In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings(action= 'ignore')
```

In [2]:

```
#reading the excel file
data = pd.read_excel('Project 6-Segmenting customers into clusters-Dataset.xlsx')
```

In [3]:

```
data.head()
```

Out[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [4]:

```
#checking for null values
data.isnull().sum()
```

Out[4]:

```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

In [5]:

```
#shape oof the data
data.shape
```

Out[5]:

```
(541909, 8)
```

In [6]:

```
#percentage of missing values
data.isnull().sum()/len(data)*100
```

Out[6]:

```
InvoiceNo       0.000000
StockCode       0.000000
Description     0.268311
Quantity        0.000000
InvoiceDate     0.000000
UnitPrice       0.000000
CustomerID     24.926694
Country         0.000000
dtype: float64
```

In [7]:

```
data.dtypes
```

Out[7]:

```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
dtype: object
```

In [8]:

```
#typecasting the variables
data['StockCode'] = data['StockCode'].astype('category')
data['Quantity'] = data['Quantity'].astype('float64')
data['Description'] = data['Description'].astype('category')
data['Country'] = data['Country'].astype('category')
data['InvoiceNo'] = data['InvoiceNo'].astype('category')
```

In [9]:

```
#converting datetime variable
data['InvoiceDate']= pd.to_datetime(data['InvoiceDate'])
```

In [10]:

```
#extracting new variables from datetime
data['weekday'] = data.InvoiceDate.dt.weekday
data['hour'] = data.InvoiceDate.dt.hour
data['month'] = data.InvoiceDate.dt.month
```

In [11]:

```
data.head(10)
```

Out[11]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	weekday	hour	month
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6.0	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2	8	12
1	536365	71053	WHITE METAL LANTERN	6.0	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2	8	12
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8.0	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2	8	12
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6.0	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2	8	12
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6.0	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2	8	12
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2.0	2010-12-01 08:26:00	7.65	17850.0	United Kingdom	2	8	12
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6.0	2010-12-01 08:26:00	4.25	17850.0	United Kingdom	2	8	12
7	536366	22633	HAND WARMER UNION JACK	6.0	2010-12-01 08:28:00	1.85	17850.0	United Kingdom	2	8	12
8	536366	22632	HAND WARMER RED POLKA DOT	6.0	2010-12-01 08:28:00	1.85	17850.0	United Kingdom	2	8	12
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32.0	2010-12-01 08:34:00	1.69	13047.0	United Kingdom	2	8	12

In [12]:

```
data.shape
```

Out[12]:

```
(541909, 11)
```

In [13]:

```
data['Description'].nunique()
```

Out[13]:

```
4223
```

In [14]:

```
data['Description'].mode()
```

Out[14]:

```
0    WHITE HANGING HEART T-LIGHT HOLDER
Name: Description, dtype: category
Categories (4223, object): [20713, ' 4 PURPLE FLOCK DINNER CANDLES', ' 50'S CHRISTMAS GIFT BAG LARGE', ' DOLLY GIRL BEAKER', ..., 'wrongly marked. 23343 in box', 'wrongly sold (227
19) barcode', 'wrongly sold as sets', 'wrongly sold sets']
```

In [15]:

```
#imputing the missing values
data['Description'].fillna("WHITE HANGING HEART T-LIGHT HOLDER", inplace= True)
```

In [16]:

```
data['Description'].isnull().sum()
```

Out[16]:

```
0
```

In [17]:

```
data['CustomerID'].nunique()
```

Out[17]:

```
4372
```

In [18]:

```
data['CustomerID'].mode()
```

Out[18]:

```
0    17841.0
dtype: float64
```

In [19]:

```
data['CustomerID'].fillna(17841.0 , inplace= True)
```

In [20]:

```
data.isnull().sum()
```

Out[20]:

```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
weekday        0
hour           0
month          0
dtype: int64
```

In [21]:

```
data.shape
```

Out[21]:

```
(541909, 11)
```

In [22]:

```
#summary of the data
data.describe()
```

Out[22]:

	Quantity	UnitPrice	CustomerID	weekday	hour	month
count	541909.000000	541909.000000	541909.000000	541909.000000	541909.000000	541909.000000
mean	9.552250	4.611114	15924.146207	2.431277	13.078729	7.553128
std	218.081158	96.759853	1850.531104	1.844709	2.443270	3.509055
min	-80995.000000	-11062.060000	12346.000000	0.000000	6.000000	1.000000
25%	1.000000	1.250000	14367.000000	1.000000	11.000000	5.000000
50%	3.000000	2.080000	16249.000000	2.000000	13.000000	8.000000
75%	10.000000	4.130000	17841.000000	4.000000	15.000000	11.000000
max	80995.000000	38970.000000	18287.000000	6.000000	20.000000	12.000000

In [23]:

```
#separating the numerical columns
df = data.select_dtypes(include='number')
```

In [24]:

```
df.head()
```

Out[24]:

	Quantity	UnitPrice	CustomerID	weekday	hour	month
0	6.0	2.55	17850.0	2	8	12
1	6.0	3.39	17850.0	2	8	12
2	8.0	2.75	17850.0	2	8	12
3	6.0	3.39	17850.0	2	8	12
4	6.0	3.39	17850.0	2	8	12

In [25]:

```
data = data.drop(['Quantity','UnitPrice','CustomerID','weekday','hour','month','InvoiceDate','StockCode'], axis = 1)
```

In [26]:

```
data.head()
```

Out[26]:

	InvoiceNo	Description	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	United Kingdom
1	536365	WHITE METAL LANTERN	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	United Kingdom

In [27]:

```
from sklearn.preprocessing import StandardScaler
```

In [28]:

```
#standardizing the numerical data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
df = pd.DataFrame(df_scaled)
```

In [29]:

```
df.head()
```

Out[29]:

	0	1	2	3	4	5
0	-0.016289	-0.021301	1.040704	-0.233792	-2.078662	1.267257
1	-0.016289	-0.012620	1.040704	-0.233792	-2.078662	1.267257
2	-0.007118	-0.019234	1.040704	-0.233792	-2.078662	1.267257
3	-0.016289	-0.012620	1.040704	-0.233792	-2.078662	1.267257
4	-0.016289	-0.012620	1.040704	-0.233792	-2.078662	1.267257

In [30]:

```
km = KMeans(n_clusters = 2)
```

In [31]:

```
#fitting data
km.fit(df)
```

Out[31]:

```
KMeans(n_clusters=2)
```

In [32]:

```
#getting predictions
pred_k = km.predict(df)
```

In [33]:

```
pred_k
```

Out[33]:

```
array([1, 1, 1, ..., 0, 0, 0])
```

In [34]:

```
pd.Series(pred_k).value_counts()
```

Out[34]:

```
1    291013
0    250896
dtype: int64
```

In [35]:

```
km.inertia_
```

Out[35]:

```
2786610.845142297
```

In [36]:

```
km.score(df)
```

Out[36]:

```
-2786610.8451422974
```

In [37]:

```
#creating empty list
list = []
```

In [38]:

```
#running loop for 1 to 20 clusters each, fitting the data and appending result in the empty list created earlier
for clusters in range(1,20):
    km = KMeans(n_jobs = -1, n_clusters = clusters)
    km.fit(df)
    list.append(km.inertia_)
```

In [39]:

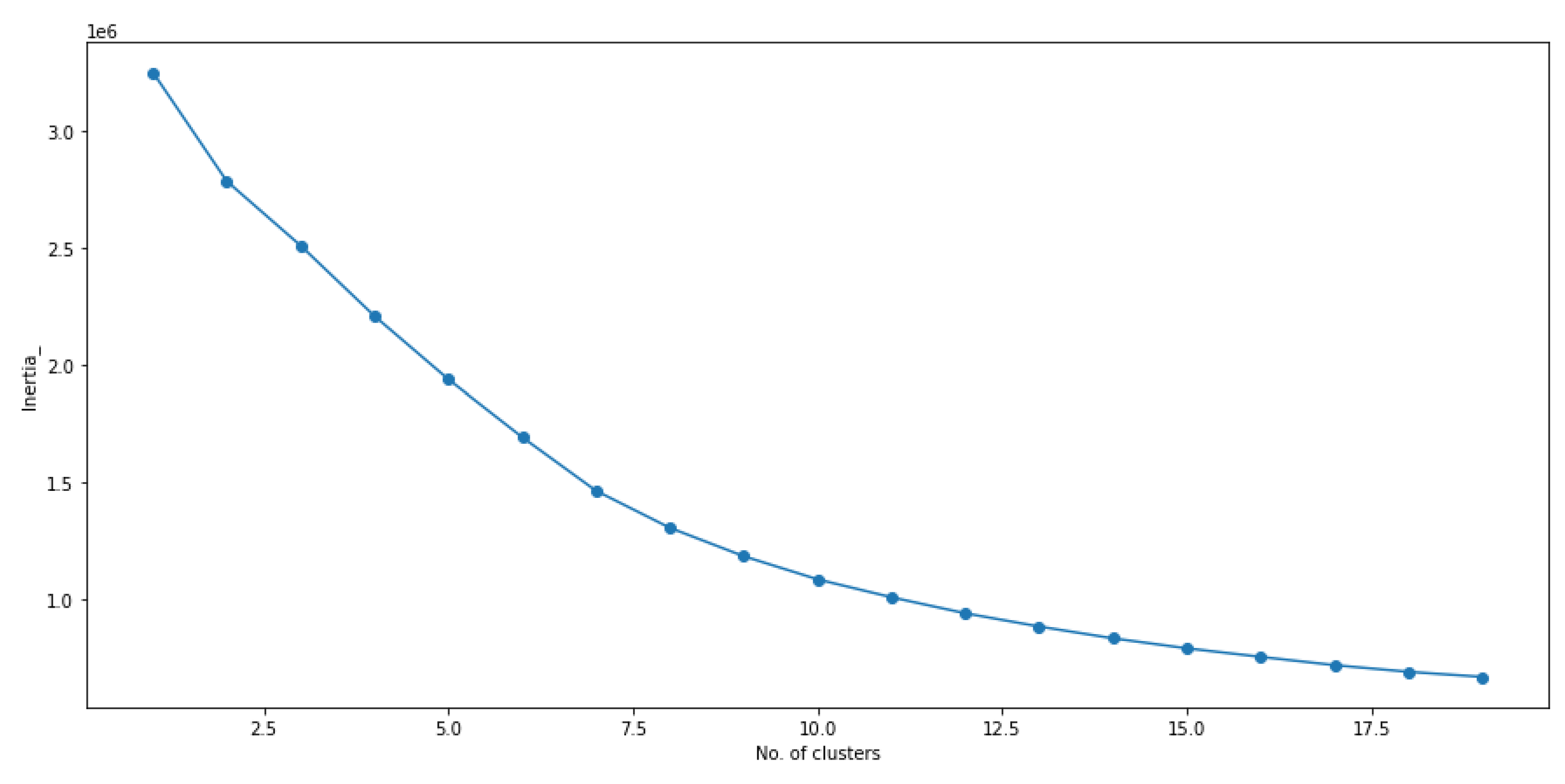
```
#converting results in a dataframe
frame = pd.DataFrame({'Cluster':range(1,20), 'SSE': list})
```

In [40]:

```
#plotting the elbow curve
plt.figure(figsize =(15,7))
plt.plot(frame['Cluster'], frame['SSE'], marker = 'o')
plt.xlabel('No. of clusters')
plt.ylabel('Inertia_')
```

Out[40]:

```
Text(0, 0.5, 'Inertia_')
```



In [50]:

```
#fitting data and getting predictions
kmeans = KMeans(n_jobs = -1, n_clusters=4)
kmeans.fit(df)
pred_k = kmeans.predict(df)
```

In [51]:

```
pred_k
```

Out[51]:

```
array([0, 0, 0, ..., 3, 3, 3])
```

In [52]:

```
#putting results in a dataframe
frame = pd.DataFrame(df)
frame['clusters'] = pred_k
```

In [53]:

```
#counting values in each cluster
frame['clusters'].value_counts()
```

Out[53]:

```
0    156589
3    151267
2    124164
1    109889
Name: clusters, dtype: int64
```

In [54]:

```
frame.head()
```

Out[54]:

	0	1	2	3	4	5	clusters
0	-0.016289	-0.021301	1.040704	-0.233792	-2.078662	1.267257	0
1	-0.016289	-0.012620	1.040704	-0.233792	-2.078662	1.267257	0
2	-0.007118	-0.019234	1.040704	-0.233792	-2.078662	1.267257	0
3	-0.016289	-0.012620	1.040704	-0.233792	-2.078662	1.267257	0
4	-0.016289	-0.012620	1.040704	-0.233792	-2.078662	1.267257	0

- We can see addition of a new column called cluster which tells us which cluster each observation belongs to.