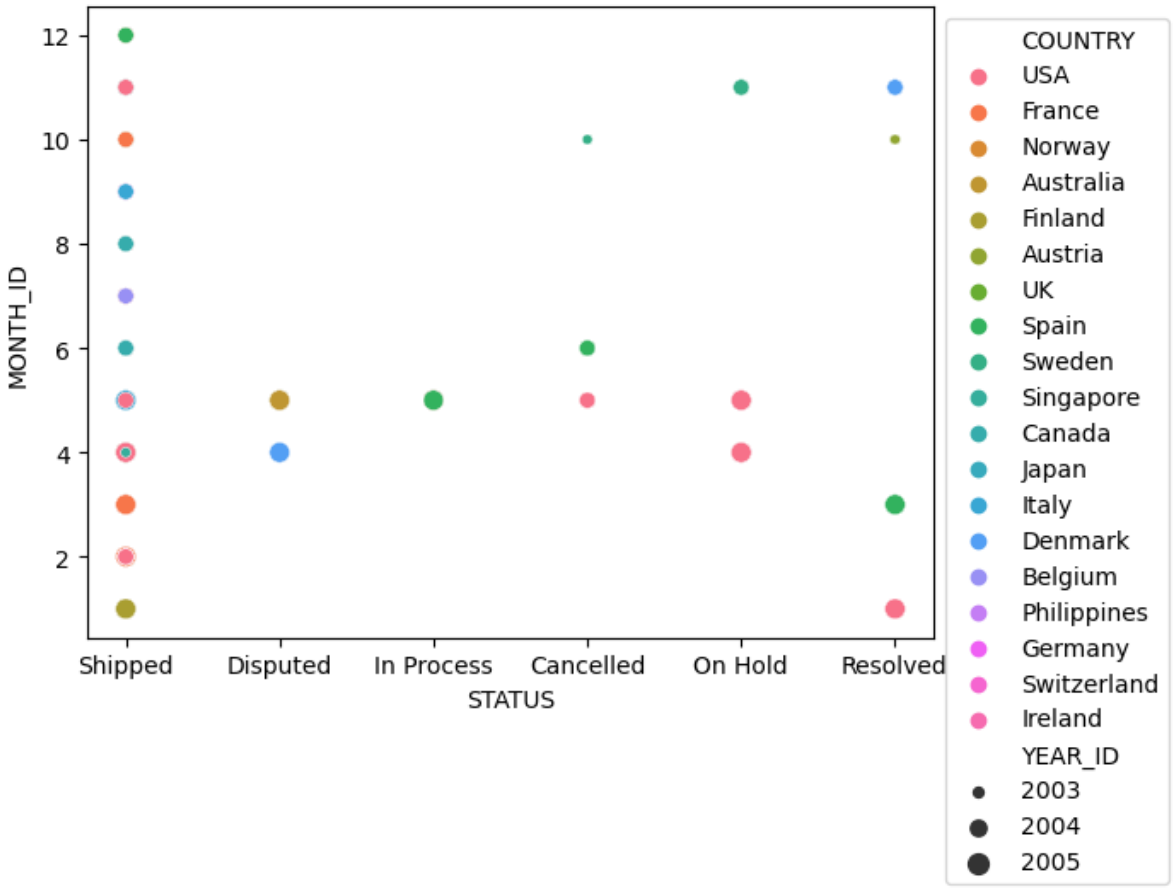



In [32]:
```
#Plotting Scatterplot
sns.scatterplot( x="STATUS", y="MONTH_ID", data=df1,
                hue='COUNTRY', size='YEAR_ID')

# Placing Legend outside the Figure
```

```
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```

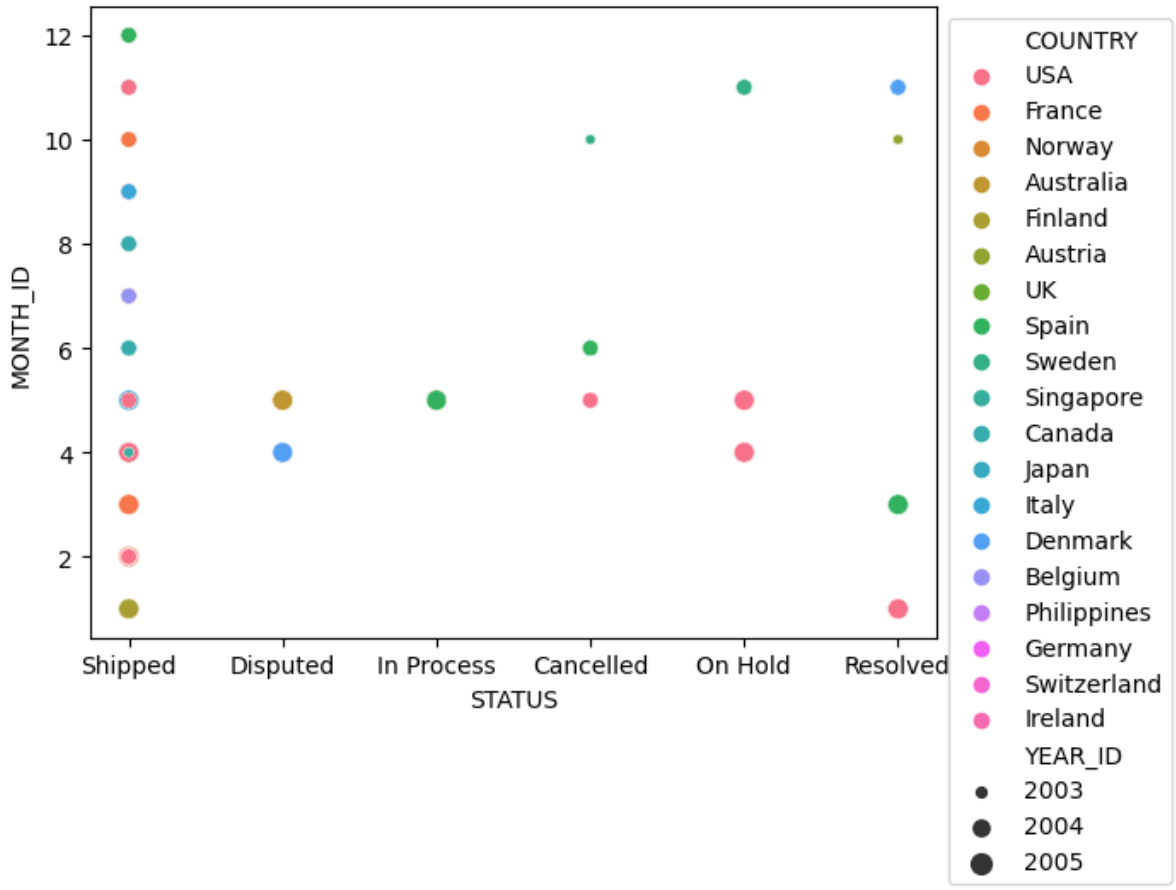



In [33]:
```
#Plotting Scatterplot
sns.scatterplot( x="STATUS", y="MONTH_ID", data=df2,
                hue='COUNTRY', size='YEAR_ID')

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```




In [34]:

```
#Plotting Scatterplot
sns.scatterplot( x="STATUS", y="MONTH_ID", data=df3,
                hue='COUNTRY', size='YEAR_ID')

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```




In [35]:
```
#Checking the data only for shipped STATUS
data1=df1[df1["STATUS"]=='Shipped']
data1.head()
```

Out[35]:

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE |
|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 |

5 rows × 25 columns

In [36]:
```
#Checking the data only for shipped STATUS
data2=df2[df2["STATUS"]=='Shipped']
data2.head()
```

Out[36]:

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE |
|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 |

5 rows × 25 columns

In [37]:
```
#Checking the data only for shipped STATUS
data3=df3[df3["STATUS"]=='Shipped']
data3.head()
```

Out[37]:

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE |
|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 |

5 rows × 25 columns

In [38]:
```
data1.shape
```

Out[38]:
```
(2617, 25)
```

In [39]:
```
data2.shape
```

Out[39]:
```
(2617, 25)
```

In [40]:
```
data3.shape
```

Out[40]:
```
(2617, 25)
```

In [41]:
```
#Calculating sum for sales column
sum_sales = df1['SALES'].sum()
print("Addition of all sales",sum_sales)
```

```
Addition of all sales 10032628.85
```

In [42]:
```
#Calculating sum for sales column
sum_sales = df2['SALES'].sum()
print("Addition of all sales",sum_sales)
```

```
Addition of all sales 10032628.85
```

In [43]:
```
#Calculating sum for sales column
sum_sales = df3['SALES'].sum()
print("Addition of all sales",sum_sales)
```

```
Addition of all sales 10032628.85
```

In [44]:
```
#Calulating average for sales column
sales_avg = df1['SALES'].mean()
print("Average of total sales = ",sales_avg)
```

```
Average of total sales =  3553.88907190932
```

In [45]:
```
#Calulating average for sales column
sales_avg = df2['SALES'].mean()
print("Average of total sales = ",sales_avg)
```

```
Average of total sales =  3553.88907190932
```

In [46]:
```
#Calulating average for sales column
sales_avg = df3['SALES'].mean()
print("Average of total sales = ",sales_avg)
```

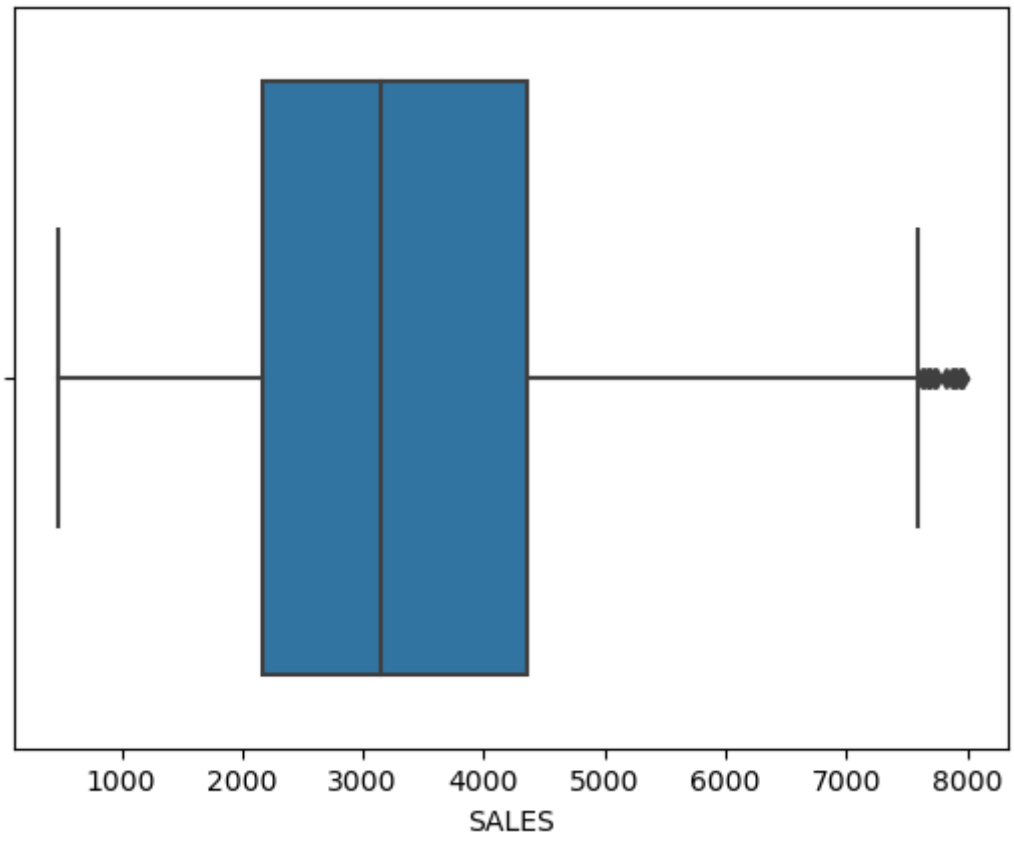
```
Average of total sales =  3553.88907190932
```

In [47]:
```
import sklearn
import pandas as pd
import seaborn as sns

# IQR
Q1 = np.percentile(df1['SALES'], 25,
                                  interpolation = 'midpoint')

Q3 = np.percentile(df1['SALES'], 75,
                                  interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df1.shape)

# Upper bound
upper = np.where(df1['SALES'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(df1['SALES'] <= (Q1-1.5*IQR))

# Removing the Outliers
df1.drop(upper[0], inplace = True)
df1.drop(lower[0], inplace = True)

print("New Shape: ", df1.shape)

sns.boxplot(x='SALES', data=df1)
```

```
Old Shape:  (2823, 25)
New Shape:  (2742, 25)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_4980\1327906522.py:6: Deprecat
ionWarning: the `interpolation=` argument to percentile was renamed to `method=`,
which has additional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to r
eview the method they used. (Deprecated NumPy 1.22)
  Q1 = np.percentile(df1['SALES'], 25,
C:\Users\Administrator\AppData\Local\Temp\ipykernel_4980\1327906522.py:9: Deprecat
ionWarning: the `interpolation=` argument to percentile was renamed to `method=`,
which has additional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to r
eview the method they used. (Deprecated NumPy 1.22)
  Q3 = np.percentile(df1['SALES'], 75,
```

Out[47]:
```
<AxesSubplot:xlabel='SALES'>
```




In [48]:
```
import sklearn
import pandas as pd
import seaborn as sns

# IQR
Q1 = np.percentile(df2['SALES'], 25,
                                  interpolation = 'midpoint')

Q3 = np.percentile(df2['SALES'], 75,
                                  interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df2.shape)

# Upper bound
upper = np.where(df2['SALES'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(df2['SALES'] <= (Q1-1.5*IQR))

# Removing the Outliers
df2.drop(upper[0], inplace = True)
```

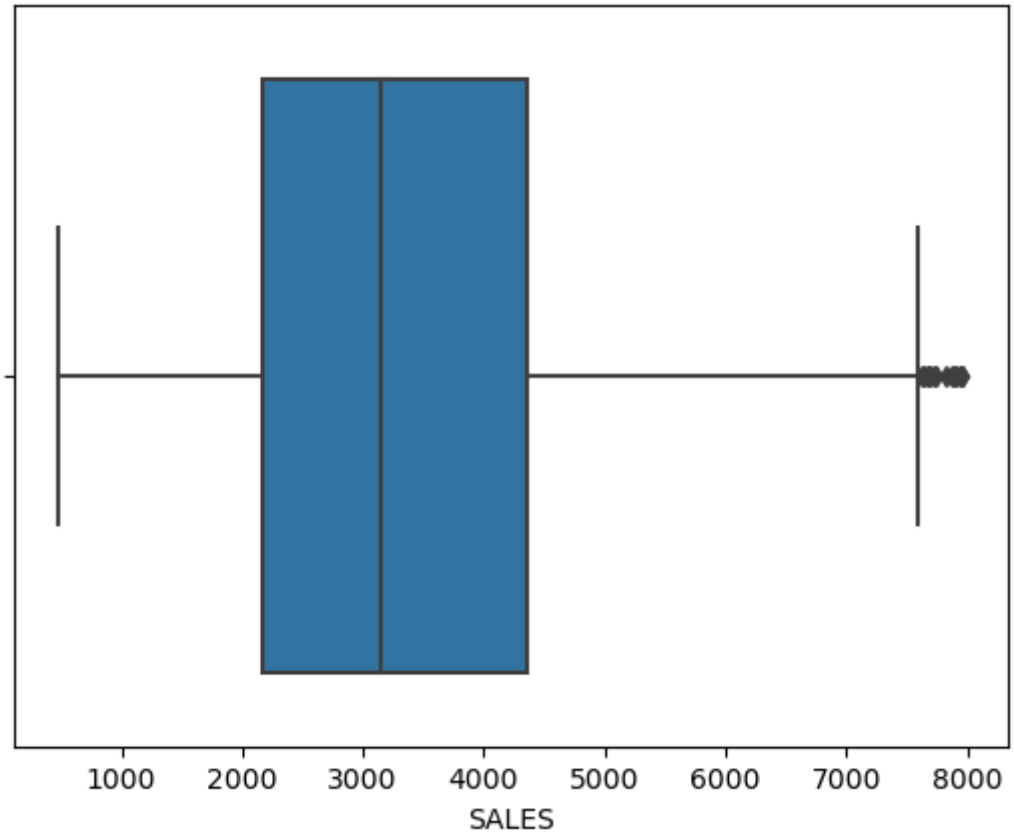
```
df2.drop(lower[0], inplace = True)

print("New Shape: ", df2.shape)

sns.boxplot(x='SALES', data=df2)
```

```
Old Shape:  (2823, 25)
New Shape:  (2742, 25)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_4980\3488628968.py:6: Deprecat
ionWarning: the `interpolation=` argument to percentile was renamed to `method=`,
which has additional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to r
eview the method they used. (Deprecated NumPy 1.22)
  Q1 = np.percentile(df2['SALES'], 25,
C:\Users\Administrator\AppData\Local\Temp\ipykernel_4980\3488628968.py:9: Deprecat
ionWarning: the `interpolation=` argument to percentile was renamed to `method=`,
which has additional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to r
eview the method they used. (Deprecated NumPy 1.22)
  Q3 = np.percentile(df2['SALES'], 75,
```

Out[48]:

```
<AxesSubplot:xlabel='SALES'>
```




In [49]:

```
import sklearn
import pandas as pd
import seaborn as sns

# IQR
Q1 = np.percentile(df3['SALES'], 25,
                                  interpolation = 'midpoint')

Q3 = np.percentile(df3['SALES'], 75,
                                  interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df3.shape)
```

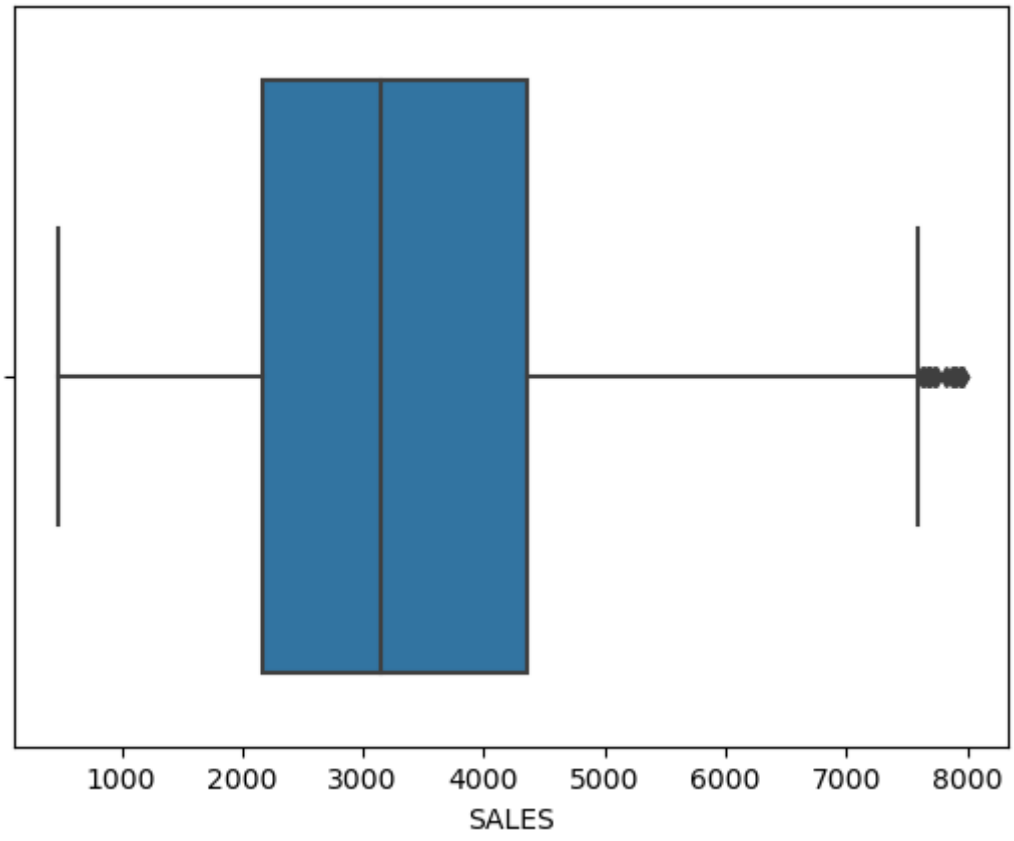
```
# Upper bound
upper = np.where(df3['SALES'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(df3['SALES'] <= (Q1-1.5*IQR))

# Removing the Outliers
df3.drop(upper[0], inplace = True)
df3.drop(lower[0], inplace = True)

print("New Shape: ", df3.shape)

sns.boxplot(x='SALES', data=df3)
```

```
Old Shape:  (2823, 25)
New Shape:  (2742, 25)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_4980\4040548419.py:6: Deprecat
ionWarning: the `interpolation=` argument to percentile was renamed to `method=`,
which has additional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to r
eview the method they used. (Deprecated NumPy 1.22)
  Q1 = np.percentile(df3['SALES'], 25,
C:\Users\Administrator\AppData\Local\Temp\ipykernel_4980\4040548419.py:9: Deprecat
ionWarning: the `interpolation=` argument to percentile was renamed to `method=`,
which has additional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to r
eview the method they used. (Deprecated NumPy 1.22)
  Q3 = np.percentile(df3['SALES'], 75,
```

Out[49]:

```
<AxesSubplot:xlabel='SALES'>
```




In [ ]: