In [46]:
```
#Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
%matplotlib inline
```

In [47]:
```
#read data and display the first 5 rows

auto_df = pd.read_excel("/content/Sales_Data.xlsx")

auto_df.head()
```

Out[47]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	DAYS_SINCE_LASTORDER	STA
0	10107	30	95.70	2	2871.00	43155	828	Ship
1	10121	34	81.35	5	2765.90	43227	757	Ship
2	10134	41	94.74	2	3884.34	43282	703	Ship
3	10145	45	83.26	6	3746.70	43337	649	Ship
4	10168	36	96.66	1	3479.76	43401	586	Ship

In [48]:
```
#View data information....data types for columns
auto_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2747 entries, 0 to 2746
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   ORDERNUMBER           2747 non-null   int64  
 1   QUANTITYORDERED       2747 non-null   int64  
 2   PRICEEACH             2747 non-null   float64
 3   ORDERLINENUMBER       2747 non-null   int64  
 4   SALES                 2747 non-null   float64
 5   ORDERDATE             2747 non-null   int64  
 6   DAYS_SINCE_LASTORDER  2747 non-null   int64  
 7   STATUS                2747 non-null   object 
 8   PRODUCTLINE           2747 non-null   object 
 9   MSRP                  2747 non-null   int64  
 10  PRODUCTCODE           2747 non-null   object 
 11  CUSTOMERNAME          2747 non-null   object 
 12  PHONE                 2747 non-null   object 
 13  ADDRESSLINE1          2747 non-null   object 
 14  CITY                  2747 non-null   object 
 15  POSTALCODE            2747 non-null   object 
 16  COUNTRY               2747 non-null   object 
 17  CONTACTLASTNAME       2747 non-null   object 
 18  CONTACTFIRSTNAME      2747 non-null   object 
 19  DEALSIZE              2747 non-null   object 
dtypes: float64(2), int64(6), object(12)
memory usage: 429.3+ KB
```

The data constitutes of the following data types;

1. 6 columns with int64 data types
2. columns with float64 data types
3. The remaining columns have object data type

In [49]:

```
#display column names
auto_df.columns
```

Out[49]:

```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
       'SALES', 'ORDERDATE', 'DAYS_SINCE_LASTORDER', 'STATUS', 'PRODUCTLINE',
       'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE', 'ADDRESSLINE1', 'CITY',
       'POSTALCODE', 'COUNTRY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME',
       'DEALSIZE'],
      dtype='object')
```

In [50]:

```
#to make EDA easier we drop some fields and convert date fields to date format
temp = ['PHONE','ADDRESSLINE1', 'CITY', 'POSTALCODE', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME']
auto_df.drop(temp, axis=1, inplace=True)
```

In [51]:

```
auto_df.head()
```

Out[51]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	DAYS_SINCE_LASTORDER	STA
0	10107	30	95.70	2	2871.00	43155	828	Ship
1	10121	34	81.35	5	2765.90	43227	757	Ship
2	10134	41	94.74	2	3884.34	43282	703	Ship
3	10145	45	83.26	6	3746.70	43337	649	Ship
4	10168	36	96.66	1	3479.76	43401	586	Ship

In [52]:

```
#count the number of NaN values in each
auto_df.isnull().sum()
```

Out[52]:

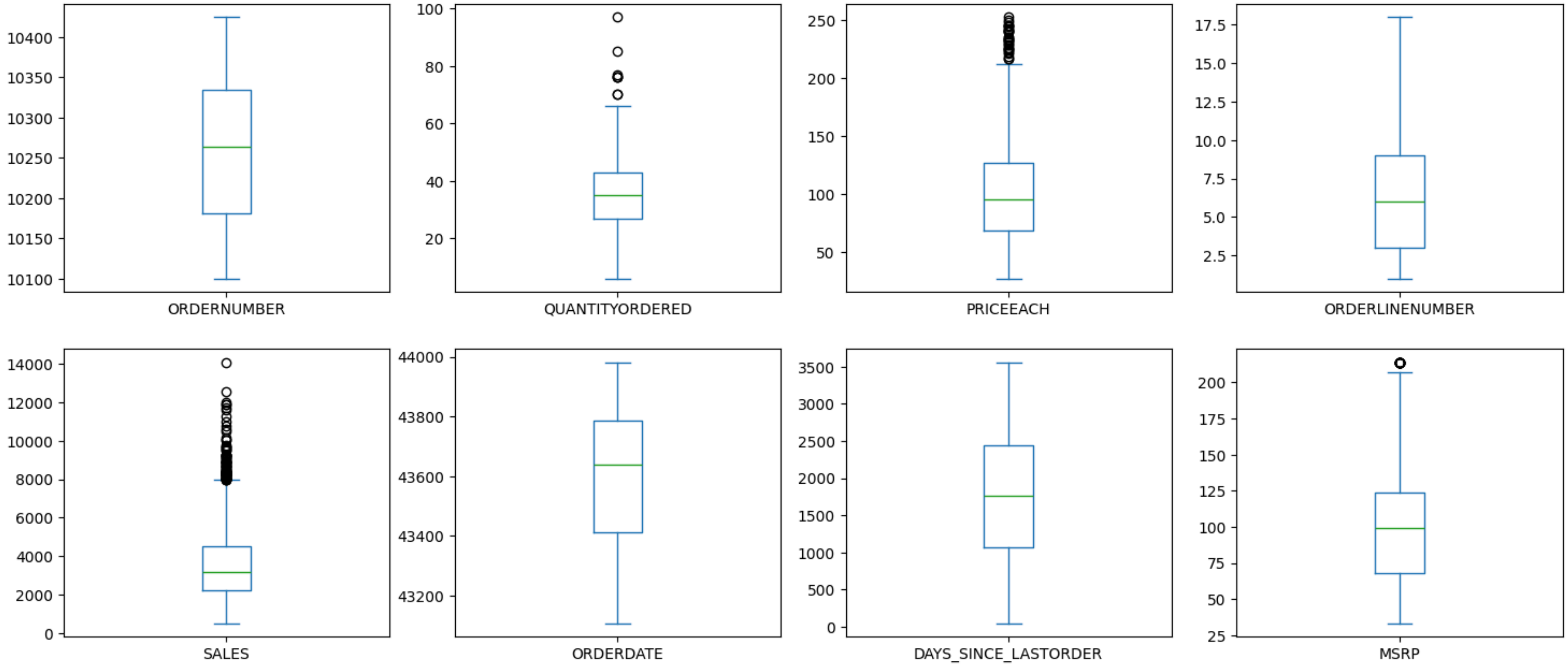
	0
ORDERNUMBER	0
QUANTITYORDERED	0
PRICEEACH	0
ORDERLINENUMBER	0
SALES	0
ORDERDATE	0
DAYS_SINCE_LASTORDER	0
STATUS	0
PRODUCTLINE	0
MSRP	0
PRODUCTCODE	0
CUSTOMERNAME	0
COUNTRY	0
DEALSIZE	0

dtype: int64

There are no null values in the data set

In [53]:

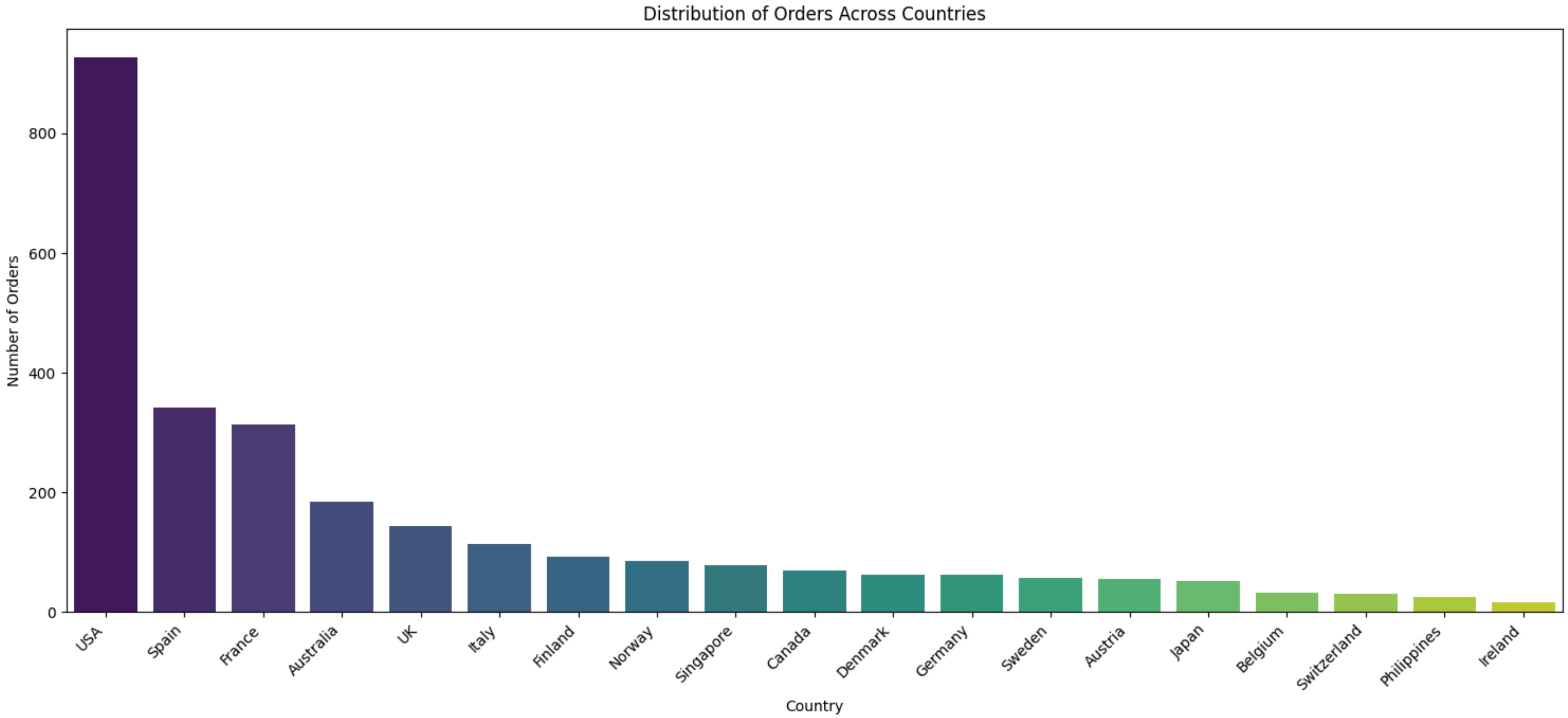
```python
#generate box plots for numarical variables to see how the values are distributed
plt.rcParams['figure.figsize'] = [18,16]
auto_df.plot(kind="box", subplots=True, layout=(4,4), sharex=False, sharey=False)
plt.show()
```



Most of the values are distributed normally around the mean apart from QUANTITYORDERED, PRICEEACH and SALES which have outliers

In [54]:

```
country_order_counts = auto_df['COUNTRY'].value_counts().reset_index()
country_order_counts.columns = ['Country', 'Order_Count']

plt.figure(figsize=(15, 7))
sns.barplot(x='Country', y='Order_Count', data=country_order_counts, palette='viridis', hue='Country', legend=False)
plt.title('Distribution of Orders Across Countries')
plt.xlabel('Country')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Most of the orders are coming from the USA followed Spain, France and Australia in that order

In [55]:

```
#Generate a table showing how the orders are distibuted across counties in percantages
country_order_counts['Percentage'] = (country_order_counts['Order_Count'] / country_order_counts['Order_Count'].sum()
display(country_order_counts.sort_values(by='Percentage', ascending=False))
```

	Country	Order_Count	Percentage
0	USA	928	33.782308
1	Spain	342	12.449945
2	France	314	11.430652
3	Australia	185	6.734620
4	UK	144	5.242082
5	Italy	113	4.113578
6	Finland	92	3.349108
7	Norway	85	3.094285
8	Singapore	79	2.875865
9	Canada	70	2.548234
10	Denmark	63	2.293411
11	Germany	62	2.257008
12	Sweden	57	2.074991
13	Austria	55	2.002184
14	Japan	52	1.892974
15	Belgium	33	1.201311
16	Switzerland	31	1.128504
17	Philippines	26	0.946487
18	Ireland	16	0.582454

Ordesr from the USA constitute of 33.78% of the orders placed. Ireland has the fewest orders constituting of 0.58%

In [56]:

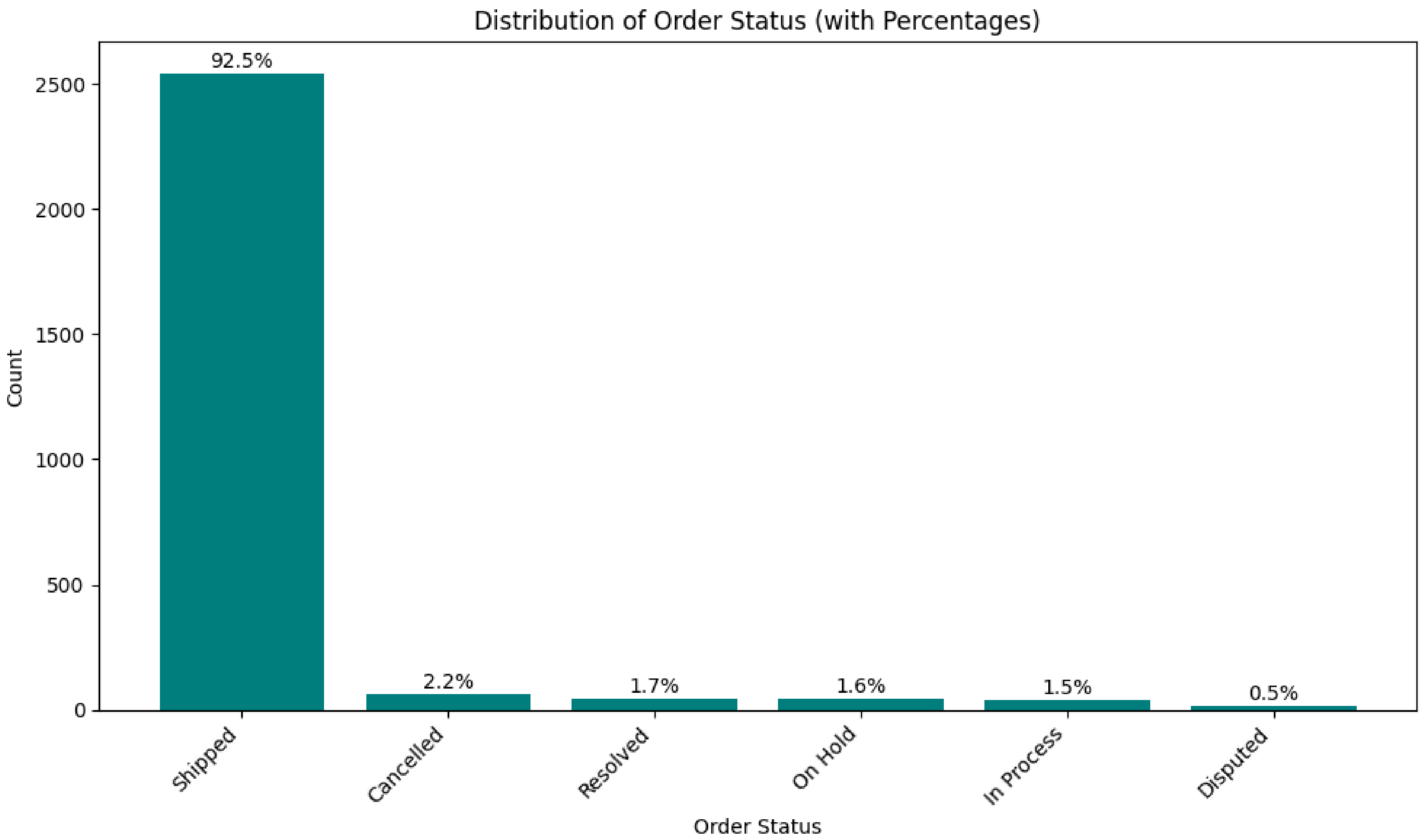
```
#show the value counts for unique values under 'STATUS', 'PRODUCTLINE' and 'DEALSIZE'
display(auto_df['STATUS'].value_counts())
```

	count
STATUS	
Shipped	2541
Cancelled	60
Resolved	47
On Hold	44
In Process	41
Disputed	14

dtype: int64

In [57]:

```python
#generate a chart showing the percentage of each status
status_counts = auto_df['STATUS'].value_counts().reset_index()
status_counts.columns = ['STATUS', 'Count']

# Calculate percentages
total_count = status_counts['Count'].sum()
status_counts['Percentage'] = (status_counts['Count'] / total_count) * 100

plt.figure(figsize=(10, 6))
bars = plt.bar(status_counts['STATUS'], status_counts['Count'], color='teal')

# Set labels and title
plt.title('Distribution of Order Status (with Percentages)')
plt.xlabel('Order Status')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right') # Rotate status names for better readability

# Annotate with Percentages
for i, bar in enumerate(bars):
    height = bar.get_height()
    percentage_value = status_counts['Percentage'].iloc[i]

    plt.text(bar.get_x() + bar.get_width() / 2.,
             height + (total_count * 0.002), # Position text slightly above the bar, scaled to total
             f'{percentage_value:.1f}%',
             ha='center',
             va='bottom',
             fontsize=10)

plt.tight_layout()
plt.savefig('status_bar_chart_with_percentages.png')
plt.show()
```



the shipped orders consist of 92,5 percent of all the orders

In [58]:

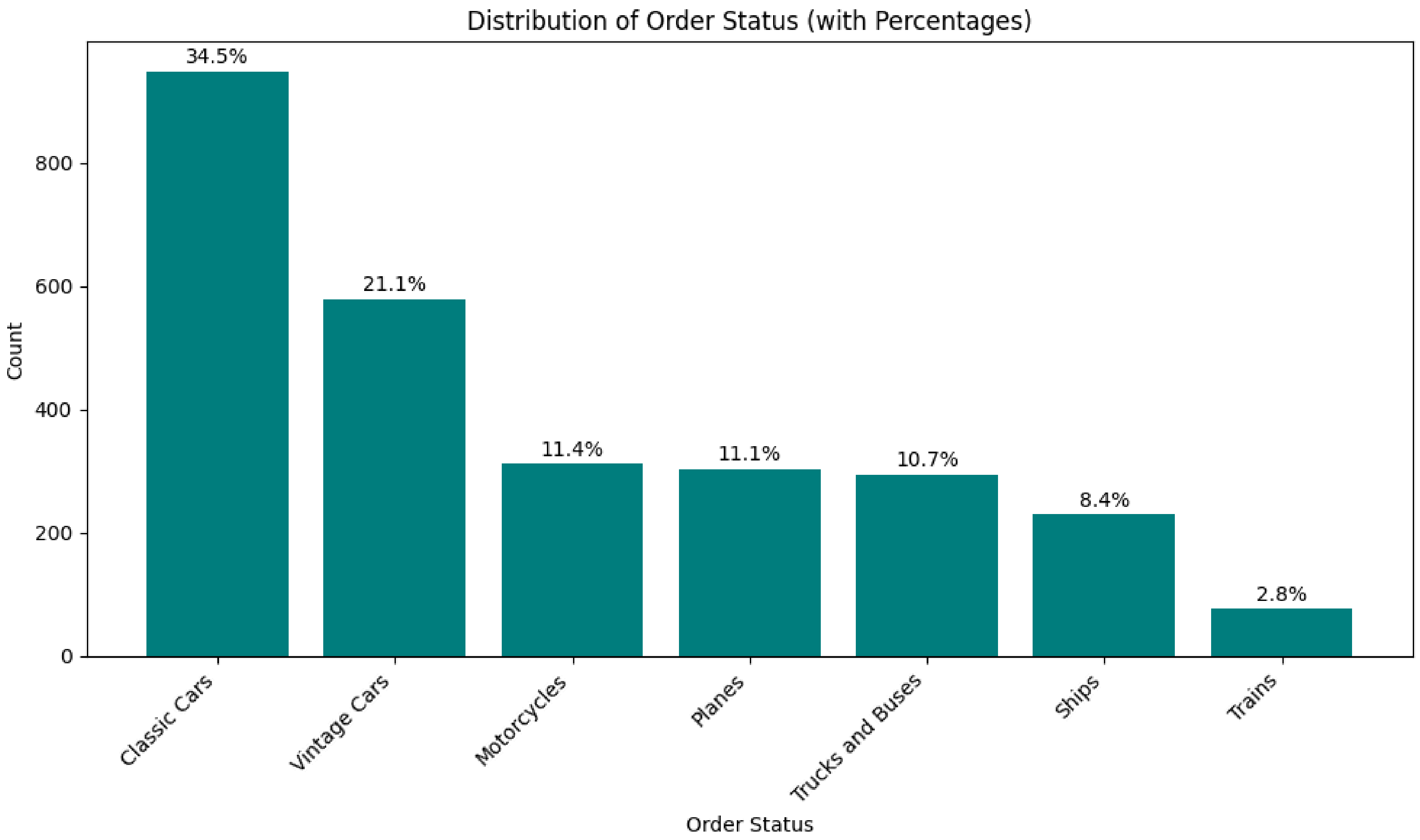
```
display(auto_df['PRODUCTLINE'].value_counts())
```

PRODUCTLINE	count
Classic Cars	949
Vintage Cars	579
Motorcycles	313
Planes	304
Trucks and Buses	295
Ships	230
Trains	77

dtype: int64

In [59]:

```
#generate a chart showing the percentage of each product line
status_counts = auto_df['PRODUCTLINE'].value_counts().reset_index()
status_counts.columns = ['PRODUCTLINE', 'Count']

# Calculate percentages
total_count = status_counts['Count'].sum()
status_counts['Percentage'] = (status_counts['Count'] / total_count) * 100

plt.figure(figsize=(10, 6))
bars = plt.bar(status_counts['PRODUCTLINE'], status_counts['Count'], color='teal')

# Set labels and title
plt.title('Distribution of Order Status (with Percentages)')
plt.xlabel('Order Status')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right') # Rotate status names for better readability

# Annotate with Percentages
for i, bar in enumerate(bars):
    height = bar.get_height()
    percentage_value = status_counts['Percentage'].iloc[i]

    plt.text(bar.get_x() + bar.get_width() / 2.,
             height + (total_count * 0.002), # Position text slightly above the bar, scaled to total
             f'{percentage_value:.1f}%',
             ha='center',
             va='bottom',
             fontsize=10)

plt.tight_layout()
plt.savefig('status_bar_chart_with_percentages.png')
plt.show()
```



Classic cars constitute of 34.5%, Vintage are 21.1% and Motocycles are 11.4 percent. The productline with least products is Trains at 2.8%.

```
In [60]: display(auto_df['DEALSIZE'].value_counts())
```

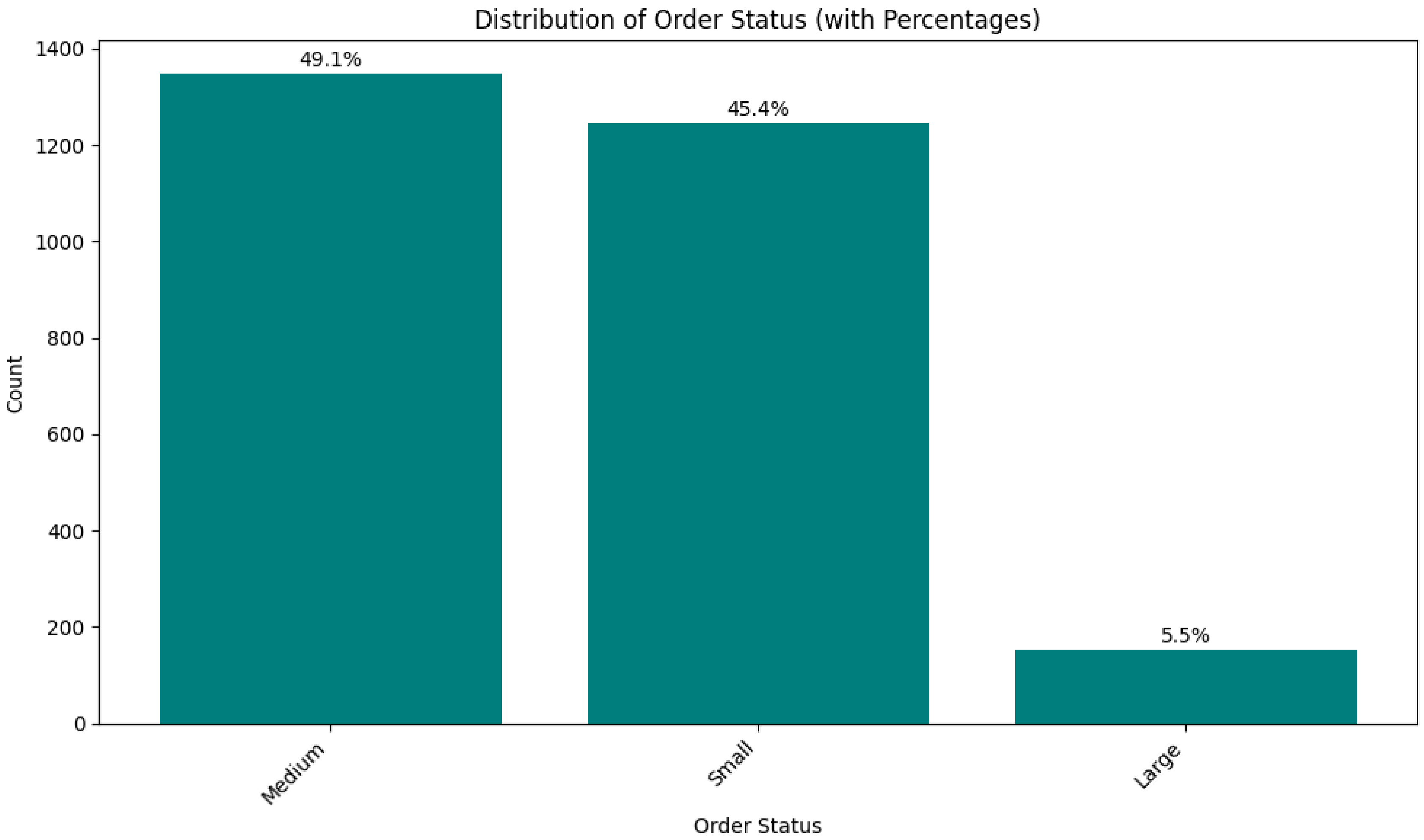
	count
DEALSIZE	
Medium	1349
Small	1246
Large	152

dtype: int64

In [61]:

```
#generate a chart showing the percentage of each DEALSIZE
status_counts = auto_df['DEALSIZE'].value_counts().reset_index()
status_counts.columns = ['DEALSIZE', 'Count']

# Calculate percentages
total_count = status_counts['Count'].sum()
status_counts['Percentage'] = (status_counts['Count'] / total_count) * 100

plt.figure(figsize=(10, 6))
bars = plt.bar(status_counts['DEALSIZE'], status_counts['Count'], color='teal')

# Set labels and title
plt.title('Distribution of Order Status (with Percentages)')
plt.xlabel('Order Status')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right') # Rotate status names for better readability

# Annotate with Percentages
for i, bar in enumerate(bars):
    height = bar.get_height()
    percentage_value = status_counts['Percentage'].iloc[i]

    plt.text(bar.get_x() + bar.get_width() / 2.,
             height + (total_count * 0.002), # Position text slightly above the bar, scaled to total
             f'{percentage_value:.1f}%',
             ha='center',
             va='bottom',
             fontsize=10)
```

```
plt.tight_layout()
plt.savefig('status_bar_chart_with_percentages.png')
plt.show()
```



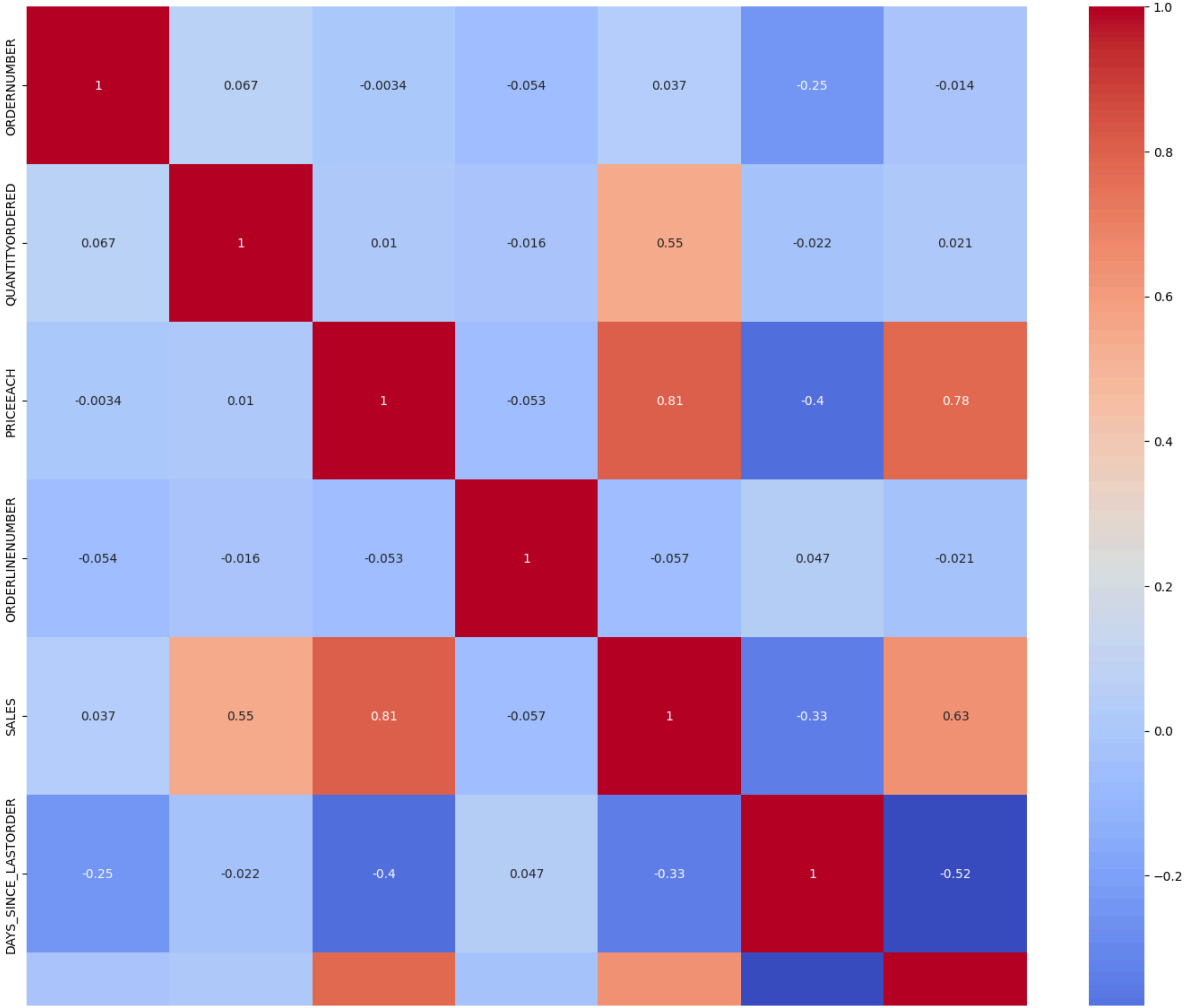
The Large orders are very few constituting of 5.5% while Medium and Small constitute 49.1% and 45.4% respectively.

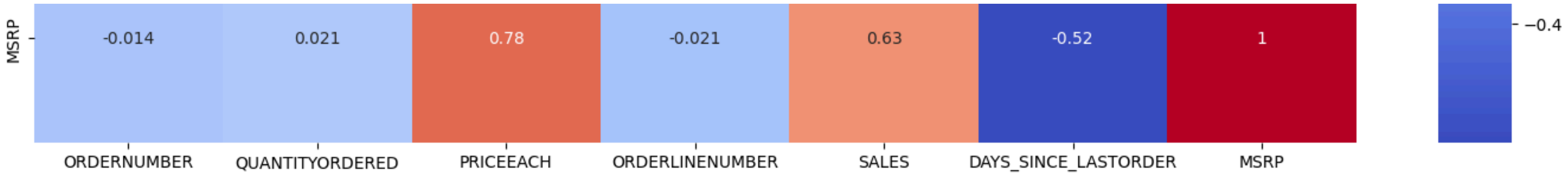
In [62]:
```
#convert the ORDERDATE to a date variable
auto_df['ORDERDATE'] = pd.to_datetime(auto_df['ORDERDATE'])
auto_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2747 entries, 0 to 2746
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   ORDERNUMBER          2747 non-null   int64         
 1   QUANTITYORDERED      2747 non-null   int64         
 2   PRICEEACH            2747 non-null   float64       
 3   ORDERLINENUMBER      2747 non-null   int64         
 4   SALES                2747 non-null   float64       
 5   ORDERDATE            2747 non-null   datetime64[ns]
 6   DAYS_SINCE_LASTORDER 2747 non-null   int64         
 7   STATUS               2747 non-null   object        
 8   PRODUCTLINE          2747 non-null   object        
 9   MSRP                 2747 non-null   int64         
 10  PRODUCTCODE          2747 non-null   object        
 11  CUSTOMERNAME         2747 non-null   object        
 12  COUNTRY              2747 non-null   object        
 13  DEALSIZE             2747 non-null   object        
dtypes: datetime64[ns](1), float64(2), int64(5), object(6)
memory usage: 300.6+ KB
```

In [63]:

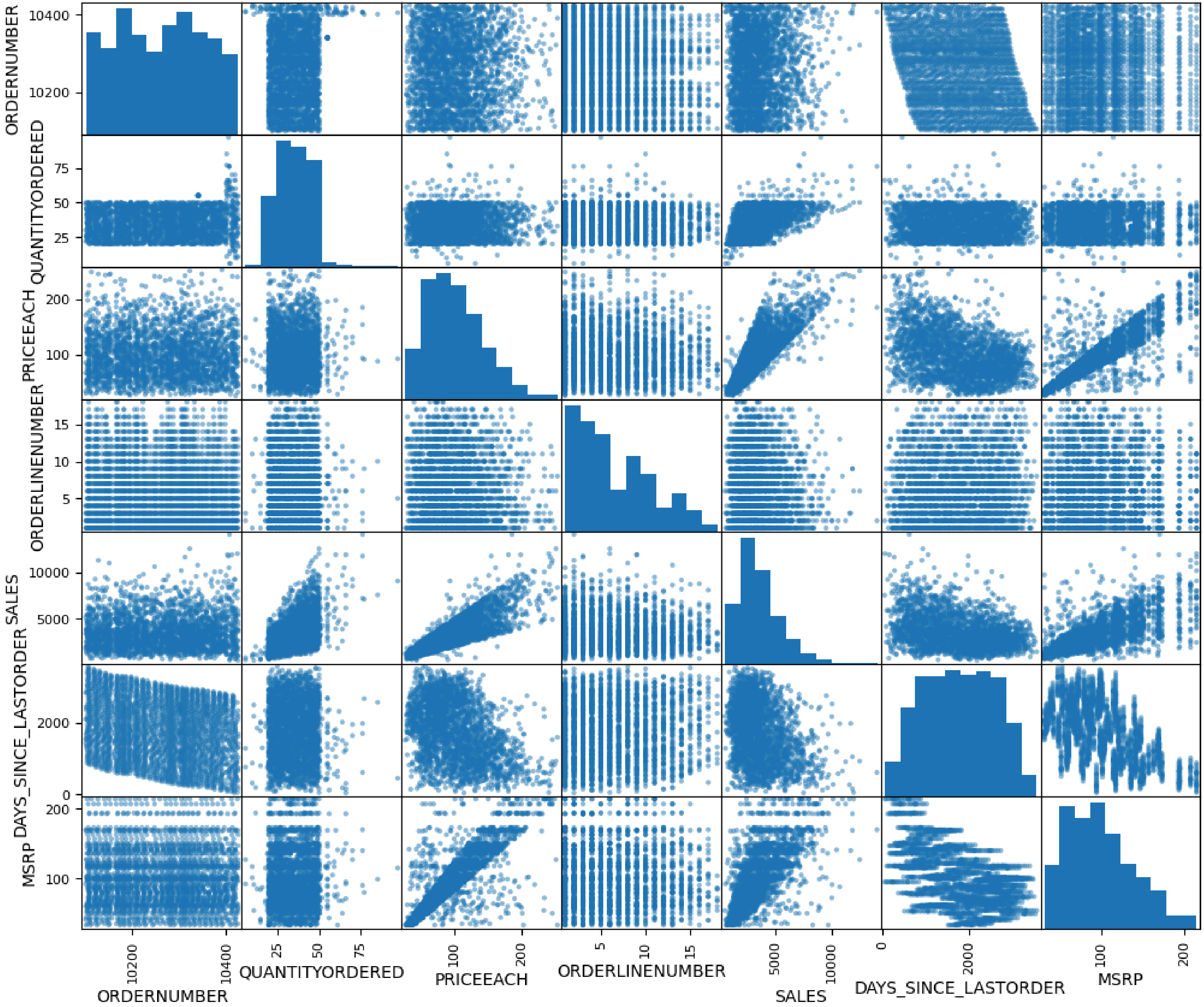
```
numeric_df = auto_df.select_dtypes(include=np.number)
corr = numeric_df.corr()

sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap='coolwarm')
plt.rcParams['figure.figsize'] = [12, 10]
plt.show()
```


ORDERNUMBER
1
0.067
-0.0034
-0.054
0.037
-0.25
-0.014
QUANTITYORDERED
0.067
1
0.01
-0.016
0.55
-0.022
0.021
PRICEEACH
-0.0034
0.01
1
-0.053
0.81
-0.4
0.78
ORDERLINENUMBER
-0.054
-0.016
-0.053
1
-0.057
0.047
-0.021
SALES
0.037
0.55
0.81
-0.057
1
-0.33
0.63
DAYS_SINCE_LASTORDER
-0.25
-0.022
-0.4
0.047
-0.33
1
-0.52
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



1. There is a fair correlation (0.81) between PRICEEACH and SALES as well as PRICEEACH and MSRP (0.78)
2. SALES and MSRP have a correlation of 0.63
3. SALES and QUANTITYORDERED have a correlation of 0.55.

In [64]:
```
#We generate a scatter matrix
scatter_matrix(auto_df)
plt.show()
```


ORDERNUMBER
10400
10200
QUANTITYORDERED
75
50
25
PRICEEACH
200
100
ORDERLINENUMBER
15
10
5
SALES
10000
5000
DAYS_SINCE_LASTORDER
2000
0
MSRP
200
100
10200
10400
ORDERNUMBER
25
50
75
QUANTITYORDERED
100
200
PRICEEACH
5
10
15
ORDERLINENUMBER
5000
10000
SALES
0
2000
DAYS_SINCE_LASTORDER
100
200
MSRP

```
In [64]:
```