Lab1: Exploratory Data Analysis

Domain: Finance and marketing

23/01/2020

In [90]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Collecting data

The provided data contains purchase information of a fictional store named Steel Wheels (available at https://www.kaggle.com/kyanyoga/sample-sales-data (https://www.kaggle.com/kyanyoga/sample-sales-data)). Each line refers to a specific product that was sold to a customer in a specific order.

- ORDERNUMBER: Unique ID for each order made by a customer
- ORDERDATE: Date of the order
- PRODUCTLINE: Type of product
- QUANTITYORDERED: Quantity of the product item included in the considered order specified by ORDERNUMBER
- SALES: Total price of the product item included in the considered order
- CUSTOMERNAME: Name of the customer of the considered order

In [113]:
```
df=pd.read_excel('C:\\Users\\madhu\\Desktop\\sales_data_sample.xlsx')
```

In [51]:
```
#Shape of the dataset
df.shape
```

Out[51]: (2823, 25)

In [52]:
```
df.head()
```

Out[52]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	A
0	10107	30	95.70	2	2871.00	2003-02-24	Shipped	1	2	2003	...	
1	10121	34	81.35	5	2765.90	2003-05-07	Shipped	2	5	2003	...	
2	10134	41	94.74	2	3884.34	2003-07-01	Shipped	3	7	2003	...	
3	10145	45	83.26	6	3746.70	2003-08-25	Shipped	3	8	2003	...	
4	10159	49	100.00	14	5205.27	2003-10-10	Shipped	4	10	2003	...	7

5 rows × 25 columns

Data Cleaning and Data Exploratory Analysis

In [114]:
```
to_drop = ['PHONE','ADDRESSLINE1','ADDRESSLINE2','STATE','POSTALCODE','TERRITORY']
df = df.drop(to_drop, axis=1)
```

Here, we remove the variables which dont add significant value for the analysis-
'PHONE','ADDRESSLINE1','ADDRESSLINE2','STATE','POSTALCODE','TERRITORY'

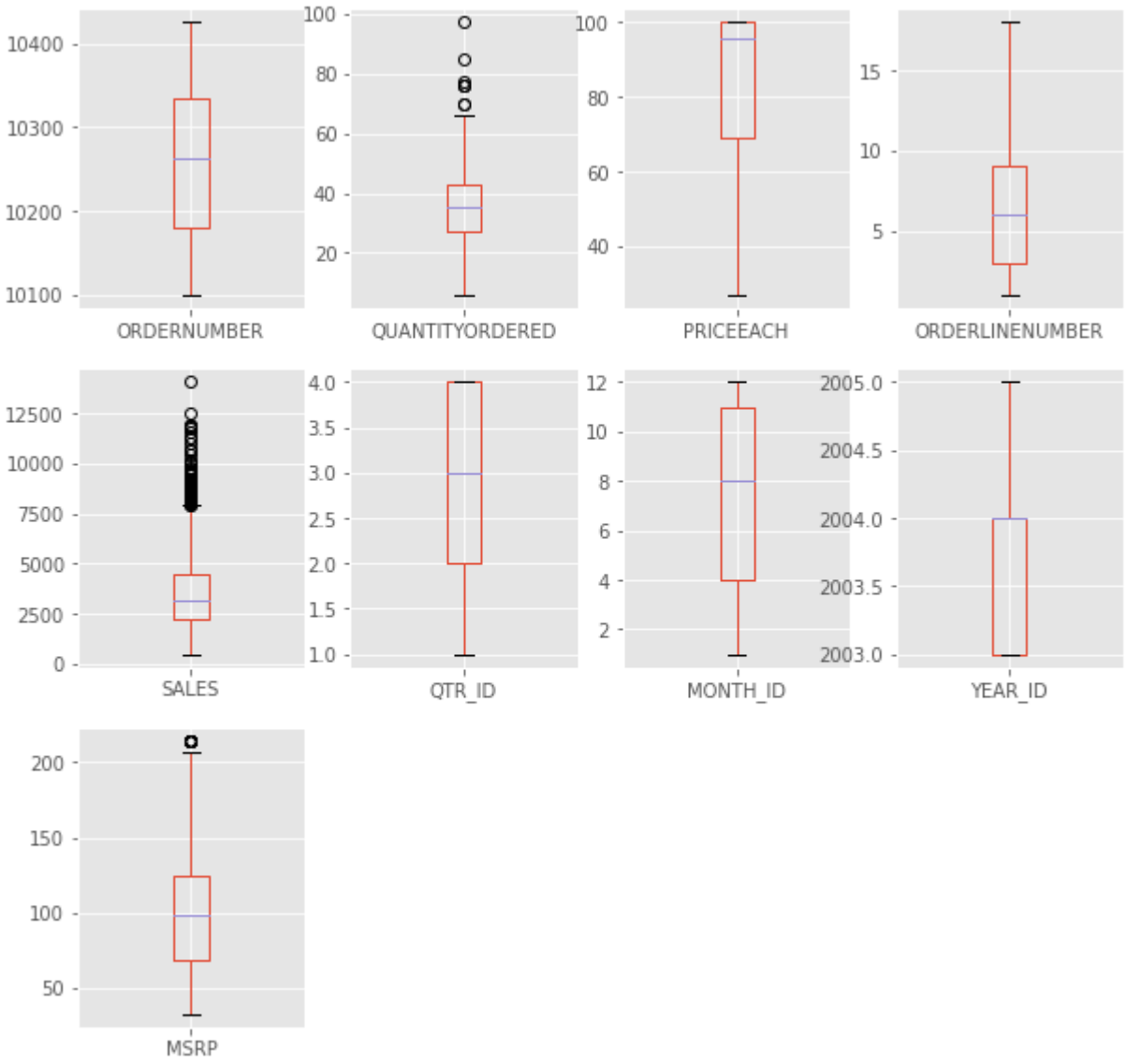
In [54]:
```
#Finding missing values
df.isnull().sum()
```

Out[54]:
```
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
CITY                0
COUNTRY             0
CONTACTLASTNAME     0
CONTACTFIRSTNAME    0
DEALSIZE            0
dtype: int64
```

The above dataframe shows that the data does not have any missing values and need no data imputation.

Outlier detection and removal

In [57]:
```
df.plot(kind='box', subplots=True, sharex=False, sharey=False, figsize=(10,10), layout=(3,4))
plt.show()
```

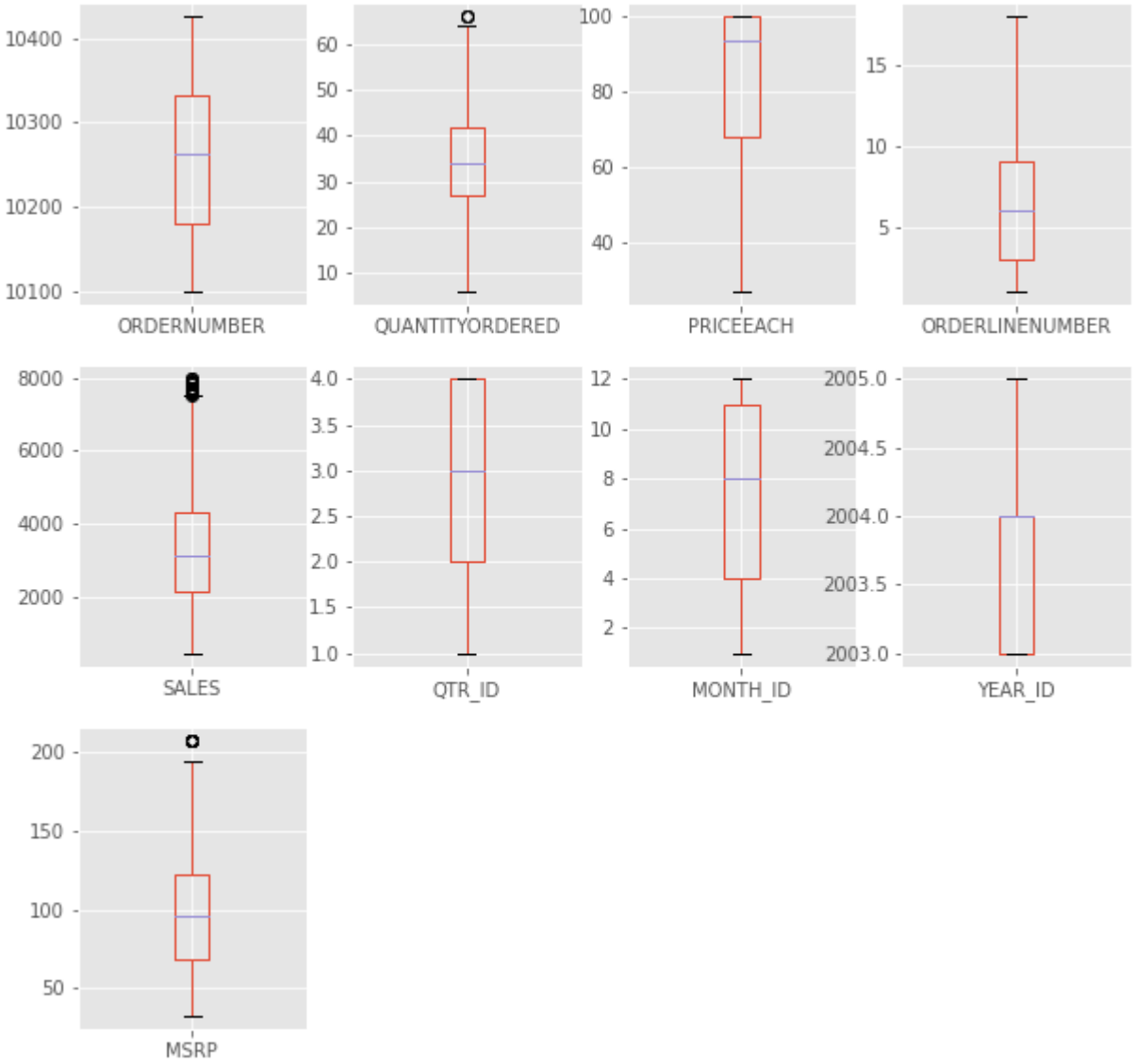


Itcan be seen that 'MSRP' has a single outlier, while the 'SALES' and 'QUANTITYORDERED' has the most number of outliers.Most of the outliers can be filtered using the filtering criteria of a boxplot to classify the point as an outlier. The criteria of a box plot for classifying a point as an outlier is if the point is greater than Q3+(1.5∗IQR) or lower than Q1−(1.5∗IQR) where, where Q1=FirstQuartile Q3=ThirdQuartile.

In [116]:
```
print('original shape of dataset :',df.shape)
cols = ['SALES', 'MSRP','QUANTITYORDERED']
new_df = df[cols]

#criteria
Q1 = new_df.quantile(0.25)
Q3 = new_df.quantile(0.75)
IQR = Q3-Q1
max_ = Q3+1.5*IQR
min_ = Q1-1.5*IQR
#filter the outlier s
condition = (new_df <= max_) & (new_df >= min_)
condition = condition.all(axis=1)
df = df[condition]
print('filtered dataset shape : ',df.shape)
df.plot(kind='box', subplots=True, sharex=False, sharey=False, figsize=(10,10), layout=(3,4))
plt.show()
```

```
original shape of dataset : (2823, 19)
filtered dataset shape :  (2719, 19)
```


10400
10300
10200
10100
ORDERNUMBER
60
50
40
30
20
10
QUANTITYORDERED
100
80
60
40
PRICEEACH
15
10
5
ORDERLINENUMBER
8000
6000
4000
2000
SALES
4.0
3.5
3.0
2.5
2.0
1.5
1.0
QTR_ID
12
10
8
6
4
2
MONTH_ID
2005.0
2004.5
2004.0
2003.5
2003.0
YEAR_ID
200
150
100
50
MSRP

After the outlier filtering, only a few outliers are existing. Therefore, we move on with the analysis.

In [59]:
```
#Checking for inconsistent data types
df.dtypes
```

Out[59]:
```
ORDERNUMBER                int64
QUANTITYORDERED            int64
PRICEEACH                float64
ORDERLINENUMBER            int64
SALES                    float64
ORDERDATE          datetime64[ns]
STATUS                     object
QTR_ID                     int64
MONTH_ID                   int64
YEAR_ID                    int64
PRODUCTLINE                object
MSRP                       int64
PRODUCTCODE                object
CUSTOMERNAME               object
CITY                       object
COUNTRY                    object
CONTACTLASTNAME            object
CONTACTFIRSTNAME           object
DEALSIZE                   object
dtype: object
```

In [60]:
```
#Changing the data type of variable 'ORDERDATE' from object to datetime
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
```

Summary stats of Quantitative variables

In [61]:
```
quant_vars = ['QUANTITYORDERED','PRICEEACH','SALES','MSRP']
df[quant_vars].describe()
```

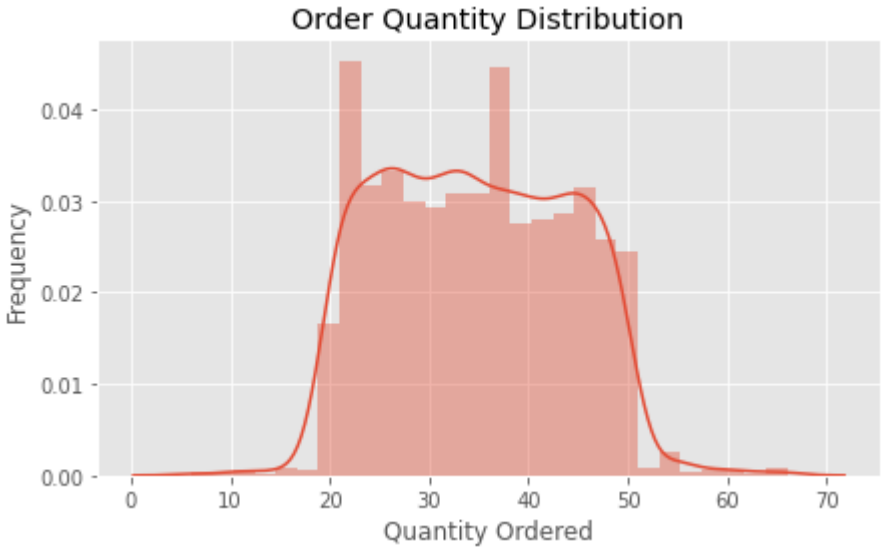
Out[61]:

	QUANTITYORDERED	PRICEEACH	SALES	MSRP
count	2719.000000	2719.000000	2719.000000	2719.000000
mean	34.665318	83.045068	3367.023913	98.107392
std	9.325080	20.304206	1560.878732	37.456615
min	6.000000	26.880000	482.130000	33.000000
25%	27.000000	67.820000	2169.540000	68.000000
50%	34.000000	93.560000	3127.880000	96.000000
75%	42.000000	100.000000	4306.400000	122.000000
max	66.000000	100.000000	7962.240000	207.000000

It can be seen that the quantitative variables consists of no negative values, which is a good indicator as the above mentioned variables cannot have negative values.

Order Quantity Distribution

In [62]:
```
plt.figure(figsize=(7,4))
sns.distplot(df['QUANTITYORDERED'])
plt.title('Order Quantity Distribution')
plt.xlabel('Quantity Ordered')
plt.ylabel('Frequency')
plt.show()
```



Most of the orders' quantity lie between 20 and 40 units. Therefore, majority of them are bulk orders.

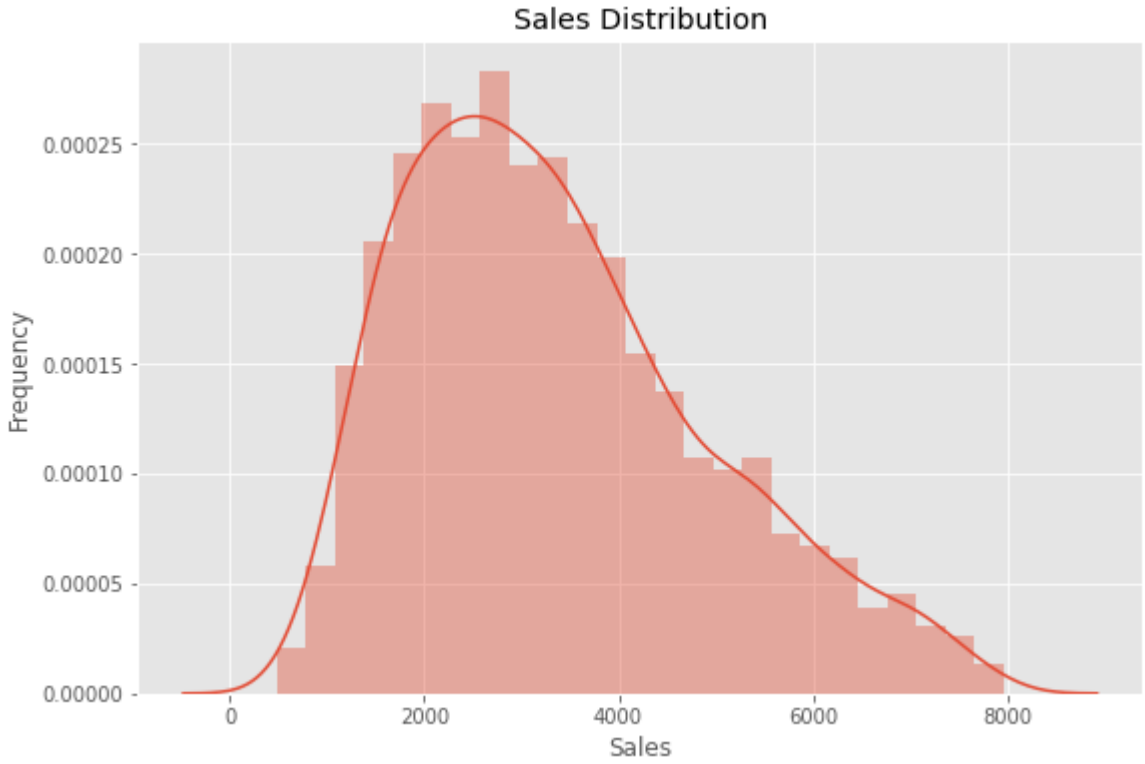
Price Distribution

In [63]:
```
plt.figure(figsize=(7,4))
sns.distplot(df['PRICEEACH'])
plt.title('Price Distribution')
plt.xlabel('Price Ordered')
plt.ylabel('Frequency')
plt.show()
```

The distribution is left skewed. Meaning, most of the orders placed are of quantities costing between $90 and 100$ per item. Also, the maximum price per item in the records is 100$.

Sales Distribution

In [64]:
```
plt.figure(figsize=(9,6))
sns.distplot(df['SALES'])
plt.title('Sales Distribution')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()
```



The distribution is right skewed.Majority of the customer sales are below the mean sales of the time period considered.

Analyzing 'STATUS' of the orders

In [65]:
```
df['STATUS'].value_counts()
```

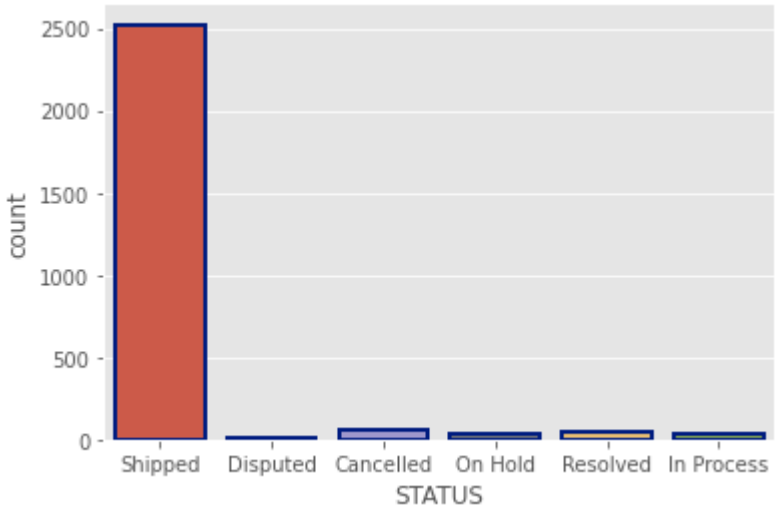
Out[65]:
```
Shipped       2523
Cancelled       60
Resolved        46
On Hold         40
In Process      39
Disputed        11
Name: STATUS, dtype: int64
```

In [66]:
```
df['STATUS'].value_counts(normalize = True)
```

Out[66]:
```
Shipped       0.927915
Cancelled     0.022067
Resolved      0.016918
On Hold       0.014711
In Process    0.014344
Disputed      0.004046
Name: STATUS, dtype: float64
```

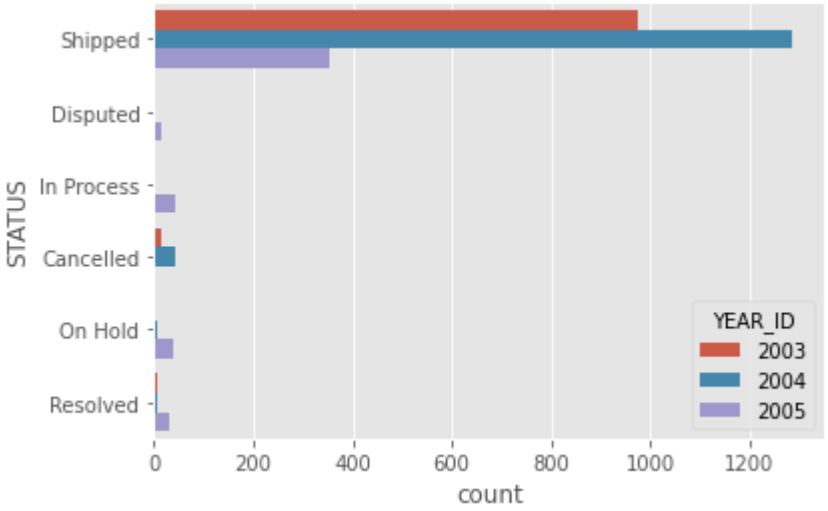
In [67]:
```
sns.countplot(x="STATUS",data=df,linewidth=2,edgecolor=sns.color_palette("dark", 1))
```

Out[67]: <AxesSubplot:xlabel='STATUS', ylabel='count'>



In [99]:
```
sns.countplot(y='STATUS',data=data,hue='YEAR_ID')
```

Out[99]: <AxesSubplot:xlabel='count', ylabel='STATUS'>



A huge majority of the orders/products have been successfully shipped. While, there was Dispute, In process and Onhold status in

2005. This highlights the performance of the company shipping in 2005.

In [68]:
```
df.groupby(['YEAR_ID'])['MONTH_ID'].nunique()
```

Out[68]:
```
YEAR_ID
2003    12
2004    12
2005     5
Name: MONTH_ID, dtype: int64
```

The above output highlights that the data is incomplete for the year 2005.

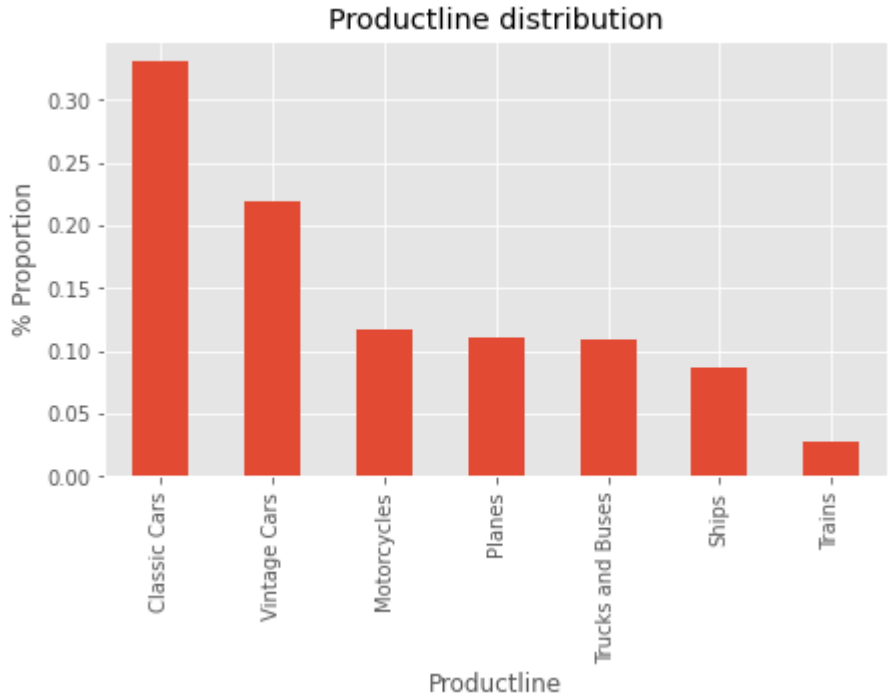
Dealsize Distribution

In [110]:
```
plt.figure(figsize=(7,4))
df['DEALSIZE'].value_counts(normalize = True).plot(kind = 'bar')
plt.title('DealSize distribution')
plt.xlabel('Deal Size')
plt.ylabel('% Proportion')
plt.show()
```

Majority of the deal size taken from the customers is found out to be 'Medium'.

Productline distribution

In [118]:
```
plt.figure(figsize=(7,4))
df['PRODUCTLINE'].value_counts(normalize = True).plot(kind = 'bar')
plt.title('Productline distribution')
plt.xlabel('Productline')
plt.ylabel('% Proportion')
plt.show()
```



The 'Classic Cars' productline has the most number of purchase recorded in the considered time period. This plot talks about demand and clearly, the 'Classic Cars' has the most demand among the other services that the company provides.

In [105]:
```
D=df.groupby(['YEAR_ID','QTR_ID'])[['SALES']].sum()
D
```

Out[105]:

		SALES
YEAR_ID	**QTR_ID**	
2003	**1**	415533.60
	2	509859.90
	3	579726.98
	4	1748525.73
2004	**1**	768195.48
	2	725356.31
	3	1028015.59
	4	1870261.67
2005	**1**	983847.24
	2	525615.52

The above output depicts the year and further quarter-wise sales.

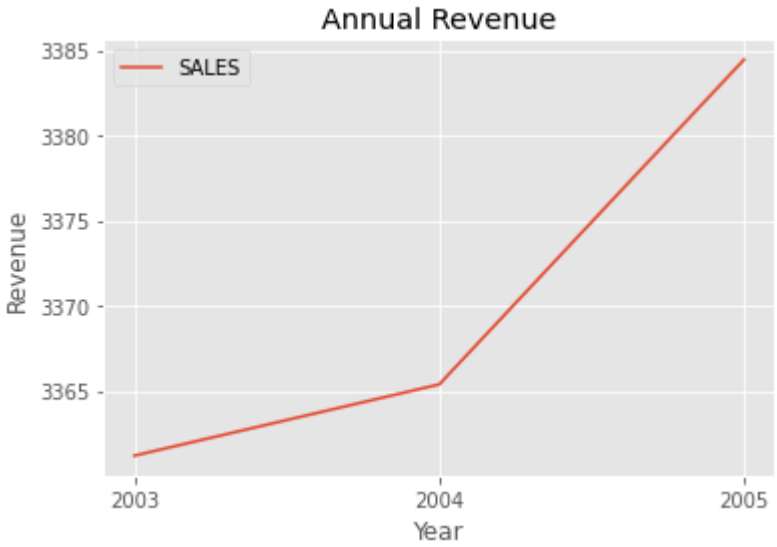
In [130]:
```
df.groupby(['YEAR_ID'])[['SALES']].mean()
```

Out[130]:

	SALES
YEAR_ID	
2003	3361.204762
2004	3365.386245
2005	3384.445650

In [128]:
```
plt.figure(figsize=(9,6))
df.groupby(['YEAR_ID'])[['SALES']].mean().plot()
plt.xlabel('Year')
plt.ylabel('Revenue')
plt.title('Annual Revenue')
plt.xticks(np.arange(2003,2006,1))
plt.show()
```

```
<Figure size 648x432 with 0 Axes>
```



The above graph is misleading. The reason being the observations for the year 2005 is not complete, resulting in less number of minthly data. So, this affects the average wrongly and hence the misleading output.

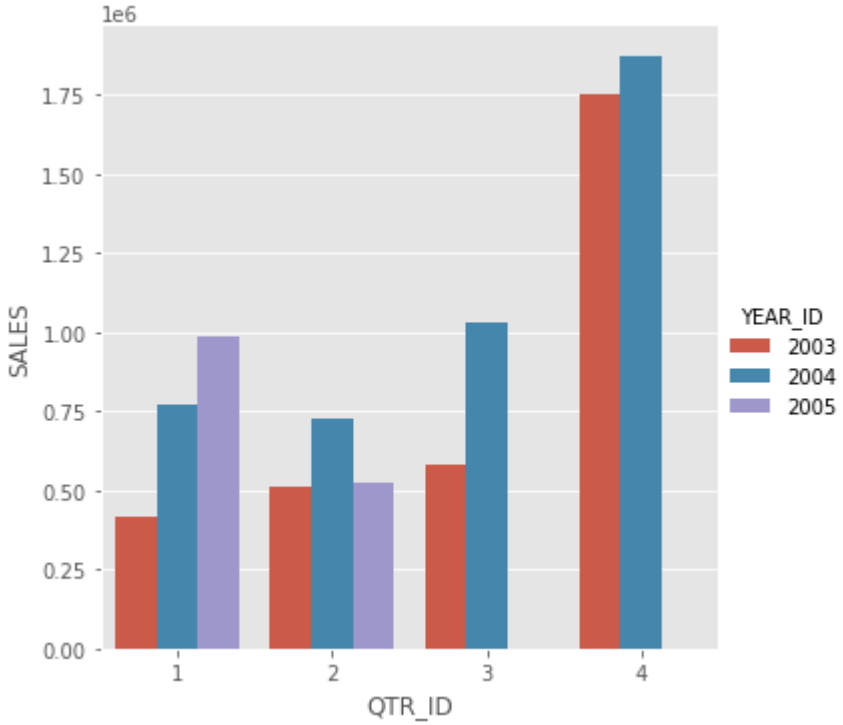
In [119]:
```
D.reset_index(inplace=True)
D.head()
```

Out[119]:

	level_0	index	YEAR_ID	QTR_ID	SALES
0	0	0	2003	1	415533.60
1	1	1	2003	2	509859.90
2	2	2	2003	3	579726.98
3	3	3	2003	4	1748525.73
4	4	4	2004	1	768195.48

In [108]:
```
sns.factorplot(y='SALES', x='QTR_ID',data=D,kind="bar" ,hue='YEAR_ID')
```

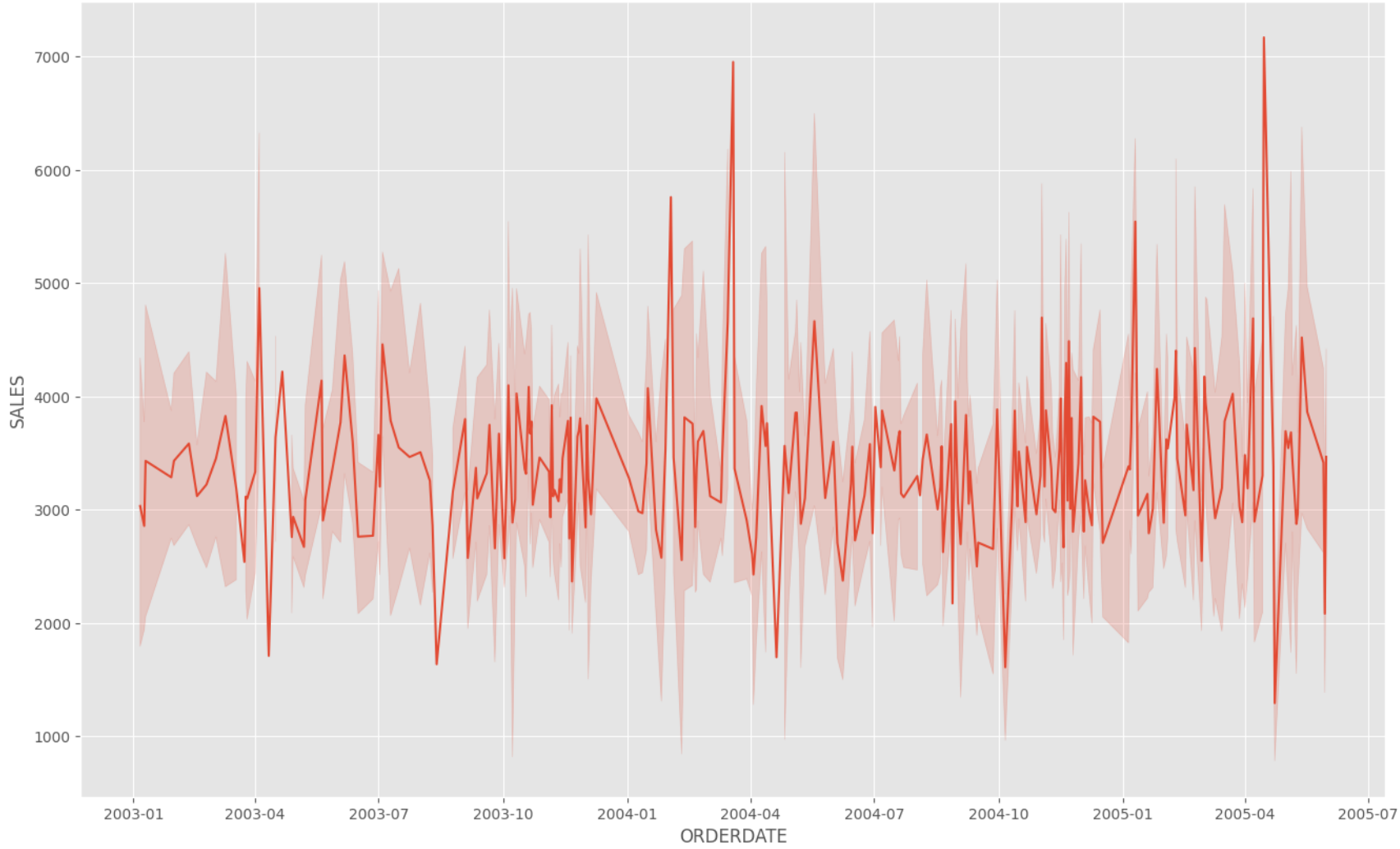
Out[108]:
```
<seaborn.axisgrid.FacetGrid at 0x21448072b70>
```



An interesting trend in the sales observed is that the sales is at its highest during the 4th quarter when compared to that of the first three quarters. The reasons could range from discount at the year end practice to the festive seasons that usually affects the sales.

In [93]:
```
plt.figure(figsize=(16,10), facecolor= 'w', dpi=100)
sns.lineplot(x='ORDERDATE', y='SALES', data=df)
```

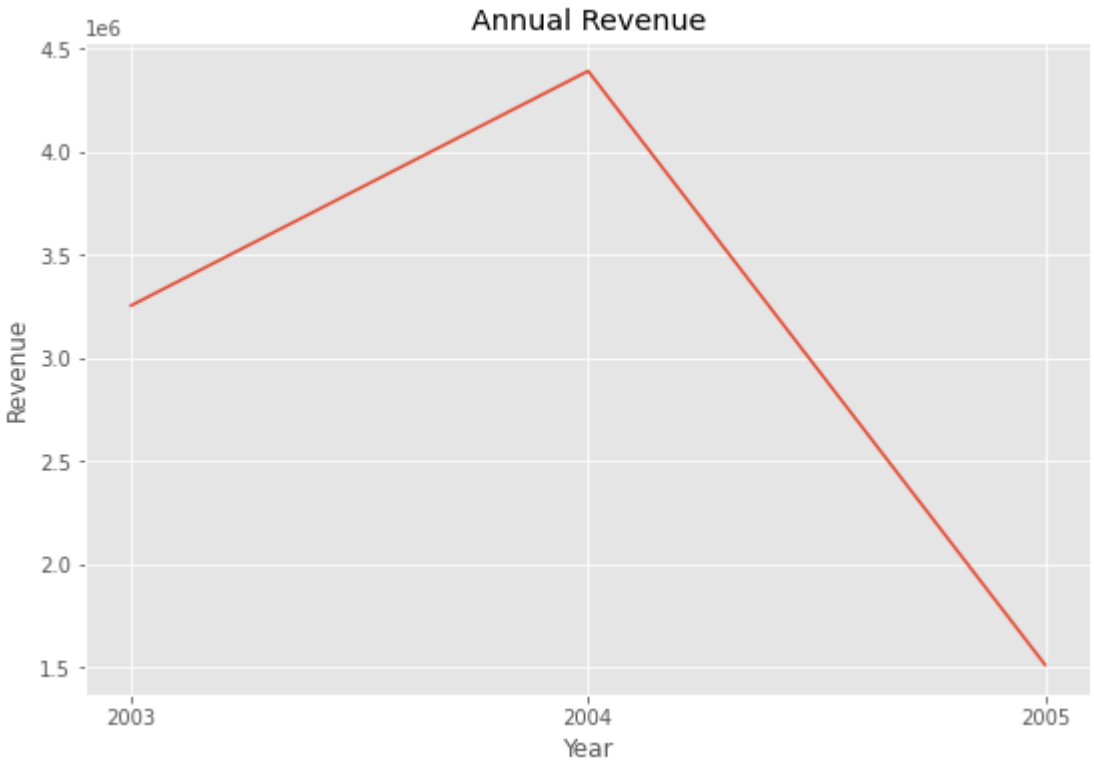
Out[93]: <AxesSubplot:xlabel='ORDERDATE', ylabel='SALES'>



From the lineplot, it can be said that the 'Sales' fluctuates throughout the years considered.

Annual Revenue

In [73]:
```
#Annual Revenue
plt.figure(figsize=(9,6))
df.groupby(['YEAR_ID'])['SALES'].sum().plot()
plt.xlabel('Year')
plt.ylabel('Revenue')
plt.title('Annual Revenue')
plt.xticks(np.arange(2003,2006,1))
plt.show()
```

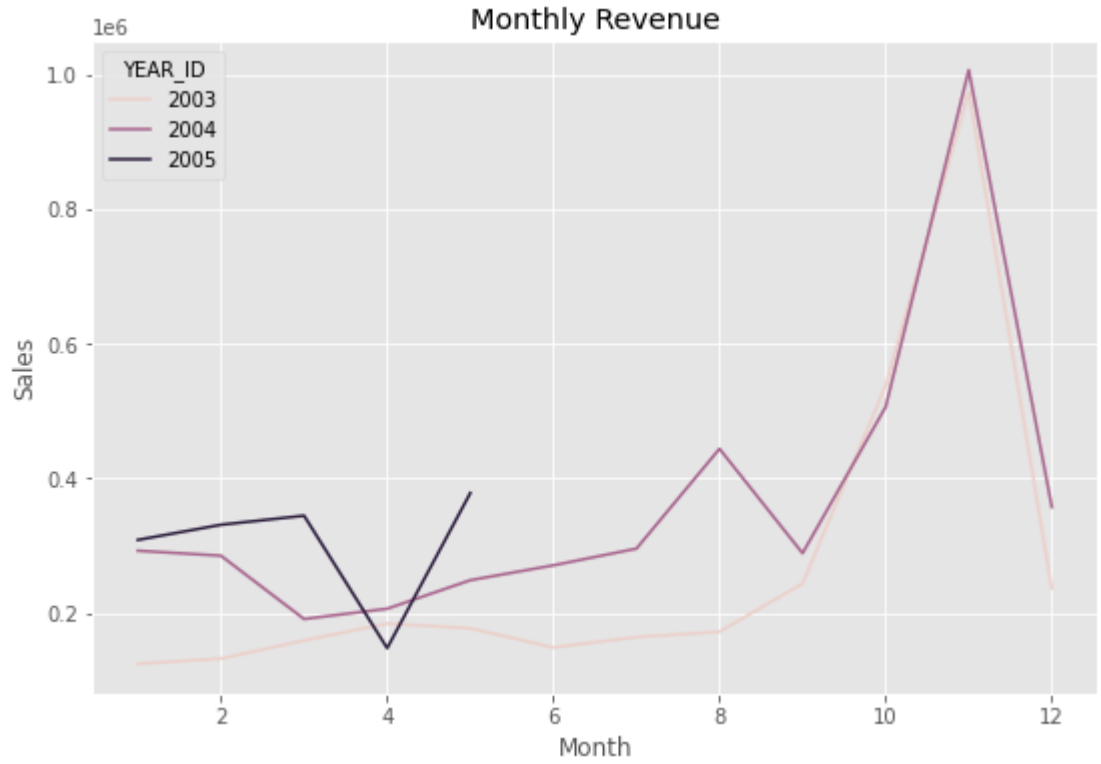


The above graph is not reliable and possibly could be misleading as as we dont have the complete data for 2005. Analyzing Monthy Revenue could be a better option.

Monthly Revenue

In [134]:
```
#Monthly Revenue
plt.figure(figsize=(9,6))

monthly_revenue_= df.groupby(['YEAR_ID','MONTH_ID'])['SALES'].sum().reset_index()
monthly_revenue_
sns.lineplot(x="MONTH_ID", y="SALES",hue="YEAR_ID", data=monthly_revenue_)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Monthly Revenue')
plt.show()
```



The above procured plot depicts that the revenue is growing especially in the months of October and November. The reason could possibly be due to seasonality(result of festivals). Also, 2005 is performing better than the other years in terms of revenue reaching the

maximum sales in all the months that is from Jan to May. The reason behind this increase of sales in 2005 can be further examined in order to maintain high sales in future of the company .

Monthly Revenue Growth Rate

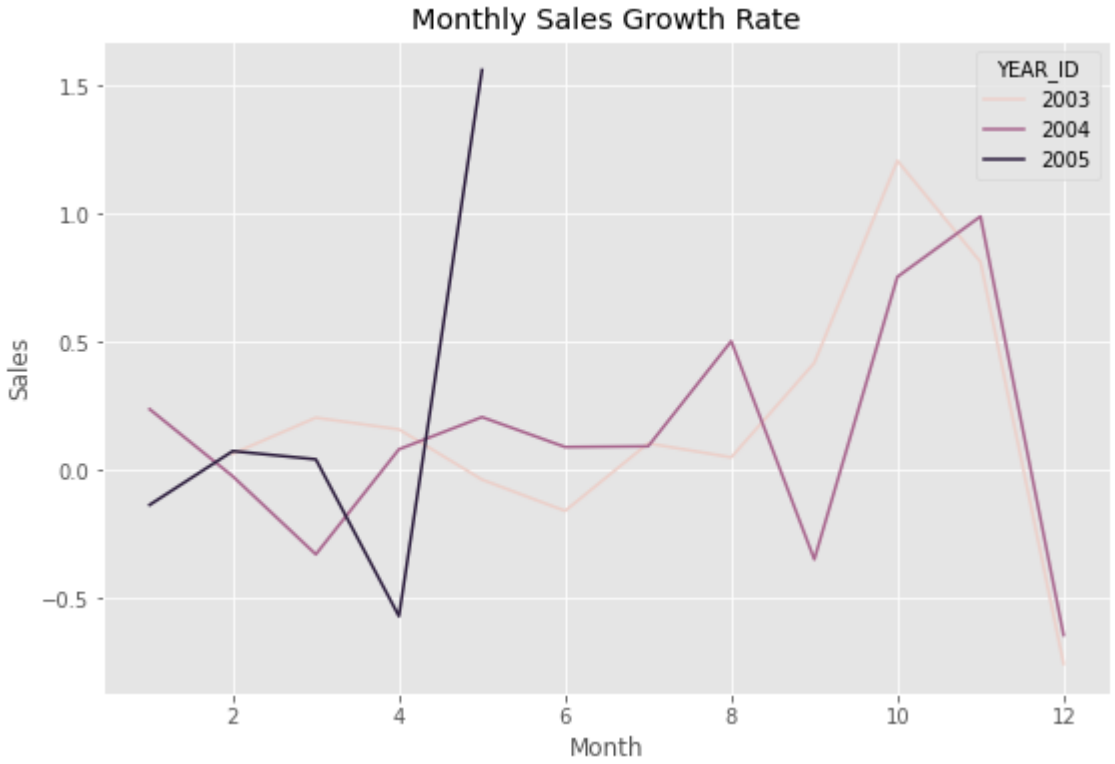
In [135]:
```
monthly_revenue_['MONTHLY_GROWTH'] = monthly_revenue['SALES'].pct_change()
```

In [136]:
```
monthly_revenue_.head()
```

Out[136]:

	YEAR_ID	MONTH_ID	SALES	MONTHLY_GROWTH
0	2003	1	124348.98	NaN
1	2003	2	132145.83	0.062701
2	2003	3	159038.79	0.203510
3	2003	4	184135.25	0.157801
4	2003	5	177060.05	-0.038424

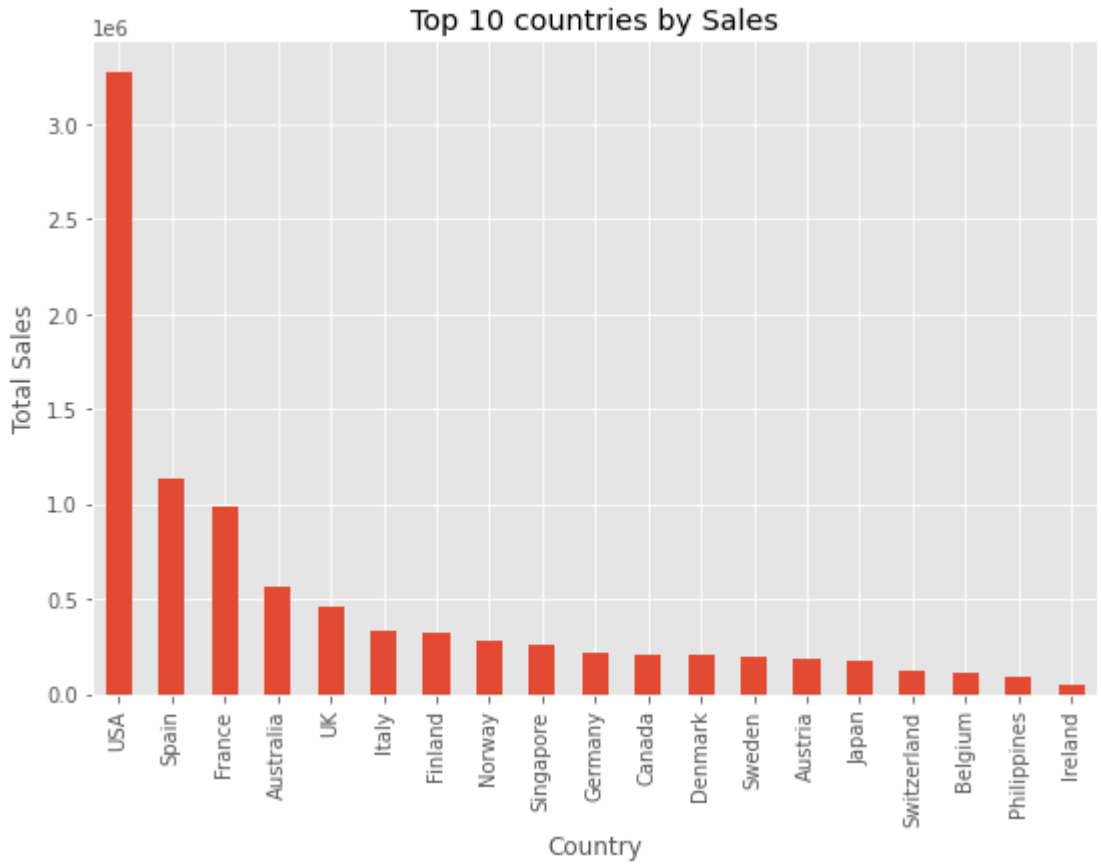
In [138]:
```
#Monthly Sales Growth Rate
plt.figure(figsize=(9,6))
sns.lineplot(x="MONTH_ID", y="MONTHLY_GROWTH",hue="YEAR_ID", data=monthly_revenue_)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Monthly Sales Growth Rate')
plt.show()
```



A high growth rate from Apr 2005 to May 2005 is seen from the above plot.

Top 10 countries by Sales

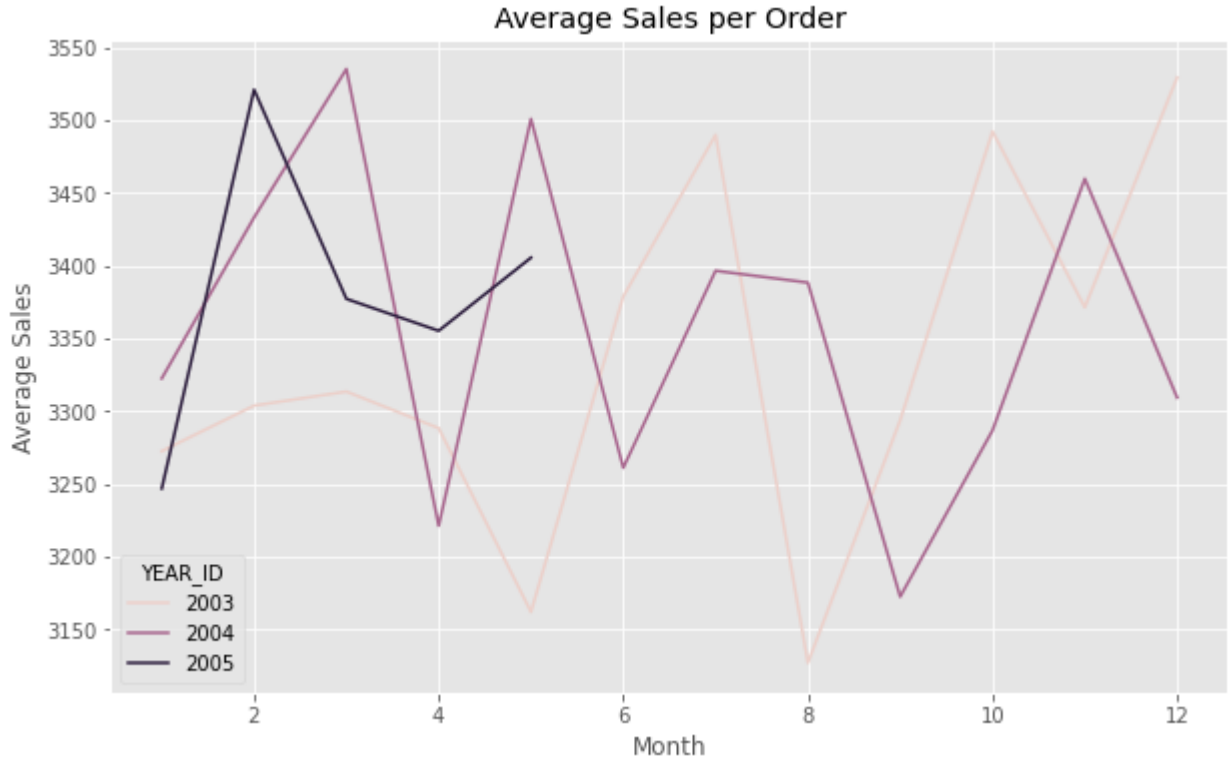
In [79]:
```
plt.figure(figsize=(9,6))
top_cities = df.groupby(['COUNTRY'])['SALES'].sum().sort_values(ascending=False)
top_cities.plot(kind = 'bar')
plt.title('Top 10 countries by Sales')
plt.xlabel('Country')
plt.ylabel('Total Sales')
plt.show()
```



USA has the highest sales recorded .

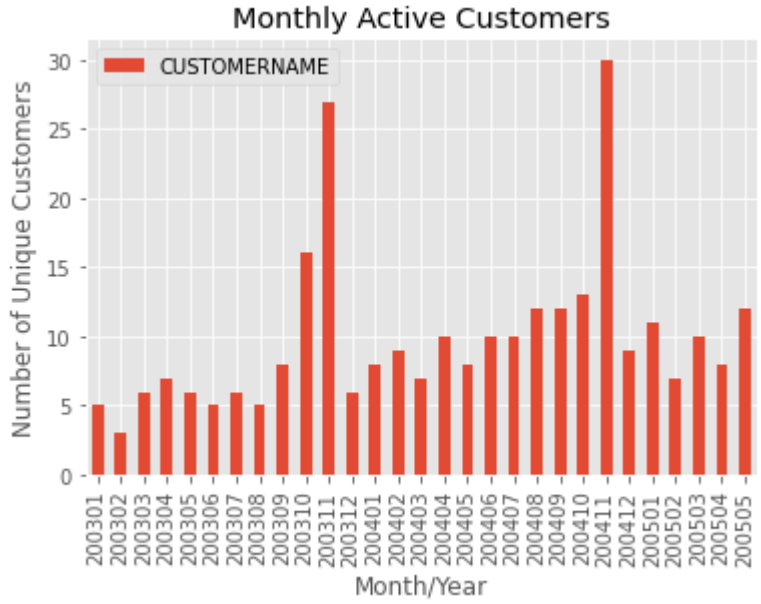
Average Sales per Order

In [80]:
```
#Average Sales per Order
average_revenue = df.groupby(['YEAR_ID','MONTH_ID'])['SALES'].mean().reset_index()
plt.figure(figsize=(10,6))
sns.lineplot(x="MONTH_ID", y="SALES",hue="YEAR_ID", data=average_revenue)
plt.xlabel('Month')
plt.ylabel('Average Sales')
plt.title('Average Sales per Order')
plt.show()
```



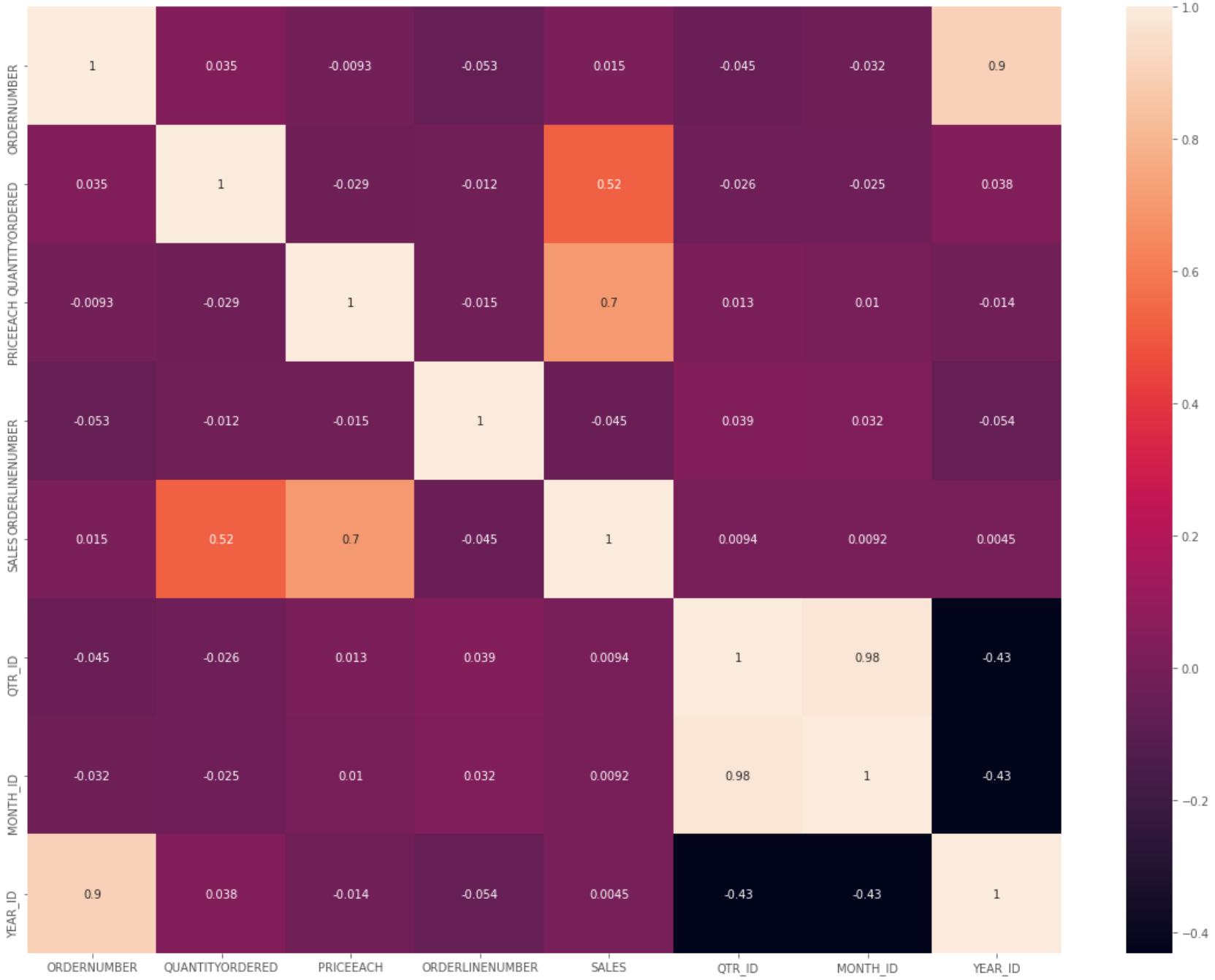
Monthly Active Customers

In [82]:
```
#plt.figure(figsize=(10,8))
df['YEAR_MONTH'] = df['YEAR_ID'].map(str)+df['MONTH_ID'].map(str).map(lambda x: x.rjust(2,'0'))
monthly_active = df.groupby(['YEAR_MONTH'])['CUSTOMERNAME'].nunique().reset_index()
monthly_active.plot(kind='bar',x='YEAR_MONTH',y='CUSTOMERNAME')
#plt.figure(figsize=(10,8))
plt.title('Monthly Active Customers')
plt.xlabel('Month/Year')
plt.ylabel('Number of Unique Customers')
plt.xticks(rotation=90)
#plt.figure(figsize=(10,8))
plt.show()
```



As expected, customers are highly active during the months of November and October. The number of active customers increased from 2003 to 2004 which indicates that the company is successful in retention/acquisition of ol/new customers.

In [139]:
```
#Correlation matrix
plt.figure(figsize = (20, 15))
corr_matrix = df.iloc[:, :10].corr()
sns.heatmap(corr_matrix, annot=True);
```



- From the above output, it can be seen that MSRP is positively correlated to PRICEEACH and SALES.
- PRODUCTCODE is negatively correlated with MSRP, PRICEEACH and SALES.

- There exists positive correlation btw SALES, PRICEEACH, QUANTITYORDERED

Techniques proposed

Segmentation using KMeans Clustering:

The overall aim of segmentation is to identify high yield segments – that is, those segments that are likely to be the most profitable or that have growth potential – so that these can be selected for special attention.

- Check the distribution of the variables
- Removing the skewness by performing log transformation on the variables
- Checking the Distribution of Recency, Frequency and MonetaryValue after Log Transformation.
- Standardizing the variables using StandardScaler() for equal variance and mean
- Choosing number of Clusters using Elbow Method
- Running KMeans with the optimal number of clusters procured from Elbow method.