LP3 Group B Assignment 6

Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

Dataset link : https://www.kaggle.com/datasets/kyanyoga/sample-sales-data

In [1]:

```
#Importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

In [2]:

```
df = pd.read_csv('sales_data_sample.csv', encoding = 'unicode_escape') #Reading the csv file.
df.head()
```

Out[2]:

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

5 rows × 25 columns

In [3]:

```
#Removing the coloumns which dont add value for the analysis.
to_drop = ['PHONE','ADDRESSLINE1','ADDRESSLINE2','CITY','STATE','POSTALCODE','TERRITORY','CONTACTLASTNAME','CONTACT
df = df.drop(to_drop, axis=1)
df.head()
```

Out[3]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	STATUS	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE
0	30	95.70	2	2871.00	Shipped	2	2003	Motorcycles	95	S10_1678
1	34	81.35	5	2765.90	Shipped	5	2003	Motorcycles	95	S10_1678
2	41	94.74	2	3884.34	Shipped	7	2003	Motorcycles	95	S10_1678
3	45	83.26	6	3746.70	Shipped	8	2003	Motorcycles	95	S10_1678
4	49	100.00	14	5205.27	Shipped	10	2003	Motorcycles	95	S10_1678

In [4]:

```
df.nunique() #Checking unique values.
```

Out[4]:

```
QUANTITYORDERED      58
PRICEEACH          1016
ORDERLINENUMBER      18
SALES              2763
STATUS                6
MONTH_ID             12
YEAR_ID               3
PRODUCTLINE           7
MSRP                 80
PRODUCTCODE         109
COUNTRY              19
DEALSIZE              3
dtype: int64
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
QUANTITYORDERED    0
PRICEEACH          0
ORDERLINENUMBER    0
SALES              0
STATUS             0
MONTH_ID           0
YEAR_ID            0
PRODUCTLINE        0
MSRP               0
PRODUCTCODE        0
COUNTRY            0
DEALSIZE           0
dtype: int64
```

In [6]:

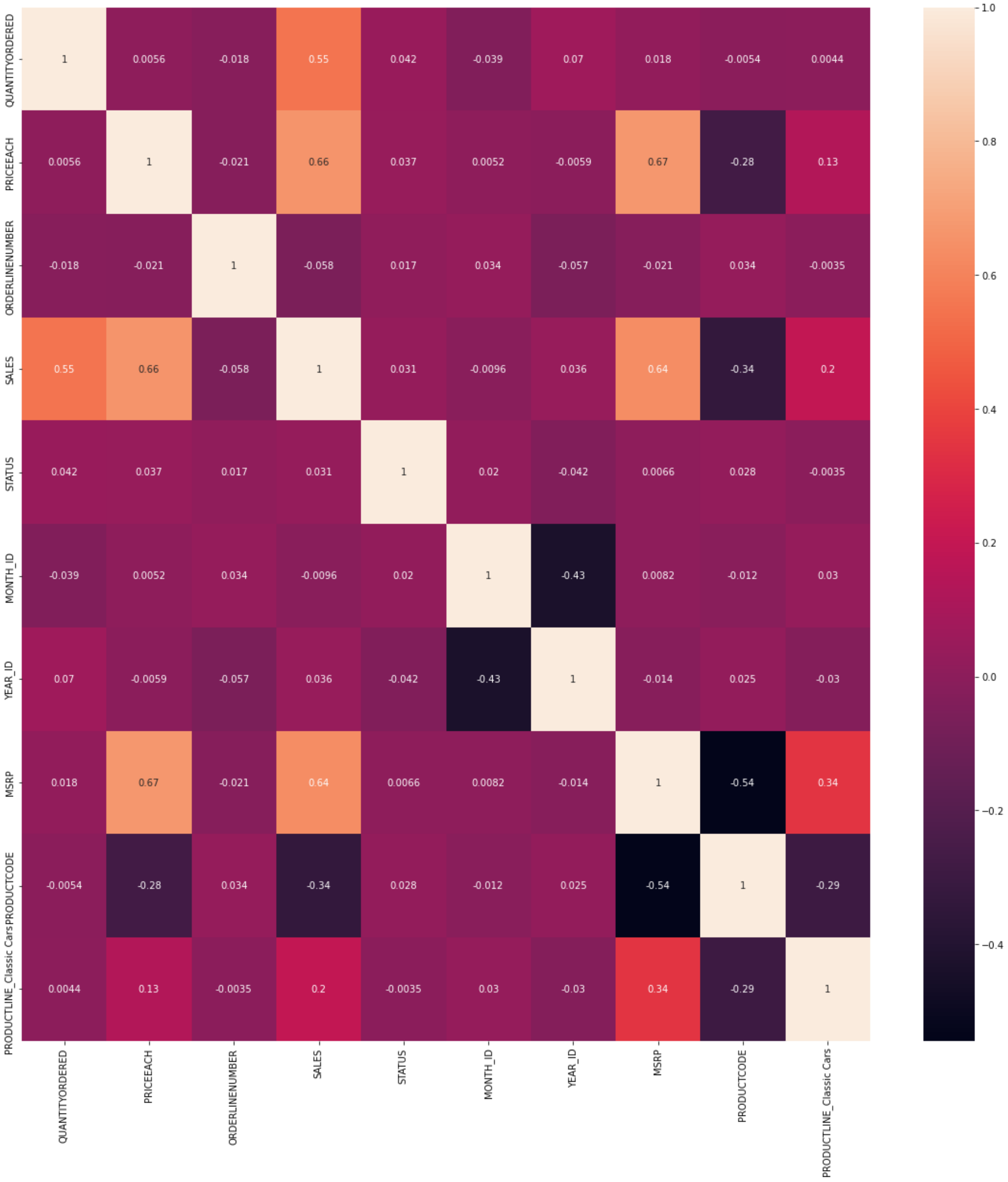
```
#Encodning Categorical Variables for easier processing.
status_dict = {'Shipped':1, 'Cancelled':2, 'On Hold':2, 'Disputed':2, 'In Process':0, 'Resolved':0}
df['STATUS'].replace(status_dict, inplace=True)
df['PRODUCTCODE'] = pd.Categorical(df['PRODUCTCODE']).codes
df = pd.get_dummies(data=df, columns=['PRODUCTLINE', 'DEALSIZE', 'COUNTRY'])
df.dtypes
```

Out[6]:

```
QUANTITYORDERED                    int64
PRICEEACH                        float64
ORDERLINENUMBER                    int64
SALES                            float64
STATUS                             int64
MONTH_ID                           int64
YEAR_ID                            int64
MSRP                               int64
PRODUCTCODE                         int8
PRODUCTLINE_Classic Cars           uint8
PRODUCTLINE_Motorcycles            uint8
PRODUCTLINE_Planes                 uint8
PRODUCTLINE_Ships                  uint8
PRODUCTLINE_Trains                 uint8
PRODUCTLINE_Trucks and Buses       uint8
PRODUCTLINE_Vintage Cars           uint8
DEALSIZE_Large                     uint8
DEALSIZE_Medium                    uint8
DEALSIZE_Small                     uint8
COUNTRY_Australia                  uint8
COUNTRY_Austria                    uint8
COUNTRY_Belgium                    uint8
COUNTRY_Canada                     uint8
COUNTRY_Denmark                    uint8
COUNTRY_Finland                    uint8
COUNTRY_France                     uint8
COUNTRY_Germany                    uint8
COUNTRY_Ireland                    uint8
COUNTRY_Italy                      uint8
COUNTRY_Japan                      uint8
COUNTRY_Norway                     uint8
COUNTRY_Philippines                uint8
COUNTRY_Singapore                  uint8
COUNTRY_Spain                      uint8
COUNTRY_Sweden                     uint8
COUNTRY_Switzerland                uint8
COUNTRY_UK                         uint8
COUNTRY_USA                        uint8
dtype: object
```

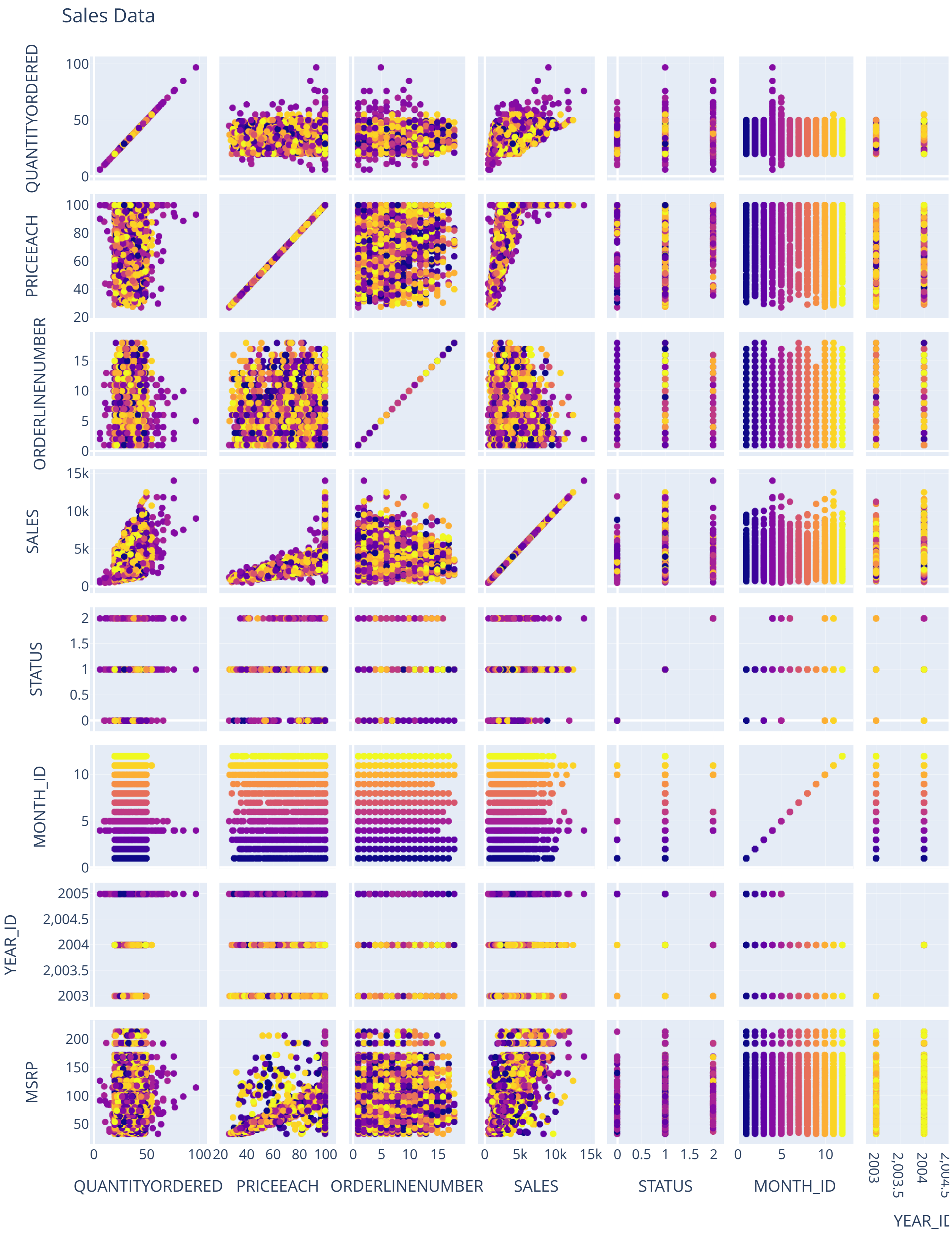
In [7]:

```
#Using Heatmaps to find links between the data
plt.figure(figsize = (20, 20))
corr_matrix = df.iloc[:, :10].corr()
sns.heatmap(corr_matrix, annot=True);
```



In [8]:

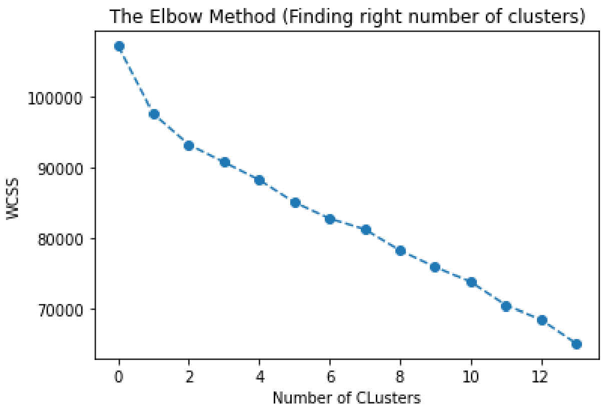
```
#Finding correlation between variables using pairplots
fig = px.scatter_matrix(df, dimensions=df.columns[:8], color='MONTH_ID') #Fill color by months
fig.update_layout(title_text='Sales Data', width=1100, height=1100)
fig.show()
```



In [9]:

```
# Scale the data
std = StandardScaler()
sdf = std.fit_transform(df)
wcss = []
for i in range(1,15):
    km = KMeans(n_clusters=i)
    km.fit(sdf)
    wcss.append(km.inertia_) # intertia is the Sum of squared distances of samples to their closest cluster center

plt.plot(wcss, marker='o', linestyle='--')
plt.title('The Elbow Method (Finding right number of clusters)')
plt.xlabel('Number of CLusters')
plt.ylabel('WCSS')
plt.show()
```



In [10]:

```
#Applying k-means with 5 clusters as the elbow seems to form at 5 clusters
km = KMeans(n_clusters=5, random_state=1)
km.fit(sdf)
cluster_labels = km.labels_
df = df.assign(Cluster=cluster_labels)
df.head()
```

Out[10]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	STATUS	MONTH_ID	YEAR_ID	MSRP	PRODUCTCODE	PRODUCTLINE_
0	30	95.70	2	2871.00	1	2	2003	95	0	
1	34	81.35	5	2765.90	1	5	2003	95	0	
2	41	94.74	2	3884.34	1	7	2003	95	0	
3	45	83.26	6	3746.70	1	8	2003	95	0	
4	49	100.00	14	5205.27	1	10	2003	95	0	

5 rows × 39 columns

In [11]:

```
df = df.groupby(['Cluster']).mean() #Grouping by Cluster
df.head()
```

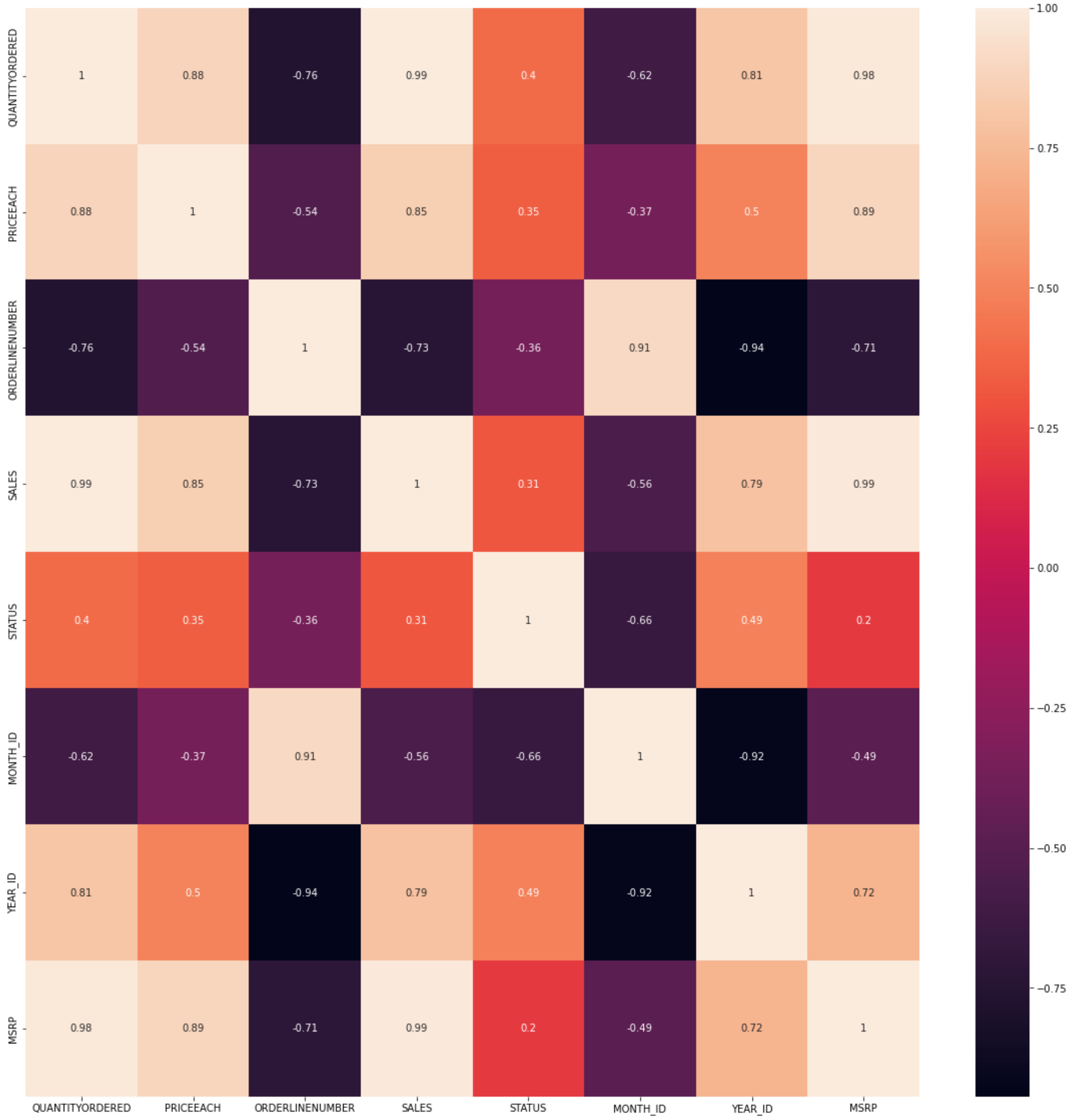
Out[11]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	STATUS	MONTH_ID	YEAR_ID	MSRP	PRODUCTCO
Cluster									
0	30.585766	67.991387	6.575730	2030.427838	0.999088	7.075730	2003.818431	77.130474	59.3859
1	32.773585	81.409434	8.047170	2991.593208	1.000000	7.566038	2003.745283	92.452830	60.3773
2	47.222930	99.799554	5.369427	8293.753248	1.038217	6.770701	2003.910828	158.184713	26.2420
3	37.802589	95.667306	6.319579	4442.814086	1.008091	7.137540	2003.805016	117.737864	45.1359
4	34.793860	83.801711	6.754386	3055.849079	1.065789	6.929825	2003.820175	86.078947	89.5043

5 rows × 38 columns

In [12]:

```
#Heatmap after Kmeans clustering
plt.figure(figsize = (20, 20))
corr_matrix = df.iloc[:, :8].corr()
sns.heatmap(corr_matrix, annot=True);
```


QUANTITYORDERED 1 0.88 -0.76 0.99 0.4 -0.62 0.81 0.98
PRICEEACH 0.88 1 -0.54 0.85 0.35 -0.37 0.5 0.89
ORDERLINENUMBER -0.76 -0.54 1 -0.73 -0.36 0.91 -0.94 -0.71
SALES 0.99 0.85 -0.73 1 0.31 -0.56 0.79 0.99
STATUS 0.4 0.35 -0.36 0.31 1 -0.66 0.49 0.2
MONTH_ID -0.62 -0.37 0.91 -0.56 -0.66 1 -0.92 -0.49
YEAR_ID 0.81 0.5 -0.94 0.79 0.49 -0.92 1 0.72
MSRP 0.98 0.89 -0.71 0.99 0.2 -0.49 0.72 1
QUANTITYORDERED
PRICEEACH
ORDERLINENUMBER
SALES
STATUS
MONTH_ID
YEAR_ID
MSRP
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

In [ ]: