In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.cluster import KMeans
```

In [2]:
```
data = pd.read_csv("sales_data_sample.csv", encoding='unicode_escape')
data.head()
```

Out[2]:

	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 [3]:
```
df = data.copy()
```

In [4]:
```
df.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 [5]:
```
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
df.dtypes
```

Out[5]:
```
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
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 [6]:
```
df.isnull().sum()
```

Out[6]:
```
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 [7]:
```
to_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'POSTALCODE', 'CITY', 'TERRITORY', 'P
df = df.drop(to_drop, axis = 1)
df.head()
```

Out[7]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_II
0	30	95.70	2	2871.00	2003-02-24	Shipped	
1	34	81.35	5	2765.90	2003-05-07	Shipped	
2	41	94.74	2	3884.34	2003-07-01	Shipped	
3	45	83.26	6	3746.70	2003-08-25	Shipped	
4	49	100.00	14	5205.27	2003-10-10	Shipped	

In [8]:
```
df.isnull().sum().sum()
```

Out[8]:
```
0
```

In [9]:
```
df.nunique()
```

Out[9]:
```
QUANTITYORDERED      58
PRICEEACH          1016
ORDERLINENUMBER      18
SALES              2763
ORDERDATE           252
STATUS                6
QTR_ID                4
MONTH_ID             12
YEAR_ID               3
PRODUCTLINE           7
MSRP                 80
PRODUCTCODE         109
COUNTRY              19
DEALSIZE              3
dtype: int64
```

In [10]:
```
df.COUNTRY.unique()
```

Out[10]:
```
array(['USA', 'France', 'Norway', 'Australia', 'Finland', 'Austria', 'UK',
       'Spain', 'Sweden', 'Singapore', 'Canada', 'Japan', 'Italy',
       'Denmark', 'Belgium', 'Philippines', 'Germany', 'Switzerland',
       'Ireland'], dtype=object)
```

In [11]:
```
df.COUNTRY.value_counts()
```

Out[11]:
```
USA            1004
Spain           342
France          314
Australia       185
UK              144
Italy           113
Finland          92
Norway           85
Singapore        79
Canada           70
Denmark          63
Germany          62
Sweden           57
Austria          55
Japan            52
Belgium          33
Switzerland      31
Philippines      26
Ireland          16
Name: COUNTRY, dtype: int64
```

Encode categorical variables

In [12]:
```
status_dict = {'Shipped':1, 'Cancelled':2, 'On Hold':2, 'Disputed':2, 'In Process
df['STATUS'].replace(status_dict, inplace=True)
```

In [13]:
```
df = pd.get_dummies(data=df, columns=['PRODUCTLINE', 'DEALSIZE', 'COUNTRY'])
df.shape
```

Out[13]:
```
(2823, 40)
```

In [14]:
```
df.head()
```

Out[14]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_II
0	30	95.70	2	2871.00	2003-02-24	1	
1	34	81.35	5	2765.90	2003-05-07	1	
2	41	94.74	2	3884.34	2003-07-01	1	
3	45	83.26	6	3746.70	2003-08-25	1	
4	49	100.00	14	5205.27	2003-10-10	1	

5 rows × 40 columns

In [15]:
```
pd.Categorical(df['PRODUCTCODE'])
```

Out[15]:
```
['S10_1678', 'S10_1678', 'S10_1678', 'S10_1678', 'S10_1678', ..., 'S72_3212',
'S72_3212', 'S72_3212', 'S72_3212', 'S72_3212']
Length: 2823
Categories (109, object): ['S10_1678', 'S10_1949', 'S10_2016', 'S10_4698', ...,
'S700_3962', 'S700_4002', 'S72_1253', 'S72_3212']
```

In [16]:
```
pd.Categorical(df['PRODUCTCODE']).codes
```

Out[16]:
```
array([  0,   0,   0, ..., 108, 108, 108], dtype=int8)
```

In [17]:
```
df['PRODUCTCODE'] = pd.Categorical(df['PRODUCTCODE']).codes
```

In [18]:
```
date_group = df.groupby('ORDERDATE').sum()
date_group
```

Out[18]:

ORDERDATE	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	STATUS	QTR_ID
2003-01-06	151	288.78	10	12133.25	4	4
2003-01-09	142	284.96	10	11432.34	4	4
2003-01-10	80	150.14	3	6864.05	2	2
2003-01-29	541	1417.54	136	54702.00	16	16
2003-01-31	443	1061.89	91	44621.96	13	13
...	...	...	...	...	...	...
2005-05-13	259	561.18	21	31821.90	12	12
2005-05-17	509	1269.43	105	59475.10	14	28
2005-05-29	607	1148.40	94	51233.18	0	30
2005-05-30	187	542.16	18	14578.75	0	14
2005-05-31	696	1561.40	112	78918.03	0	38

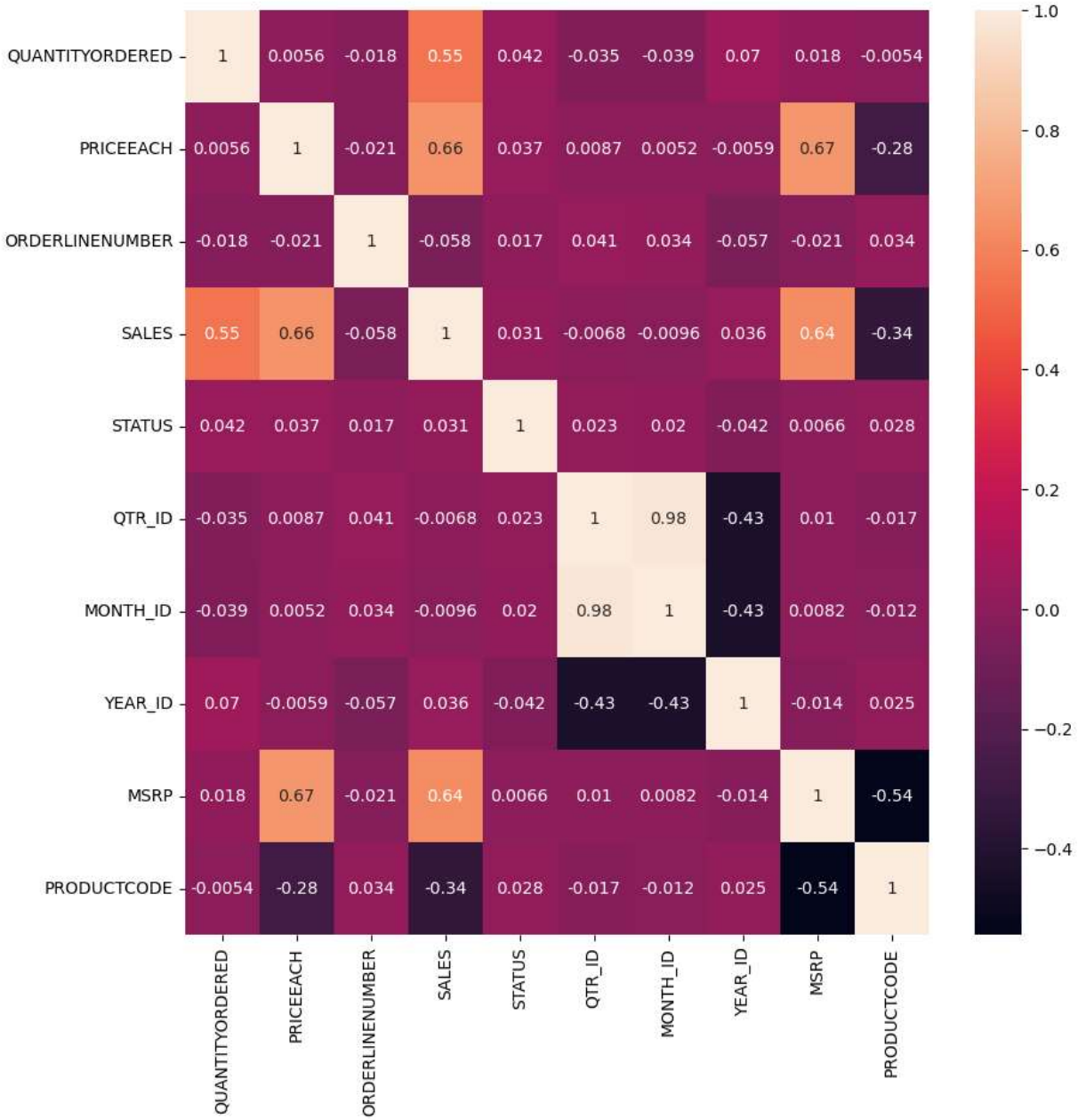
252 rows × 39 columns

In [19]:
```
df.drop("ORDERDATE", axis = 1, inplace = True)
df.shape
```

Out[19]:
```
(2823, 39)
```

In [20]:
```
plt.figure(figsize = (10, 10))
corr_matrix = df.iloc[:, :10].corr()
sns.heatmap(corr_matrix, annot=True)
```

Out[20]: <AxesSubplot: >



OBESRVATIONS

> High co-relation in Quarter ID and the monthly IDs
> SRP is +vely correlated to PRICEEACH and SALES
> RODUCTCODE is -vely correlated with MSRP, PRICEEACH and Sales
> +ve correlation btw SALES, PRICEEACH, QUANTITYORDERED

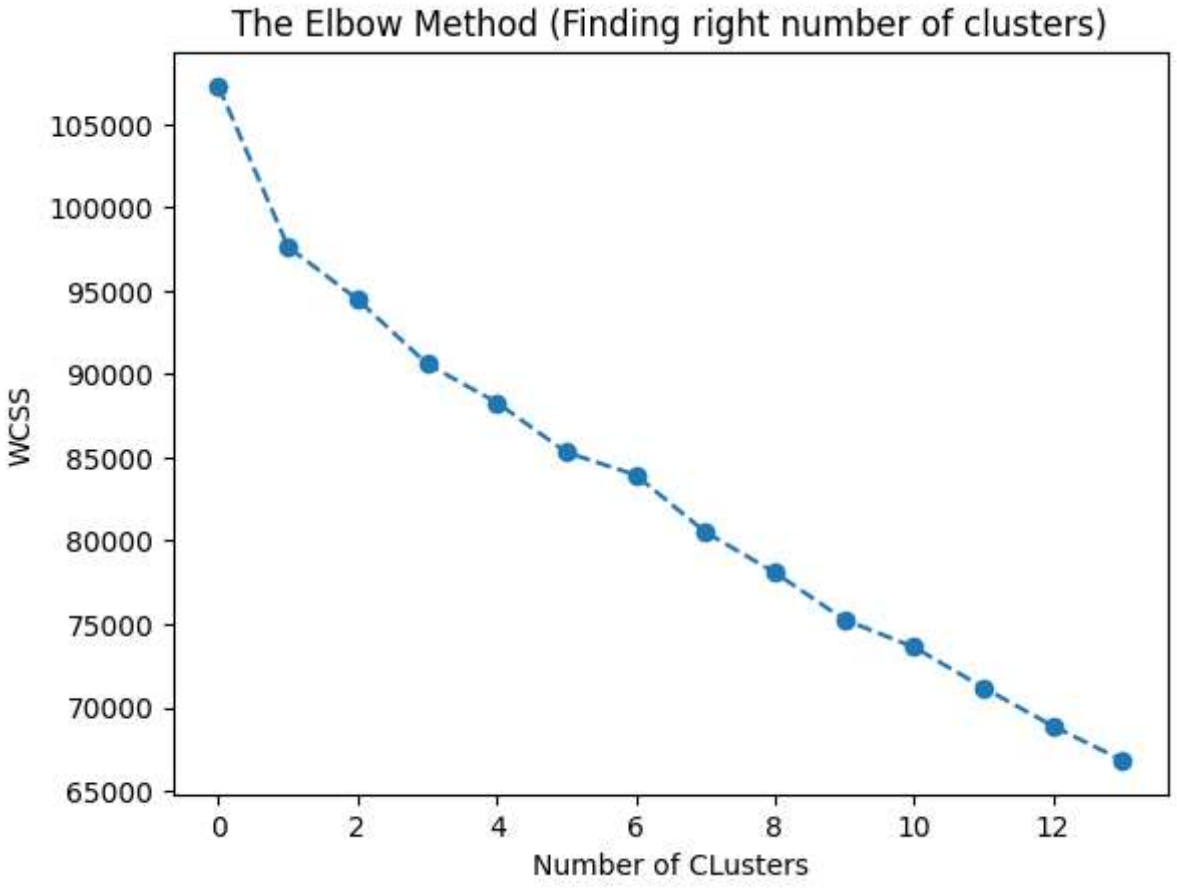
In [21]:
```
df.drop("QTR_ID", axis = 1, inplace = True)
df.shape
```

Out[21]: (2823, 38)

In [22]:
```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
```

In [23]:
```
wcss = [] # Within Cluster Sum of Squares
for i in range(1,15):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_) # intertia is the Sum of squared distances of sa

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()
```



From this we can observe that, 5th cluster seems to be forming the elbow of the curve. after that we will apply auto encoders to solve this problem

In [24]:
```
kmeans = KMeans(n_clusters=5, init='k-means++')
kmeans.fit(df_scaled)
labels = kmeans.labels_
labels
```

Out[24]:
```
array([4, 4, 2, ..., 1, 1, 1])
```

In [25]:
```
cluster_centers = pd.DataFrame(data= kmeans.cluster_centers_, columns= df.columns
cluster_centers
```

Out[25]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	STATUS	MONTH_ID	YEAR
0	0.013139	-0.396807	0.160159	-0.334319	0.056857	0.010234	-0.032
1	-0.037171	0.009763	0.078809	-0.271913	0.198113	-0.043991	0.001
2	0.377900	0.623575	-0.061666	0.714258	-0.001949	0.005300	-0.003
3	-0.032713	0.191391	0.439377	0.132130	-0.039370	-0.492927	0.264
4	-0.470383	-0.768894	0.037511	-0.826199	-0.042754	0.015927	-0.000

5 rows × 38 columns