In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import sklearn
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
dm=pd.read_excel('Book1.xlsx', parse_dates=['InvoiceDate'])
```

In [3]:
```
dm.head(3)
```

Out[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-01-12 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-01-12 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-01-12 08:26:00	2.75	17850.0	United Kingdom

In [4]:
```
dm.isnull().sum()
```

Out[4]:
```
InvoiceNo            0
StockCode            0
Description       1454
Quantity             0
InvoiceDate          0
UnitPrice            0
CustomerID      133600
Country              0
dtype: int64
```

In [5]:
```
#removeing missing values for customer_id
dm['CustomerID'].fillna(dm['CustomerID'].mode()[0], inplace=True)
```

In [6]:
```
# we can see that we have a Quanatity with a negtive value
dm.Quantity.min()
```

Out[6]:
```
-80995
```

In [7]:
```
#removing the a negtive value
dm = dm[(dm['Quantity']> 0)]
```

In [8]:
```
dm['TotalSales']=dm['Quantity'] * dm['UnitPrice']
```

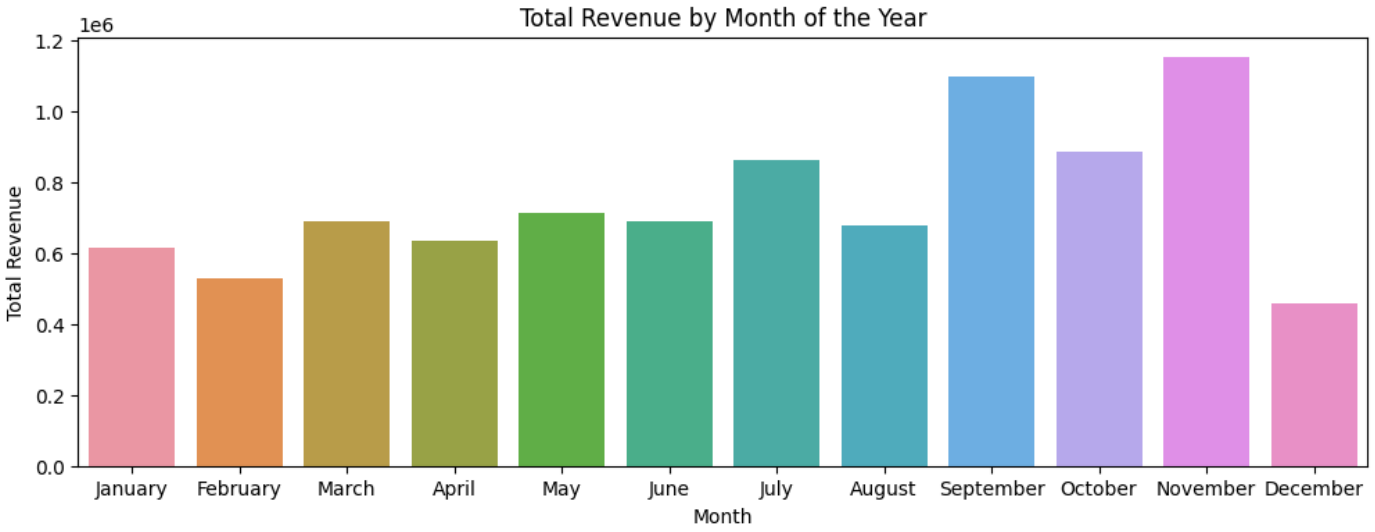
In [9]:
```
dm.shape
```

Out[9]:
```
(486286, 9)
```

EDA

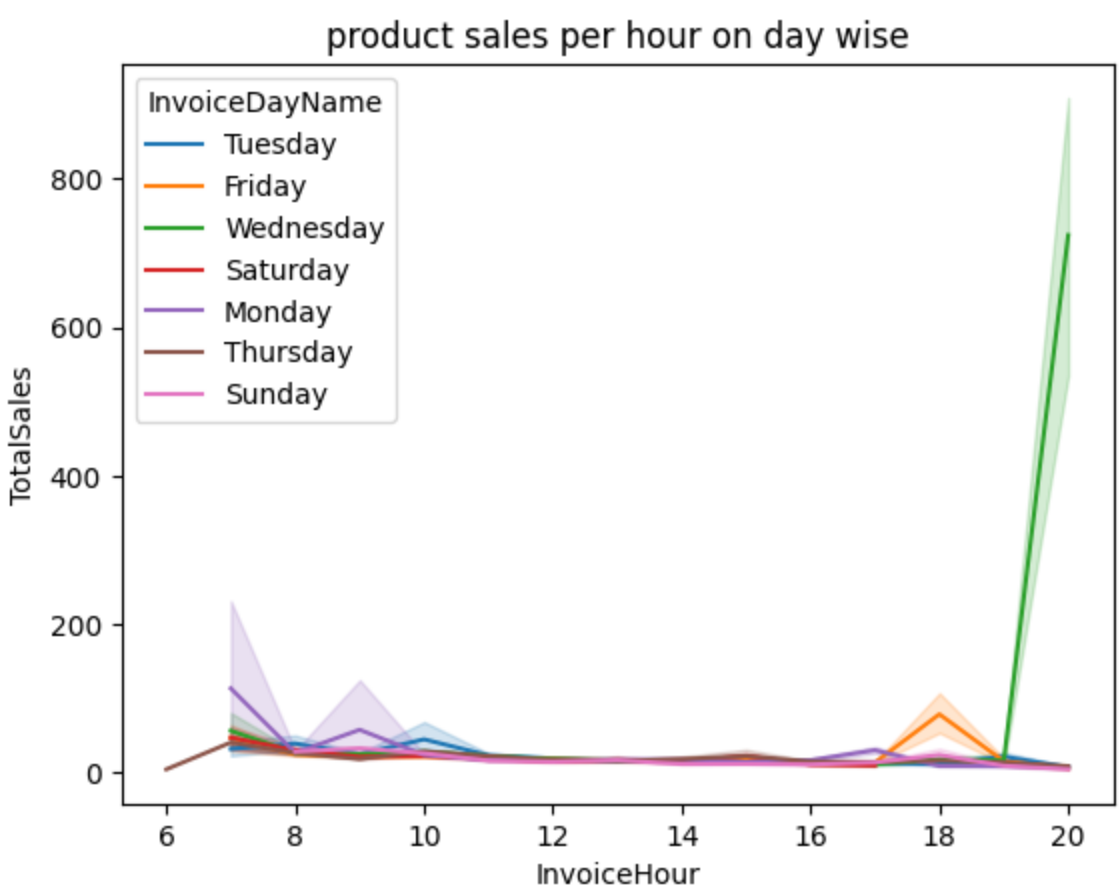
In [10]:
```
# extracting time related features from InvoiceDate

dm["InvoiceDateDay"] = dm["InvoiceDate"].dt.date
dm["InvoiceDateTime"] = dm["InvoiceDate"].dt.time
dm["InvoiceYear"] = dm["InvoiceDate"].dt.year
dm["InvoiceMonth"] = dm["InvoiceDate"].dt.month
dm["InvoiceMonthName"] = dm["InvoiceDate"].dt.month_name()
dm["InvoiceDay"] = dm["InvoiceDate"].dt.day
dm["InvoiceDayName"] = dm["InvoiceDate"].dt.day_name()
dm["InvoiceDayOfWeek"] = dm["InvoiceDate"].dt.day_of_week
dm["InvoiceWeekOfYear"] = dm["InvoiceDate"].dt.weekofyear
dm["InvoiceHour"] = dm["InvoiceDate"].dt.hour

dm["TotalSales"] = dm["Quantity"] * dm["UnitPrice"]
```

In [11]:
```
#5.2. Revenue by Month of the Year
revenue_month = dm.groupby(["InvoiceMonth","InvoiceMonthName"])["TotalSales"].sum().rese
plt.figure(figsize=(12, 4))
plt.title("Total Revenue by Month of the Year")
sns.barplot(data=revenue_month, x="InvoiceMonthName", y="TotalSales")
plt.xlabel("Month")
plt.ylabel("Total Revenue")
plt.show()
```



In [12]:
```
sns.lineplot(x='InvoiceHour',y='TotalSales',data=dm,hue='InvoiceDayName').set_title('pro
```

Out[12]:
```
Text(0.5, 1.0, 'product sales per hour on day wise')
```



we can see from invoicehours Wednesday at 19:00 more revenue collecte

Customer segmentation with KMeans

In [13]:
```
most_recent_date = dm['InvoiceDate'].max()
customer_dm = dm.groupby('CustomerID').agg({'InvoiceDate': lambda x: (most_recent_date -
                                            'InvoiceNo': 'count',
                                            'TotalSales': 'sum'})
customer_dm.rename(columns={'InvoiceDate':'Recency', 'InvoiceNo':'Frequency', 'TotalSale
customer_dm.head(5)
```

Out[13]:

CustomerID	Recency	Frequency	Monetary
12346.0	326	1	77183.60
12747.0	23	103	4196.01
12748.0	4	4596	33719.73
12749.0	23	199	4090.88
12820.0	45	59	942.34

In [14]:
```
customer_dm.describe()
```

Out[14]:

	Recency	Frequency	Monetary
count	3921.000000	3921.000000	3.921000e+03

mean	105.554195	124.020913	2.296123e+03
std	115.037406	2238.164352	2.867550e+04
min	0.000000	1.000000	0.000000e+00
25%	22.000000	17.000000	3.000400e+02
50%	61.000000	41.000000	6.518200e+02
75%	162.000000	99.000000	1.575890e+03
max	697.000000	139788.000000	1.735698e+06

In [15]:
```
#recency stat
customer_dm.Recency.describe()
```

Out[15]:
```
count    3921.000000
mean      105.554195
std       115.037406
min         0.000000
25%        22.000000
50%        61.000000
75%       162.000000
max       697.000000
Name: Recency, dtype: float64
```

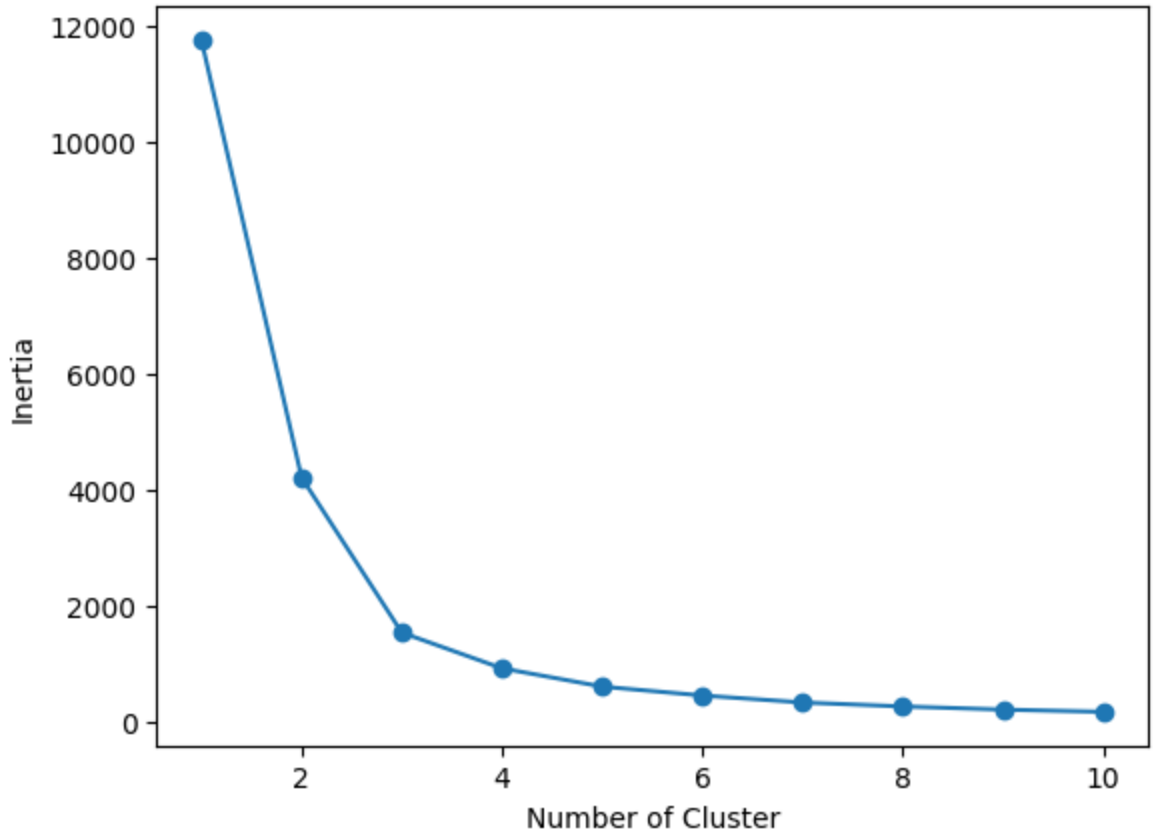
In [16]:
```
scaler = StandardScaler()
norm_dm = scaler.fit_transform(customer_dm)
norm_dm
```

Out[16]:
```
array([[ 1.91654114, -0.0549721 ,  2.61188182],
       [-0.71772067, -0.00939323,  0.06626316],
       [-0.88290541,  1.99831145,  1.09597427],
       ...,
       [ 0.96020847, -0.05005674, -0.07387291],
       [-0.83074181,  0.28240093, -0.00701883],
       [-0.91768114, -0.02413934, -0.01600326]])
```

In [17]:
```
inertia = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(norm_dm)
    inertia.append(kmeans.inertia_)
```

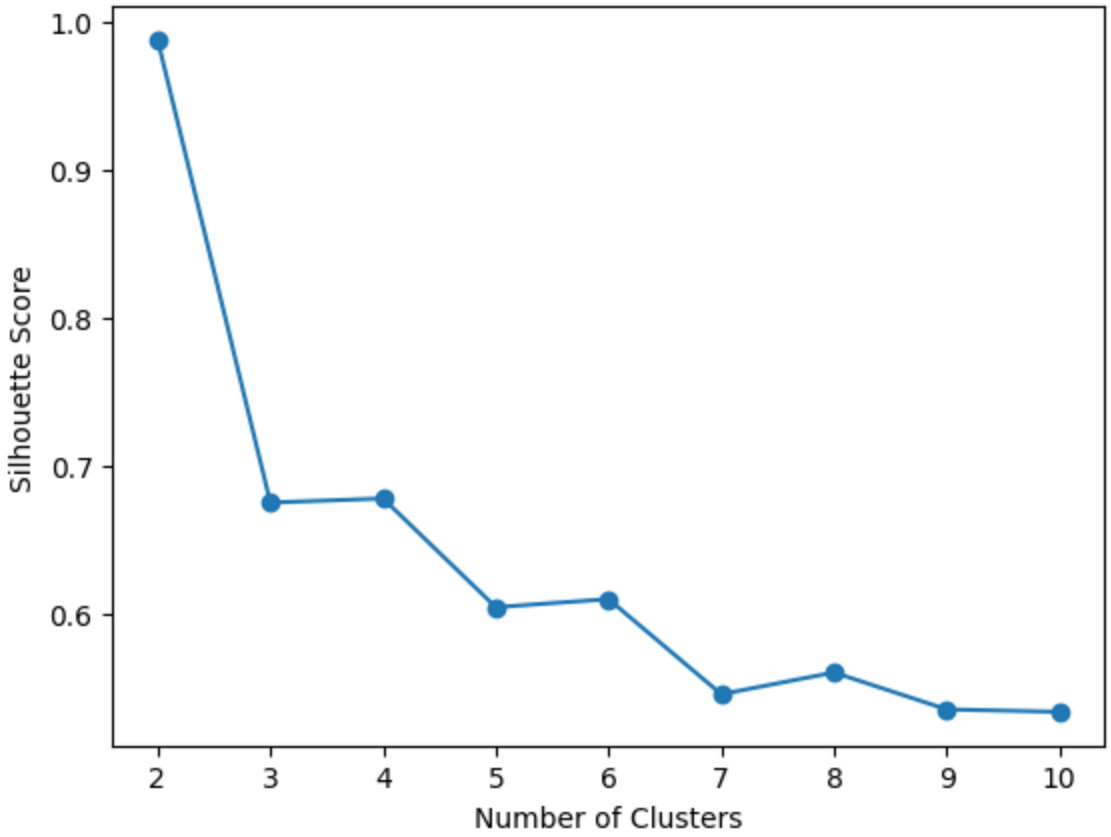
In [18]:
```
plt.plot(range(1,11),
         inertia,
         marker='o')
plt.xlabel('Number of Cluster')
plt.ylabel('Inertia')
```

Out[18]:
```
Text(0, 0.5, 'Inertia')
```



In [19]:
```
sil = []
for k in range(2,11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(norm_dm)
    sil.append(silhouette_score(norm_dm, kmeans.labels_))
```

In [20]:
```
plt.plot(range(2,11), sil, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Score")
plt.show()
```



In [21]:
```
final_kmeans = KMeans(n_clusters=3,random_state=42)
final_kmeans.fit(norm_dm)
```

Out[21]:
```
KMeans(n_clusters=3, random_state=42)
```

In [22]:
```
final_dm = pd.DataFrame(customer_dm, columns=customer_dm.columns, index=customer_dm.inde
final_dm['Cluster'] = final_kmeans.labels_ + 1 # I want to have cluster labels starting
final_dm.head(10)
```

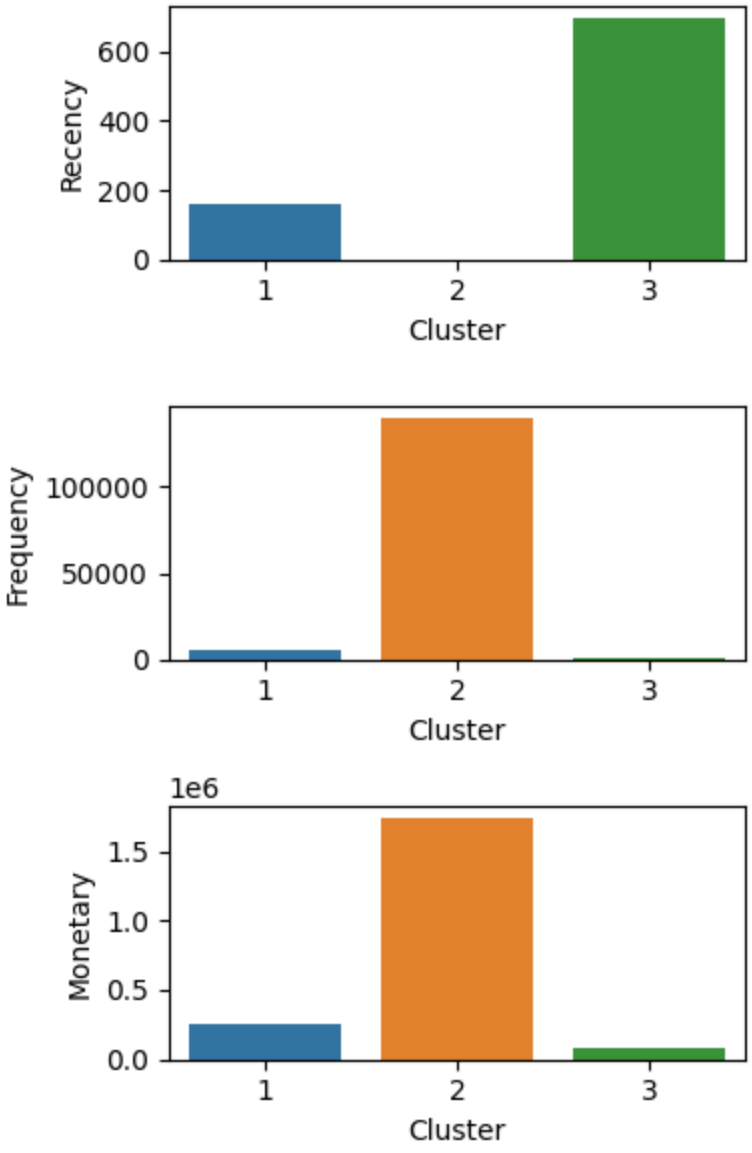
Out[22]:

CustomerID	Recency	Frequency	Monetary	Cluster
12346.0	326	1	77183.60	3
12747.0	23	103	4196.01	1
12748.0	4	4596	33719.73	1
12749.0	23	199	4090.88	1
12820.0	45	59	942.34	1
12821.0	96	6	92.72	1
12822.0	71	46	948.88	1
12823.0	75	5	1759.50	1
12824.0	30	25	397.12	1
12826.0	60	91	1474.72	1

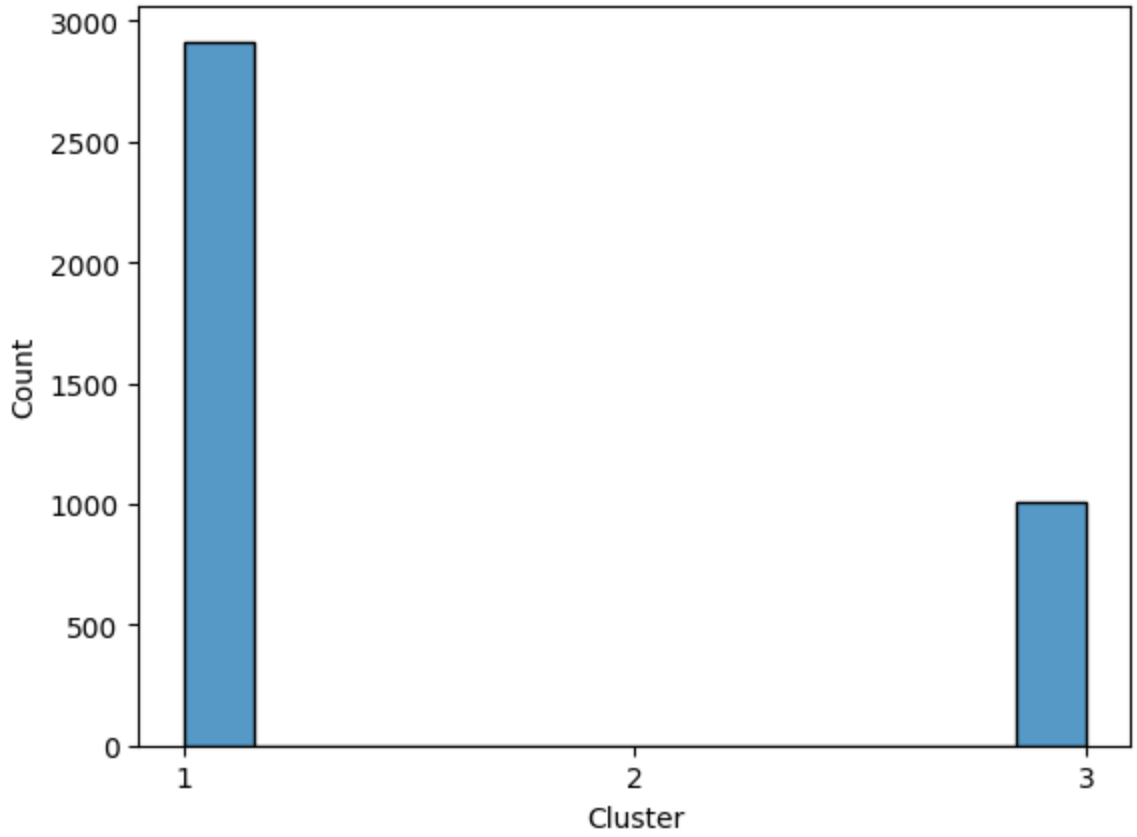
Now let's have a look at the characteristics of each

cluster, and see what customer insight we #can obtain and try to answer our questions.

In [29]:
```
Cluster_max =final_dm.groupby('Cluster')[['Recency', 'Frequency', 'Monetary']].max().res

fig, axes = plt.subplots(nrows=3, figsize=(4, 6))

sns.barplot(Cluster_max, x='Cluster', y='Recency', ax=axes[0])
sns.barplot(Cluster_max, x='Cluster', y='Frequency', ax=axes[1])
sns.barplot(Cluster_max, x='Cluster', y='Monetary', ax=axes[2])

plt.tight_layout()
```



In [37]:
```
sns.histplot(final_dm.Cluster)
plt.xticks(range(1,4))
plt.show()
```



In [38]:
```
final_dm['Cluster'].value_counts()
```

Out[38]:
```
1    2914
3    1006
2       1
Name: Cluster, dtype: int64
```

In [24]:
```
final_dm.groupby('Cluster').agg({'Monetary':'mean',
                                 'Frequency':'mean',
                                 'Recency':'mean'})
```

Out[24]:

Cluster	Monetary	Frequency	Recency
1	2.302895e+03	108.066232	49.788607
2	1.735698e+06	139788.000000	0.000000
3	5.534430e+02	31.404573	267.190855

Who is our loyal customers, and who is leaving us?

README.md file