```
#importing the dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

# Data collection
customer_data = pd.read_csv(r'C:\Users\HP\Downloads\Snapdeal Project\
Online Retail.csv')

# first 5 rows in the dataframe
customer_data.head()

  InvoiceNo StockCode                          Description
Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6

1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6


      InvoiceDate  UnitPrice  CustomerID         Country
0  12/1/2010 8:26       2.55     17850.0  United Kingdom
1  12/1/2010 8:26       3.39     17850.0  United Kingdom
2  12/1/2010 8:26       2.75     17850.0  United Kingdom
3  12/1/2010 8:26       3.39     17850.0  United Kingdom
4  12/1/2010 8:26       3.39     17850.0  United Kingdom

# finding the number of rows of columns
customer_data . shape

(541909, 8)

# getting information about the data
customer_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

```
#getting the description of the data set
customer_data.describe()
```

```
            Quantity      UnitPrice     CustomerID
count  541909.000000  541909.000000  406829.000000
mean        9.552250       4.611114   15287.690570
std       218.081158      96.759853    1713.600303
min    -80995.000000  -11062.060000   12346.000000
25%         1.000000       1.250000   13953.000000
50%         3.000000       2.080000   15152.000000
75%        10.000000       4.130000   16791.000000
max     80995.000000   38970.000000   18287.000000
```

```
# checking whether is any missing value or not
customer_data.isnull().sum()
```

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

```
# lets fill the description
customer_data['Description'].fillna('Unknown', inplace=True)
# lets drop the column customerID
df=customer_data.drop(columns=['CustomerID'])


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

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

```
# checking duplicates if present on specific columns only
duplicate_mask = df.duplicated(subset=['InvoiceNo'])
print(duplicate_mask)


0         False
1          True
2          True
3          True
4          True
          ...
541904     True
541905     True
541906     True
541907     True
541908     True
Length: 541909, dtype: bool

# let see the duplicate rows only
duplicate_rows = df[duplicate_mask]
print(duplicate_rows)

       InvoiceNo StockCode                          Description
Quantity  \
1         536365     71053                  WHITE METAL LANTERN
6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER
8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE
6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.
6
5         536365     22752         SET 7 BABUSHKA NESTING BOXES
2
...          ...       ...                                  ...
...
541904    581587     22613          PACK OF 20 SPACEBOY NAPKINS
12
541905    581587     22899         CHILDREN'S APRON DOLLY GIRL
6
541906    581587     23254        CHILDRENS CUTLERY DOLLY GIRL
4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE
4
541908    581587     22138        BAKING SET 9 PIECE RETROSPOT
3

            InvoiceDate  UnitPrice         Country
1        12/1/2010 8:26       3.39  United Kingdom
2        12/1/2010 8:26       2.75  United Kingdom
```

```
3        12/1/2010 8:26       3.39  United Kingdom
4        12/1/2010 8:26       3.39  United Kingdom
5        12/1/2010 8:26       7.65  United Kingdom
...                 ...        ...             ...
541904  12/9/2011 12:50       0.85          France
541905  12/9/2011 12:50       2.10          France
541906  12/9/2011 12:50       4.15          France
541907  12/9/2011 12:50       4.15          France
541908  12/9/2011 12:50       4.95          France

[516009 rows x 7 columns]

#drop the duplicate rows
df_cleaned = df.drop_duplicates(subset=['InvoiceNo'], keep='first')
print (df_cleaned)

       InvoiceNo StockCode                          Description
Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER
6
7         536366     22633               HAND WARMER UNION JACK
6
9         536367     84879        ASSORTED COLOUR BIRD ORNAMENT
32
21        536368     22960             JAM MAKING SET WITH JARS
6
25        536369     21756             BATH BUILDING BLOCK WORD
3
...          ...       ...                                  ...
...
541865    581583     20725              LUNCH BAG RED RETROSPOT
40
541867    581584     20832     RED FLOCK LOVE HEART PHOTO FRAME
72
541869    581585     22481        BLACK TEA TOWEL CLASSIC DESIGN
12
541890    581586     22061  LARGE CAKE STAND  HANGING STRAWBERY
8
541894    581587     22631             CIRCUS PARADE LUNCH BOX
12

            InvoiceDate  UnitPrice         Country
0        12/1/2010 8:26       2.55  United Kingdom
7        12/1/2010 8:28       1.85  United Kingdom
9        12/1/2010 8:34       1.69  United Kingdom
21       12/1/2010 8:34       4.25  United Kingdom
25       12/1/2010 8:35       5.95  United Kingdom
...                 ...        ...             ...
541865  12/9/2011 12:23       1.45  United Kingdom
541867  12/9/2011 12:25       0.72  United Kingdom
```

```
541869  12/9/2011 12:31       0.39  United Kingdom
541890  12/9/2011 12:49       2.95  United Kingdom
541894  12/9/2011 12:50       1.95          France

[25900 rows x 7 columns]

# let optimize the column  types
customer_data = df_cleaned.convert_dtypes()
print(customer_data)

       InvoiceNo StockCode                          Description
Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER
6
7         536366     22633                HAND WARMER UNION JACK
6
9         536367     84879         ASSORTED COLOUR BIRD ORNAMENT
32
21        536368     22960               JAM MAKING SET WITH JARS
6
25        536369     21756               BATH BUILDING BLOCK WORD
3
...          ...       ...                                  ...
...
541865    581583     20725                LUNCH BAG RED RETROSPOT
40
541867    581584     20832       RED FLOCK LOVE HEART PHOTO FRAME
72
541869    581585     22481         BLACK TEA TOWEL CLASSIC DESIGN
12
541890    581586     22061  LARGE CAKE STAND  HANGING STRAWBERY
8
541894    581587     22631               CIRCUS PARADE LUNCH BOX
12

            InvoiceDate  UnitPrice         Country
0         12/1/2010 8:26       2.55  United Kingdom
7         12/1/2010 8:28       1.85  United Kingdom
9         12/1/2010 8:34       1.69  United Kingdom
21        12/1/2010 8:34       4.25  United Kingdom
25        12/1/2010 8:35       5.95  United Kingdom
...                  ...        ...             ...
541865  12/9/2011 12:23       1.45  United Kingdom
541867  12/9/2011 12:25       0.72  United Kingdom
541869  12/9/2011 12:31       0.39  United Kingdom
541890  12/9/2011 12:49       2.95  United Kingdom
541894  12/9/2011 12:50       1.95          France

[25900 rows x 7 columns]
```

```
# standarise the data
numeric_cols = customer_data.select_dtypes(include='number').columns
scaled_array = std.fit_transform(customer_data[numeric_cols])
df_std = pd.DataFrame(scaled_array, columns=numeric_cols)
print (df_std)

       Quantity  UnitPrice
0     -0.008077  -0.044565
1     -0.008077  -0.046178
2      0.018268  -0.046547
3     -0.008077  -0.040646
4     -0.011117  -0.036728
...         ...        ...
25895  0.026375  -0.047100
25896  0.058800  -0.048783
25897 -0.001997  -0.049543
25898 -0.006050  -0.043643
25899 -0.001997  -0.045948

[25900 rows x 2 columns]

# One-hot encode the 'Country' column, dropping the first category to
avoid redundancy
df_encoded = pd.get_dummies(
    customer_data,
    columns=['Country'],
    drop_first=True,
    dtype=int
)

# Combine standardized numeric data with encoded categorical data
df_preprocessed = pd.concat([df_std, df_encoded], axis=1)
df_preprocessed['InvoiceDate'] = customer_data['InvoiceDate']
print(df_preprocessed)

#creating new feature
# Ensure it's a datetime type
df_preprocessed['InvoiceDate'] =
pd.to_datetime(df_preprocessed['InvoiceDate'])

# Extract date-based features
df_preprocessed['DayOfWeek'] =
df_preprocessed['InvoiceDate'].dt.dayofweek
df_preprocessed['IsWeekend']  = df_preprocessed['DayOfWeek'].isin([5,
6]).astype(int)
df_preprocessed['Month']      =
df_preprocessed['InvoiceDate'].dt.month
```