In [1]:
```
#Import Required Libraries
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

In [2]:
```
#Load Datase
df = pd.read_csv("online_retail_II.csv", encoding="ISO-8859-1")
df.head()
```

Out[2]:

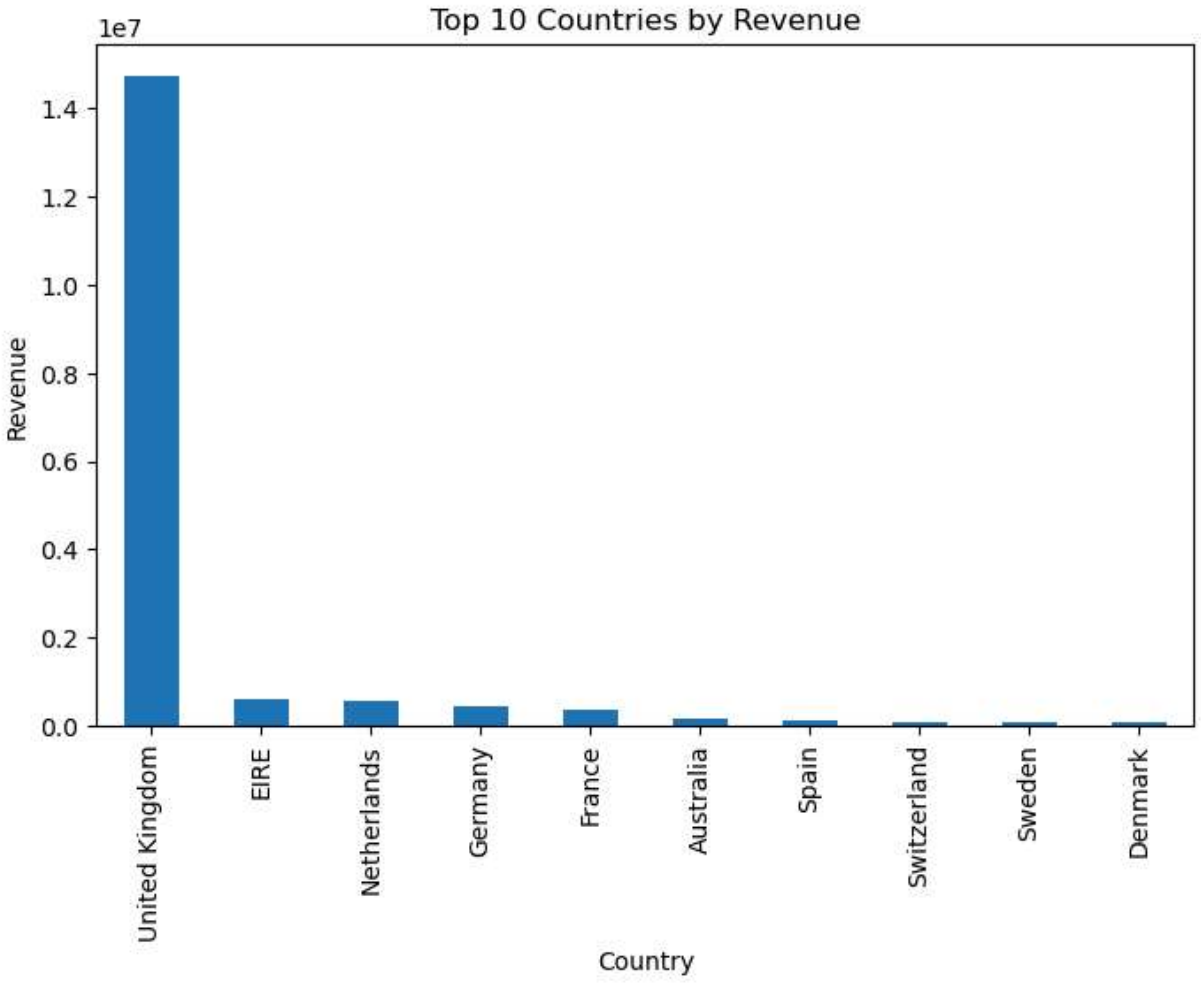
	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

In [3]:
```
#Data Cleaning
# Remove rows with missing Customer ID
df = df.dropna(subset=['Customer ID'])

# Remove cancelled / invalid transactions
df = df[df['Quantity'] > 0]
df = df[df['Price'] > 0]

# Convert InvoiceDate to datetime
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

df.info()
```

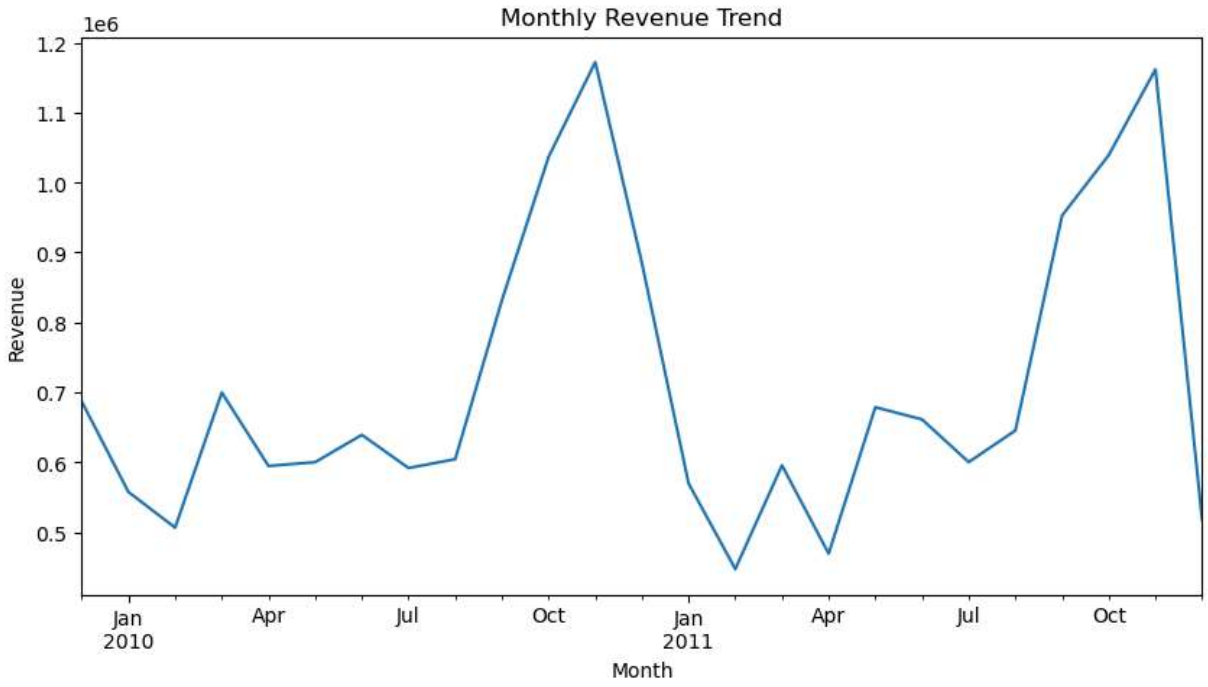
```
<class 'pandas.core.frame.DataFrame'>
Index: 805549 entries, 0 to 1067370
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      805549 non-null  object
 1   StockCode    805549 non-null  object
 2   Description  805549 non-null  object
 3   Quantity     805549 non-null  int64
 4   InvoiceDate  805549 non-null  datetime64[ns]
 5   Price        805549 non-null  float64
 6   Customer ID  805549 non-null  float64
 7   Country      805549 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 55.3+ MB
```

In [4]:
```
#Feature Engineering – Total Spend
df['Total_Spend'] = df['Quantity'] * df['Price']
```

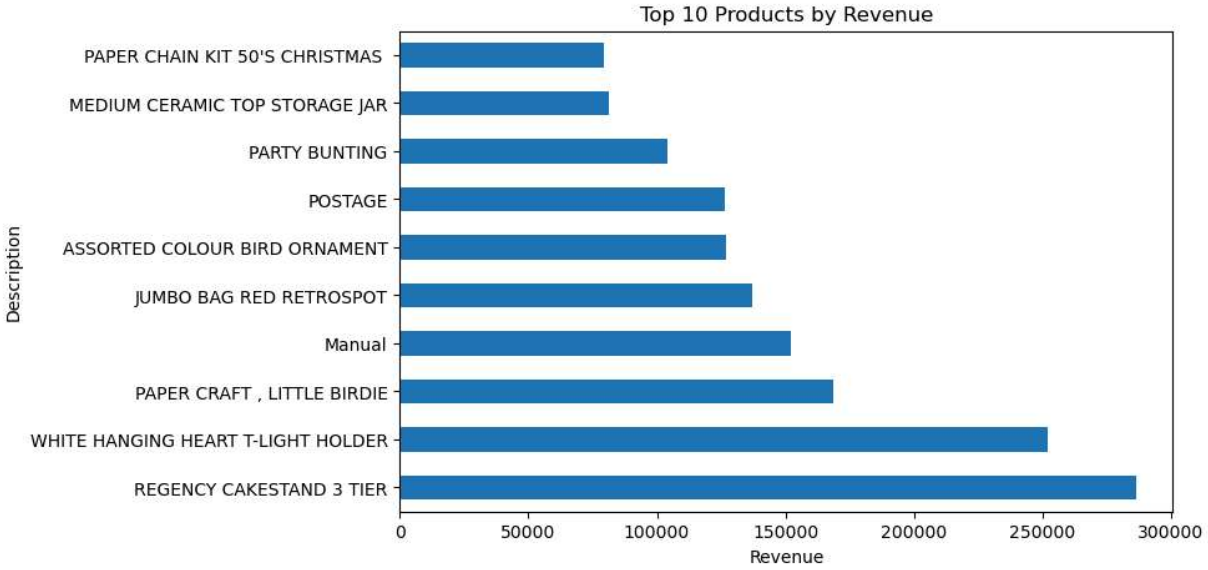
In [5]:
```
#Exploratory Data Analysis (EDA)
#Revenue by Country (Top 10)

country_revenue = (
    df.groupby('Country')['Total_Spend']
    .sum()
    .sort_values(ascending=False)
    .head(10)
)

plt.figure(figsize=(8,5))
country_revenue.plot(kind='bar')
plt.title("Top 10 Countries by Revenue")
plt.ylabel("Revenue")
plt.show()
```



In [6]:
```
#Monthly Revenue Trend
df['Month'] = df['InvoiceDate'].dt.to_period('M')

monthly_revenue = df.groupby('Month')['Total_Spend'].sum()

plt.figure(figsize=(10,5))
monthly_revenue.plot()
plt.title("Monthly Revenue Trend")
plt.ylabel("Revenue")
plt.xlabel("Month")
plt.show()
```

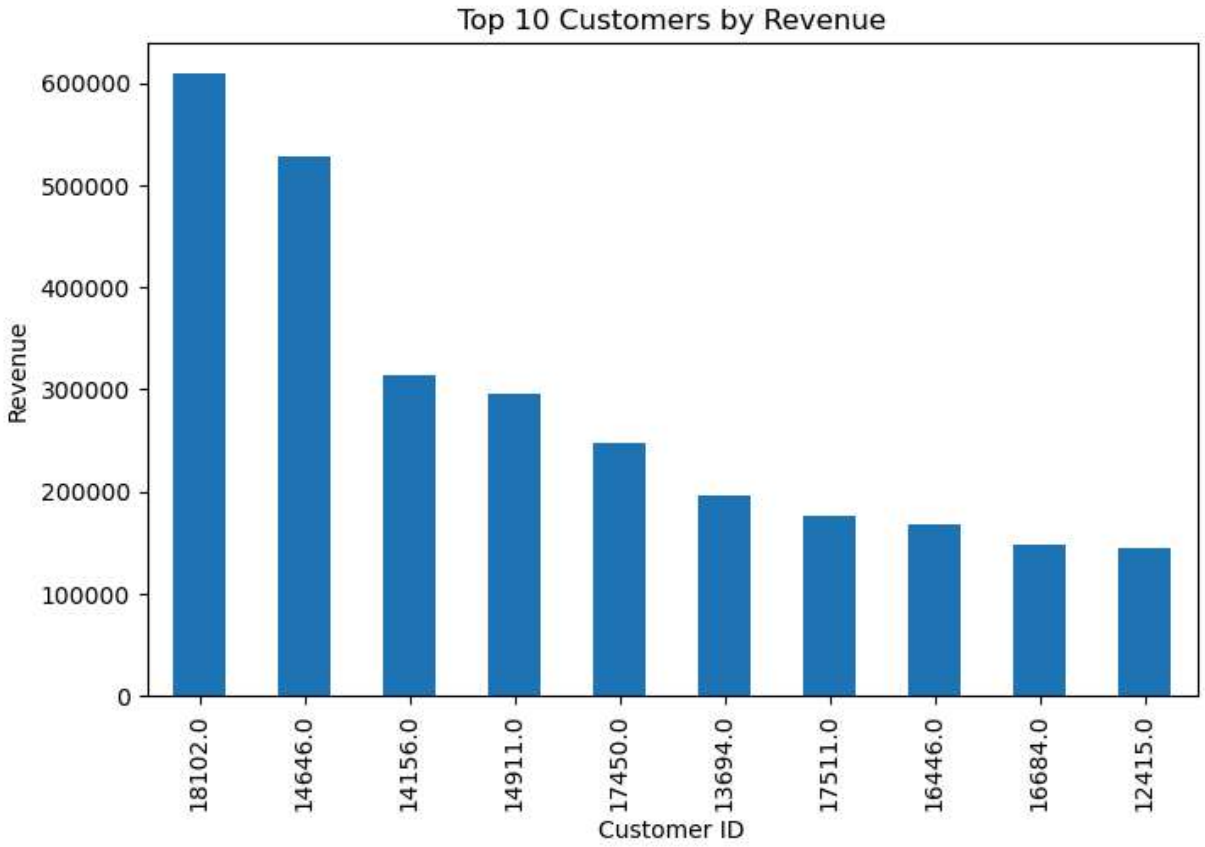


In [7]:
```
#Top 10 Products by Revenue
top_products = (
    df.groupby('Description')['Total_Spend']
    .sum()
    .sort_values(ascending=False)
    .head(10)
)

plt.figure(figsize=(8,5))
top_products.plot(kind='barh')
plt.title("Top 10 Products by Revenue")
plt.xlabel("Revenue")
plt.show()
```

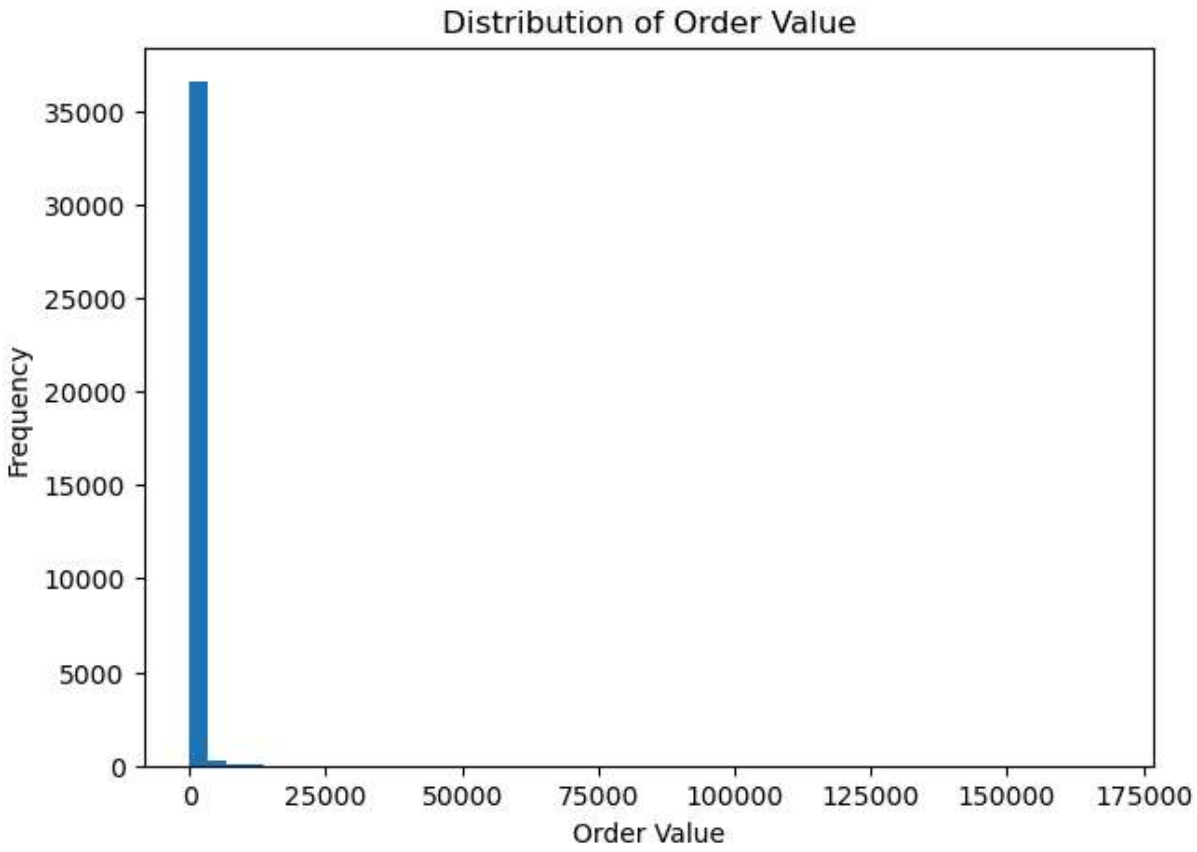


In [8]:
```
#Top Customers by Revenue
```

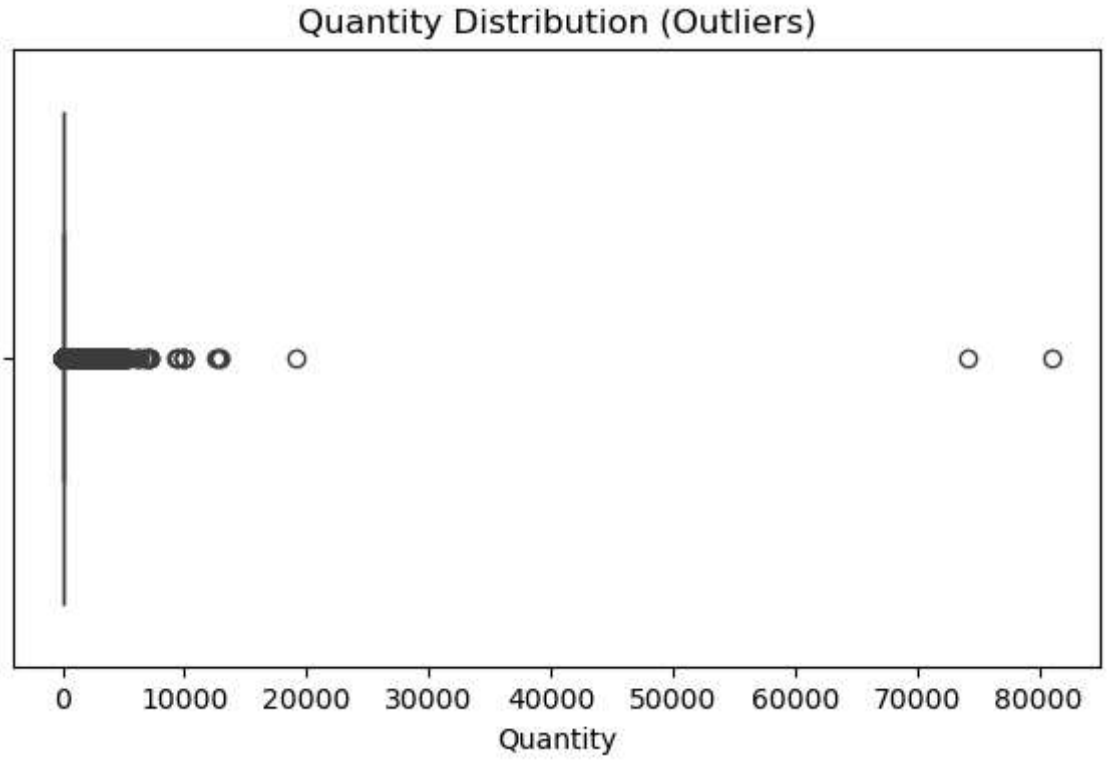
```
top_customers = (
    df.groupby('Customer ID')['Total_Spend']
    .sum()
    .sort_values(ascending=False)
    .head(10)
)

plt.figure(figsize=(8,5))
top_customers.plot(kind='bar')
plt.title("Top 10 Customers by Revenue")
plt.ylabel("Revenue")
plt.show()
```



In [9]:
```
#Order Value Distribution
order_value = df.groupby('Invoice')['Total_Spend'].sum()

plt.figure(figsize=(7,5))
plt.hist(order_value, bins=50)
plt.title("Distribution of Order Value")
plt.xlabel("Order Value")
plt.ylabel("Frequency")
plt.show()
```



In [10]:
```
#Quantity Distribution
plt.figure(figsize=(7,4))
sns.boxplot(x=df['Quantity'])
plt.title("Quantity Distribution (Outliers)")
plt.show()
```



In [11]:
```
df.columns
```

Out[11]:
```
Index(['Invoice', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'Price', 'Customer ID', 'Country', 'Total_Spend', 'Month'],
      dtype='object')
```

In [14]:
```
reference_date = df['InvoiceDate'].max() + pd.Timedelta(days=1)

rfm = df.groupby('Customer ID').agg({
    'InvoiceDate': lambda x: (reference_date - x.max()).days,
    'Invoice': 'nunique',
    'Total_Spend': 'sum'
})

rfm.rename(columns={
    'InvoiceDate': 'Recency',
    'Invoice': 'Frequency',
    'Total_Spend': 'Monetary'
}, inplace=True)

rfm.head()
```

Out[14]:

Customer ID	Recency	Frequency	Monetary
12346.0	326	12	77556.46
12347.0	2	8	5633.32
12348.0	75	5	2019.40
12349.0	19	4	4428.69
12350.0	310	1	334.40

In [15]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm[['Recency', 'Frequency', 'Monetary']])
```

In [16]:
```
from sklearn.cluster import MiniBatchKMeans

kmeans = MiniBatchKMeans(
    n_clusters=4,
    random_state=42,
    batch_size=2048
)

rfm['Cluster'] = kmeans.fit_predict(rfm_scaled)
```

In [17]:
```
rfm.head()
```

Out[17]:

Customer ID	Recency	Frequency	Monetary	Cluster
12346.0	326	12	77556.46	2
12347.0	2	8	5633.32	3
12348.0	75	5	2019.40	0
12349.0	19	4	4428.69	3
12350.0	310	1	334.40	1

In [18]:
```
rfm['Customer_Segment'] = rfm['Cluster'].map({
    0: 'High Value Customers',
    1: 'Loyal Customers',
    2: 'At Risk Customers',
    3: 'Low Value Customers'
})
```

In [19]:
```
df = df.merge(
    rfm[['Customer_Segment']],
    left_on='Customer ID',
    right_index=True,
    how='left'
)
```

In [20]:
```
df[['Customer ID', 'Customer_Segment']].head()
```

Out[20]:

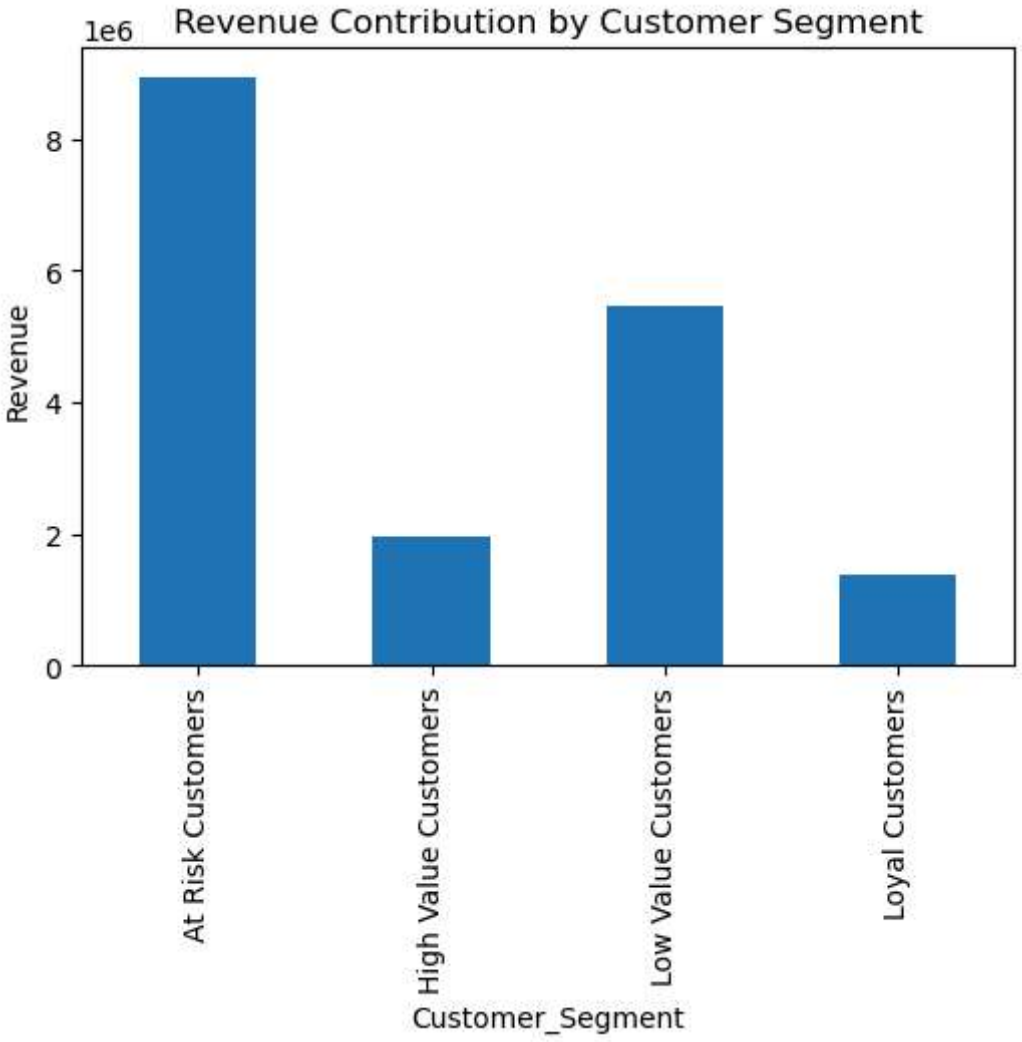
	Customer ID	Customer_Segment
0	13085.0	High Value Customers
1	13085.0	High Value Customers
2	13085.0	High Value Customers
3	13085.0	High Value Customers
4	13085.0	High Value Customers

In [21]:
```
segment_summary = rfm.groupby('Customer_Segment').agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'Monetary': 'mean',
    'Customer_Segment': 'count'
}).rename(columns={'Customer_Segment': 'Customer_Count'})

segment_summary
```

Out[21]:

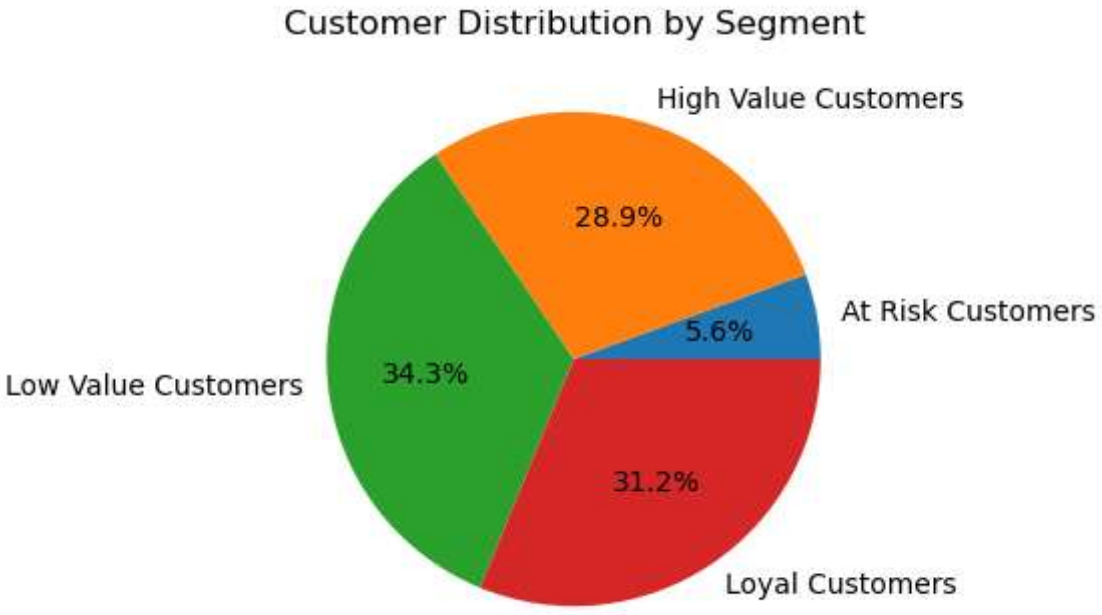
Customer_Segment	Recency	Frequency	Monetary	Customer_Count
At Risk Customers	26.975535	40.064220	27305.300346	327
High Value Customers	143.022366	3.296645	1160.715084	1699
Low Value Customers	25.671620	7.104012	2707.976407	2019
Loyal Customers	479.968358	2.140753	750.210951	1833

In [22]:
```
segment_revenue = df.groupby('Customer_Segment')['Total_Spend'].sum()

plt.figure(figsize=(6,4))
segment_revenue.plot(kind='bar')
plt.title("Revenue Contribution by Customer Segment")
plt.ylabel("Revenue")
plt.show()
```



In [23]:
```
segment_count = df[['Customer ID', 'Customer_Segment']].drop_duplicates() \
    .groupby('Customer_Segment').size()

plt.figure(figsize=(6,4))
```

```
segment_count.plot(kind='pie', autopct='%1.1f%%')
plt.title("Customer Distribution by Segment")
plt.ylabel("")
plt.show()
```



In [24]:
```
!pip install psycopg2-binary sqlalchemy
```

```
Requirement already satisfied: psycopg2-binary in c:\users\dell\anaconda3\lib\site-p
ackages (2.9.11)
Requirement already satisfied: sqlalchemy in c:\users\dell\anaconda3\lib\site-packag
es (2.0.30)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\dell\anaconda3\l
ib\site-packages (from sqlalchemy) (4.15.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\dell\anaconda3\lib\site-
packages (from sqlalchemy) (3.0.1)
```

In [31]:
```
from sqlalchemy import create_engine
from sqlalchemy.engine import URL

# Fix Period datatype
df["Month"] = df["Month"].astype(str)

url = URL.create(
    drivername="postgresql+psycopg2",
    username="postgres",
    password="Archana@30",
    host="localhost",
    port=5432,
    database="retail_db"
)

engine = create_engine(url)

engine.connect()
print("Connected successfully!")
```

```
df.to_sql(
    "online_retail_transactions",
    engine,
    if_exists="replace",
    index=False
)

print("Data successfully loaded into PostgreSQL!")
```

```
Connected successfully!
Data successfully loaded into PostgreSQL!
```

In [ ]: