In [1]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

In [2]:

df = pd.read\_csv('sales\_data\_sample.csv',encoding='unicode\_escape')

df.head()

Out[2]:

|  | **ORDERNUMBER** | **QUANTITYORDERED** | **PRICEEACH** | **ORDERLINENUMBER** | **SALES** | **ORDERDATE** | **STATUS** | **QTR\_ID** | **MONTH\_ID** | **YEAR\_ID** | **...** | **ADDRESSLINE1** | **ADDRESSLINE2** | **CITY** | **STATE** | **POSTALCODE** | **COUNTRY** | **TERRITORY** | **CONTACTLASTNAME** | **CONTACTFIRSTNAME** | **DEALSIZE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 | Shipped | 1 | 2 | 2003 | ... | 897 Long Airport Avenue | NaN | NYC | NY | 10022 | USA | NaN | Yu | Kwai | Small |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 | Shipped | 2 | 5 | 2003 | ... | 59 rue de l'Abbaye | NaN | Reims | NaN | 51100 | France | EMEA | Henriot | Paul | Small |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 | Shipped | 3 | 7 | 2003 | ... | 27 rue du Colonel Pierre Avia | NaN | Paris | NaN | 75508 | France | EMEA | Da Cunha | Daniel | Medium |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 | Shipped | 3 | 8 | 2003 | ... | 78934 Hillside Dr. | NaN | Pasadena | CA | 90003 | USA | NaN | Young | Julie | Medium |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 | Shipped | 4 | 10 | 2003 | ... | 7734 Strong St. | NaN | San Francisco | CA | NaN | USA | NaN | Brown | Julie | Medium |

5 rows × 25 columns

In [3]:

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 [4]:

df\_drop = ['ADDRESSLINE1', 'ADDRESSLINE2', 'POSTALCODE', 'CITY', 'TERRITORY', 'PHONE', 'STATE', 'CONTACTFIRSTNAME', 'CONTACTLASTNAME', 'CUSTOMERNAME', 'ORDERNUMBER']

df = df.drop(df\_drop, axis=1)

In [5]:

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2823 entries, 0 to 2822

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 QUANTITYORDERED 2823 non-null int64

1 PRICEEACH 2823 non-null float64

2 ORDERLINENUMBER 2823 non-null int64

3 SALES 2823 non-null float64

4 ORDERDATE 2823 non-null object

5 STATUS 2823 non-null object

6 QTR\_ID 2823 non-null int64

7 MONTH\_ID 2823 non-null int64

8 YEAR\_ID 2823 non-null int64

9 PRODUCTLINE 2823 non-null object

10 MSRP 2823 non-null int64

11 PRODUCTCODE 2823 non-null object

12 COUNTRY 2823 non-null object

13 DEALSIZE 2823 non-null object

dtypes: float64(2), int64(6), object(6)

memory usage: 308.9+ KB

In [6]:

for col in df.columns.values:

print(df[col].value\_counts())

34 112

21 103

46 101

27 100

31 97

41 97

45 97

26 96

29 94

48 94

25 94

20 93

33 92

22 92

32 91

24 91

38 91

49 91

36 89

44 89

37 87

43 85

39 84

28 82

40 78

42 76

30 75

23 73

35 71

47 70

50 65

55 16

66 5

15 4

51 4

61 3

18 3

60 3

76 3

59 3

56 3

19 3

64 3

10 2

6 2

11 2

54 2

70 2

97 1

85 1

62 1

52 1

16 1

13 1

58 1

65 1

12 1

77 1

Name: QUANTITYORDERED, dtype: int64

100.00 1304

59.87 6

96.34 6

57.73 5

80.55 5

...

48.30 1

87.96 1

36.21 1

98.48 1

62.24 1

Name: PRICEEACH, Length: 1016, dtype: int64

1 307

2 291

3 270

4 256

5 239

6 221

7 197

8 187

9 165

10 141

11 128

12 110

13 97

14 81

15 56

16 42

17 25

18 10

Name: ORDERLINENUMBER, dtype: int64

3003.00 3

5464.69 2

2257.92 2

5004.80 2

2172.48 2

..

2312.24 1

2793.71 1

1908.28 1

3441.37 1

2116.16 1

Name: SALES, Length: 2763, dtype: int64

11/14/2003 0:00 38

11/24/2004 0:00 35

11/12/2003 0:00 34

11/17/2004 0:00 32

11/4/2004 0:00 29

..

4/20/2004 0:00 1

8/4/2004 0:00 1

2/2/2004 0:00 1

8/28/2004 0:00 1

4/21/2003 0:00 1

Name: ORDERDATE, Length: 252, dtype: int64

Shipped 2617

Cancelled 60

Resolved 47

On Hold 44

In Process 41

Disputed 14

Name: STATUS, dtype: int64

4 1094

1 665

2 561

3 503

Name: QTR\_ID, dtype: int64

11 597

10 317

5 252

1 229

2 224

3 212

8 191

12 180

4 178

9 171

7 141

6 131

Name: MONTH\_ID, dtype: int64

2004 1345

2003 1000

2005 478

Name: YEAR\_ID, dtype: int64

Classic Cars 967

Vintage Cars 607

Motorcycles 331

Planes 306

Trucks and Buses 301

Ships 234

Trains 77

Name: PRODUCTLINE, dtype: int64

118 104

99 103

136 80

62 78

68 77

...

73 23

41 22

170 22

71 22

92 22

Name: MSRP, Length: 80, dtype: int64

S18\_3232 52

S10\_1949 28

S24\_1444 28

S10\_4962 28

S24\_2840 28

..

S18\_1749 22

S24\_2887 22

S24\_3969 22

S18\_4409 22

S18\_4933 22

Name: PRODUCTCODE, Length: 109, dtype: int64

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

Medium 1384

Small 1282

Large 157

Name: DEALSIZE, dtype: int64

In [7]:

df.drop(columns=['ORDERDATE','STATUS','MONTH\_ID','QTR\_ID','YEAR\_ID'],inplace=True)

df.head()

Out[7]:

|  | **QUANTITYORDERED** | **PRICEEACH** | **ORDERLINENUMBER** | **SALES** | **PRODUCTLINE** | **MSRP** | **PRODUCTCODE** | **COUNTRY** | **DEALSIZE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 30 | 95.70 | 2 | 2871.00 | Motorcycles | 95 | S10\_1678 | USA | Small |
| **1** | 34 | 81.35 | 5 | 2765.90 | Motorcycles | 95 | S10\_1678 | France | Small |
| **2** | 41 | 94.74 | 2 | 3884.34 | Motorcycles | 95 | S10\_1678 | France | Medium |
| **3** | 45 | 83.26 | 6 | 3746.70 | Motorcycles | 95 | S10\_1678 | USA | Medium |
| **4** | 49 | 100.00 | 14 | 5205.27 | Motorcycles | 95 | S10\_1678 | USA | Medium |
|  |  |  |  |  |  |  |  |  |  |

In [8]:

from sklearn.preprocessing import LabelEncoder

def convert\_categories(col):

le = LabelEncoder()

df[col] = le.fit\_transform(df[col].values)

In [9]:

categories = ['PRODUCTLINE','PRODUCTCODE','COUNTRY','DEALSIZE']

for col in categories:

convert\_categories(col)

In [10]:

df.head()

Out[10]:

|  | **QUANTITYORDERED** | **PRICEEACH** | **ORDERLINENUMBER** | **SALES** | **PRODUCTLINE** | **MSRP** | **PRODUCTCODE** | **COUNTRY** | **DEALSIZE** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 30 | 95.70 | 2 | 2871.00 | 1 | 95 | 0 | 18 | 2 |
| **1** | 34 | 81.35 | 5 | 2765.90 | 1 | 95 | 0 | 6 | 2 |
| **2** | 41 | 94.74 | 2 | 3884.34 | 1 | 95 | 0 | 6 | 1 |
| **3** | 45 | 83.26 | 6 | 3746.70 | 1 | 95 | 0 | 18 | 1 |
| **4** | 49 | 100.00 | 14 | 5205.27 | 1 | 95 | 0 | 18 | 1 |

In [11]:

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

data = sc.fit\_transform(df)

In [12]:

from sklearn.cluster import KMeans

wcss = []

for k in range(1,15):

kmeans = KMeans(n\_clusters=k,init='k-means++',random\_state=15)

kmeans.fit(data)

wcss.append(kmeans.inertia\_)

In [13]:

k = list(range(1,15))

plt.plot(k,wcss)

plt.xlabel('Clusters')

plt.ylabel('scores')

plt.title('Finding right number of clusters')

plt.grid()

plt.show()

In [ ]: