In [1]:
```python
import pandas as pd
```

In [2]:
```python
trxn= pd.read_csv("C:/Retail_Data_Transactions.csv")
```

In [3]:
```python
trxn
```

Out[3]:

	customer_id	trans_date	tran_amount
0	CS5295	11-Feb-13	35
1	CS4768	15-Mar-15	39
2	CS2122	26-Feb-13	52
3	CS1217	16-Nov-11	99
4	CS1850	20-Nov-13	78
...	...	...	...
124995	CS8433	26-Jun-11	64
124996	CS7232	19-Aug-14	38
124997	CS8731	28-Nov-14	42
124998	CS8133	14-Dec-13	13
124999	CS7996	13-Dec-14	36

125000 rows × 3 columns

In [4]:
```python
respon= pd.read_csv("C:/Retail_Data_Response.csv")
respon
```

Out[4]:

	customer_id	response
0	CS1112	0
1	CS1113	0
2	CS1114	1
3	CS1115	1
4	CS1116	1
...	...	...
6879	CS8996	0
6880	CS8997	0
6881	CS8998	0
6882	CS8999	0
6883	CS9000	0

6884 rows × 2 columns

In [5]:
```python
mergedata=trxn.merge(respon, on='customer_id', how='left')
mergedata
```

Out[5]:

	customer_id	trans_date	tran_amount	response
0	CS5295	11-Feb-13	35	1.0
1	CS4768	15-Mar-15	39	1.0
2	CS2122	26-Feb-13	52	0.0
3	CS1217	16-Nov-11	99	0.0
4	CS1850	20-Nov-13	78	0.0
...	...	...	...	...
124995	CS8433	26-Jun-11	64	0.0
124996	CS7232	19-Aug-14	38	0.0
124997	CS8731	28-Nov-14	42	0.0
124998	CS8133	14-Dec-13	13	0.0
124999	CS7996	13-Dec-14	36	0.0

125000 rows × 4 columns

In [6]:
```python
mergedata.dtypes
```

Out[6]:
```
customer_id     object
trans_date      object
tran_amount      int64
response       float64
dtype: object
```

In [7]:
```python
mergedata.shape
```

Out[7]:
```
(125000, 4)
```

In [8]:
```python
mergedata.head()
```

Out[8]:

	customer_id	trans_date	tran_amount	response
0	CS5295	11-Feb-13	35	1.0
1	CS4768	15-Mar-15	39	1.0
2	CS2122	26-Feb-13	52	0.0
3	CS1217	16-Nov-11	99	0.0
4	CS1850	20-Nov-13	78	0.0

In [9]:
```python
mergedata.tail()
```

Out[9]:

	customer_id	trans_date	tran_amount	response
124995	CS8433	26-Jun-11	64	0.0
124996	CS7232	19-Aug-14	38	0.0
124997	CS8731	28-Nov-14	42	0.0
124998	CS8133	14-Dec-13	13	0.0
124999	CS7996	13-Dec-14	36	0.0

In [10]:
```python
mergedata.describe()
```

Out[10]:

	tran_amount	response
count	125000.000000	124969.000000
mean	64.991912	0.110763
std	22.860006	0.313840
min	10.000000	0.000000
25%	47.000000	0.000000
50%	65.000000	0.000000
75%	83.000000	0.000000
max	105.000000	1.000000

In [11]:
```python
mergedata.isnull().sum()
```

Out[11]:
```
customer_id     0
trans_date      0
tran_amount     0
response       31
dtype: int64
```

In [12]:
```python
mergedata=mergedata.dropna()  #droping duplicates
mergedata
```

Out[12]:

	customer_id	trans_date	tran_amount	response
0	CS5295	11-Feb-13	35	1.0
1	CS4768	15-Mar-15	39	1.0
2	CS2122	26-Feb-13	52	0.0
3	CS1217	16-Nov-11	99	0.0
4	CS1850	20-Nov-13	78	0.0
...	...	...	...	...
124995	CS8433	26-Jun-11	64	0.0
124996	CS7232	19-Aug-14	38	0.0
124997	CS8731	28-Nov-14	42	0.0
124998	CS8133	14-Dec-13	13	0.0
124999	CS7996	13-Dec-14	36	0.0

124969 rows × 4 columns

In [13]:
```python
#changing data types
mergedata['trans_date']=pd.to_datetime(mergedata['trans_date'])
mergedata
```

```
C:\Users\test\AppData\Local\Temp\ipykernel_15344\855538845.py:2: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsin
g is consistent and as-expected, please specify a format.
  mergedata['trans_date']=pd.to_datetime(mergedata['trans_date'])
C:\Users\test\AppData\Local\Temp\ipykernel_15344\855538845.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mergedata['trans_date']=pd.to_datetime(mergedata['trans_date'])
```

Out[13]:

	customer_id	trans_date	tran_amount	response
0	CS5295	2013-02-11	35	1.0
1	CS4768	2015-03-15	39	1.0
2	CS2122	2013-02-26	52	0.0
3	CS1217	2011-11-16	99	0.0
4	CS1850	2013-11-20	78	0.0
...	...	...	...	...
124995	CS8433	2011-06-26	64	0.0
124996	CS7232	2014-08-19	38	0.0
124997	CS8731	2014-11-28	42	0.0
124998	CS8133	2013-12-14	13	0.0
124999	CS7996	2014-12-13	36	0.0

124969 rows × 4 columns

In [14]:
```python
mergedata['response']=mergedata['response'].astype('int64')
mergedata
```

```
C:\Users\test\AppData\Local\Temp\ipykernel_15344\1751815077.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mergedata['response']=mergedata['response'].astype('int64')
```

Out[14]:

	customer_id	trans_date	tran_amount	response
0	CS5295	2013-02-11	35	1
1	CS4768	2015-03-15	39	1
2	CS2122	2013-02-26	52	0
3	CS1217	2011-11-16	99	0
4	CS1850	2013-11-20	78	0
...	...	...	...	...
124995	CS8433	2011-06-26	64	0
124996	CS7232	2014-08-19	38	0
124997	CS8731	2014-11-28	42	0
124998	CS8133	2013-12-14	13	0
124999	CS7996	2014-12-13	36	0

124969 rows × 4 columns

In [15]:
```python
set(mergedata['response'])
```

Out[15]:
```
{0, 1}
```

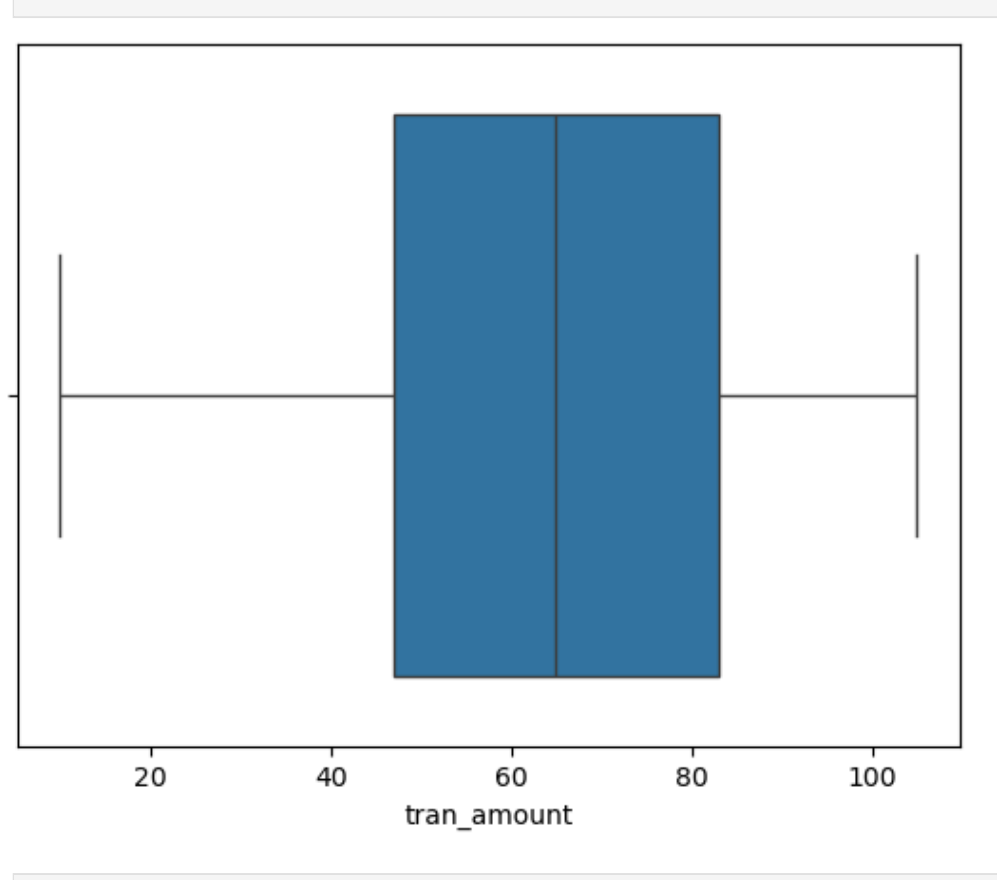
In [16]:
```python
mergedata.dtypes
```

Out[16]:
```
customer_id            object
trans_date     datetime64[ns]
tran_amount             int64
response                int64
dtype: object
```

In [17]:
```python
#check for outliers
#z-score

from scipy import stats
import numpy as np

#calc z-score
z_score=np.abs(stats.zscore(mergedata['tran_amount']))

#set threshold
threshold=3

outliers=z_score>threshold

print(mergedata[outliers])
```

```
Empty DataFrame
Columns: [customer_id, trans_date, tran_amount, response]
Index: []
```

In [18]:
```python
#box plot
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=mergedata['tran_amount'])
plt.show()
```



In [19]:
```python
#creating new column
mergedata['month']=mergedata['trans_date'].dt.month
mergedata
```

```
C:\Users\test\AppData\Local\Temp\ipykernel_15344\2998208910.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mergedata['month']=mergedata['trans_date'].dt.month
```

Out[19]:

	customer_id	trans_date	tran_amount	response	month
0	CS5295	2013-02-11	35	1	2
1	CS4768	2015-03-15	39	1	3
2	CS2122	2013-02-26	52	0	2
3	CS1217	2011-11-16	99	0	11
4	CS1850	2013-11-20	78	0	11
...	...	...	...	...	...
124995	CS8433	2011-06-26	64	0	6
124996	CS7232	2014-08-19	38	0	8
124997	CS8731	2014-11-28	42	0	11
124998	CS8133	2013-12-14	13	0	12
124999	CS7996	2014-12-13	36	0	12

124969 rows × 5 columns

In [20]:
```python
# highest transactions amount
monthly_sales=mergedata.groupby('month')['tran_amount'].sum()
monthly_sales=monthly_sales.sort_values(ascending=False).reset_index()
monthly_sales
```

Out[20]:

	month	tran_amount
0	8	726775
1	10	725058
2	1	724089
3	7	717011
4	12	709795
5	11	698024
6	6	697014
7	9	694201
8	2	645028
9	3	636475
10	5	633162
11	4	515746

In [21]:
```python
#which 3 months have had highest transactions amount
monthly_sales=mergedata.groupby('month')['tran_amount'].sum()
monthly_sales=monthly_sales.sort_values(ascending=False).reset_index().head(3)
monthly_sales
```

Out[21]:

	month	tran_amount
0	8	726775
1	10	725058
2	1	724089

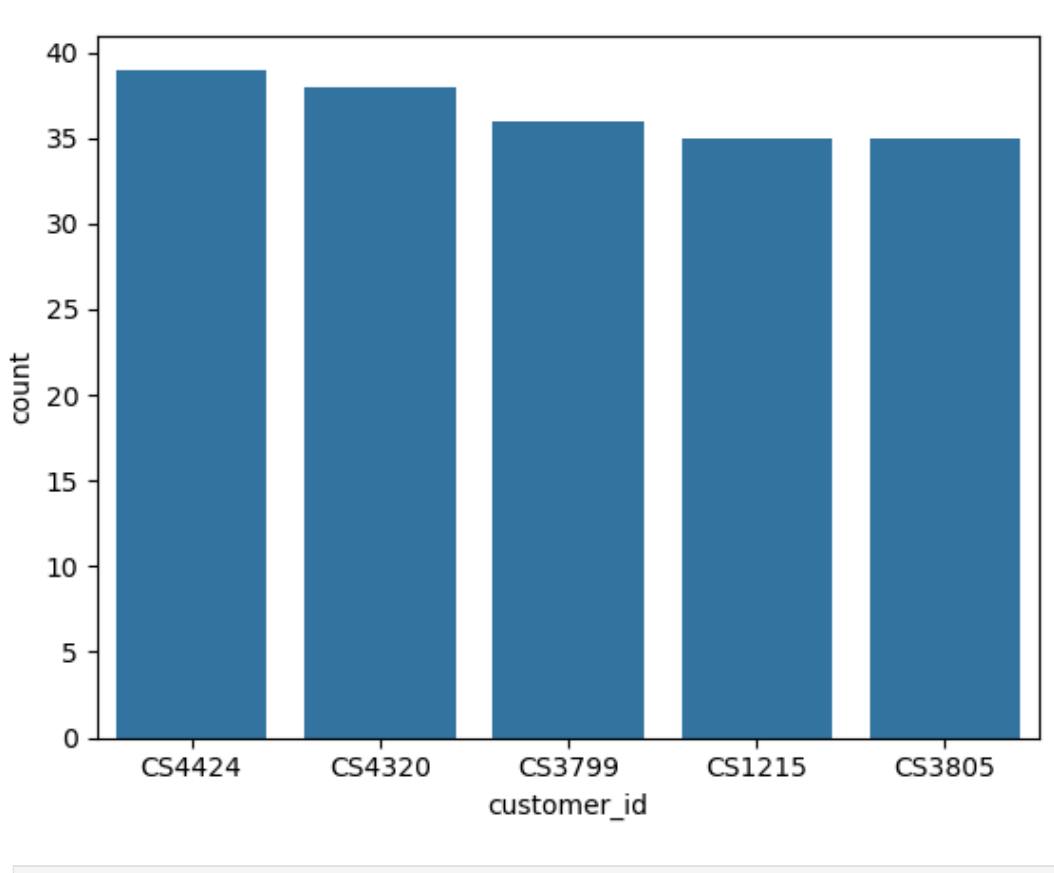
In [22]:
```python
# customers having highest number of orders
cust_count=mergedata['customer_id'].value_counts().reset_index()

#sort
top5_cust=cust_count.sort_values(by='count', ascending=False).head(5)
top5_cust
```

Out[22]:

	customer_id	count
0	CS4424	39
1	CS4320	38
2	CS3799	36
4	CS1215	35
3	CS3805	35

In [23]:
```python
#ploting barplot
sns.barplot(x='customer_id', y='count', data=top5_cust)
```

Out[23]:
```
<Axes: xlabel='customer_id', ylabel='count'>
```



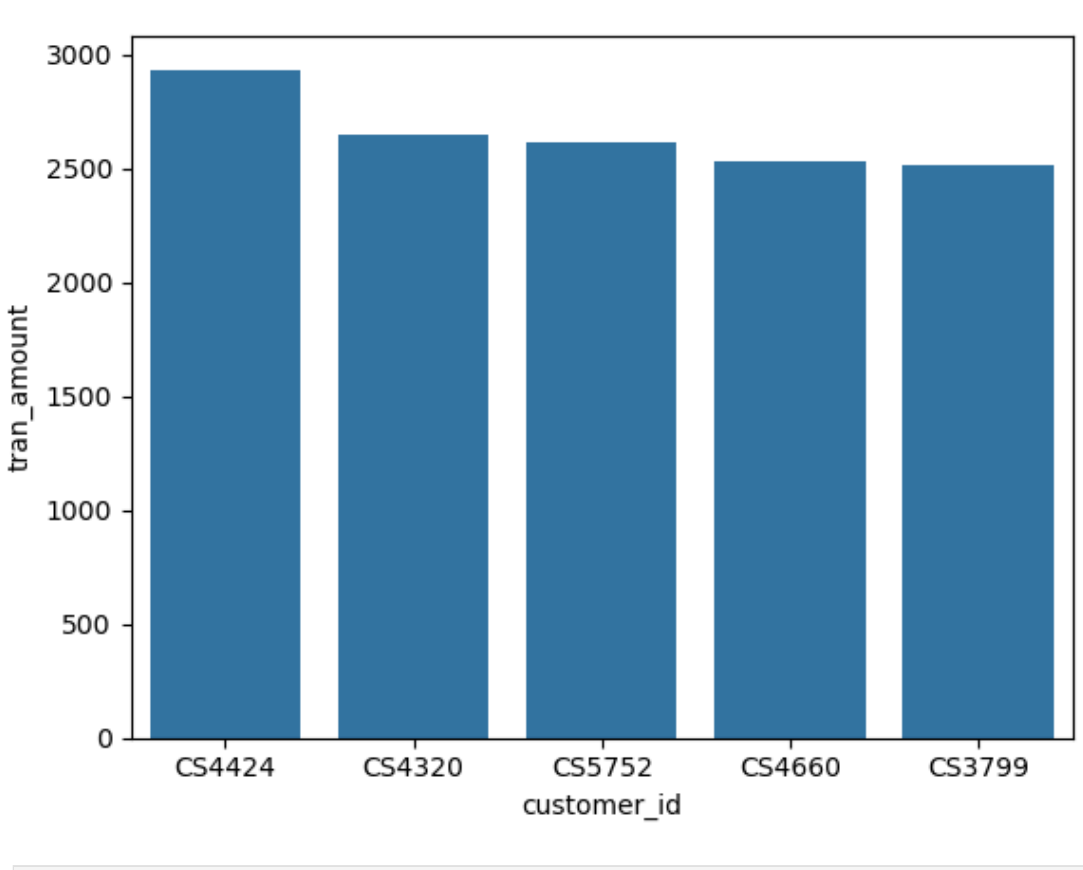
In [24]:
```python
#customers having highest transaction value
cust_sales=mergedata.groupby('customer_id')['tran_amount'].sum().reset_index()
cust_sales
#sort
top5_sales=cust_sales.sort_values(by='tran_amount', ascending=False).head(5)
top5_sales
```

Out[24]:

	customer_id	tran_amount
3312	CS4424	2933
3208	CS4320	2647
4640	CS5752	2612
3548	CS4660	2527
2687	CS3799	2513

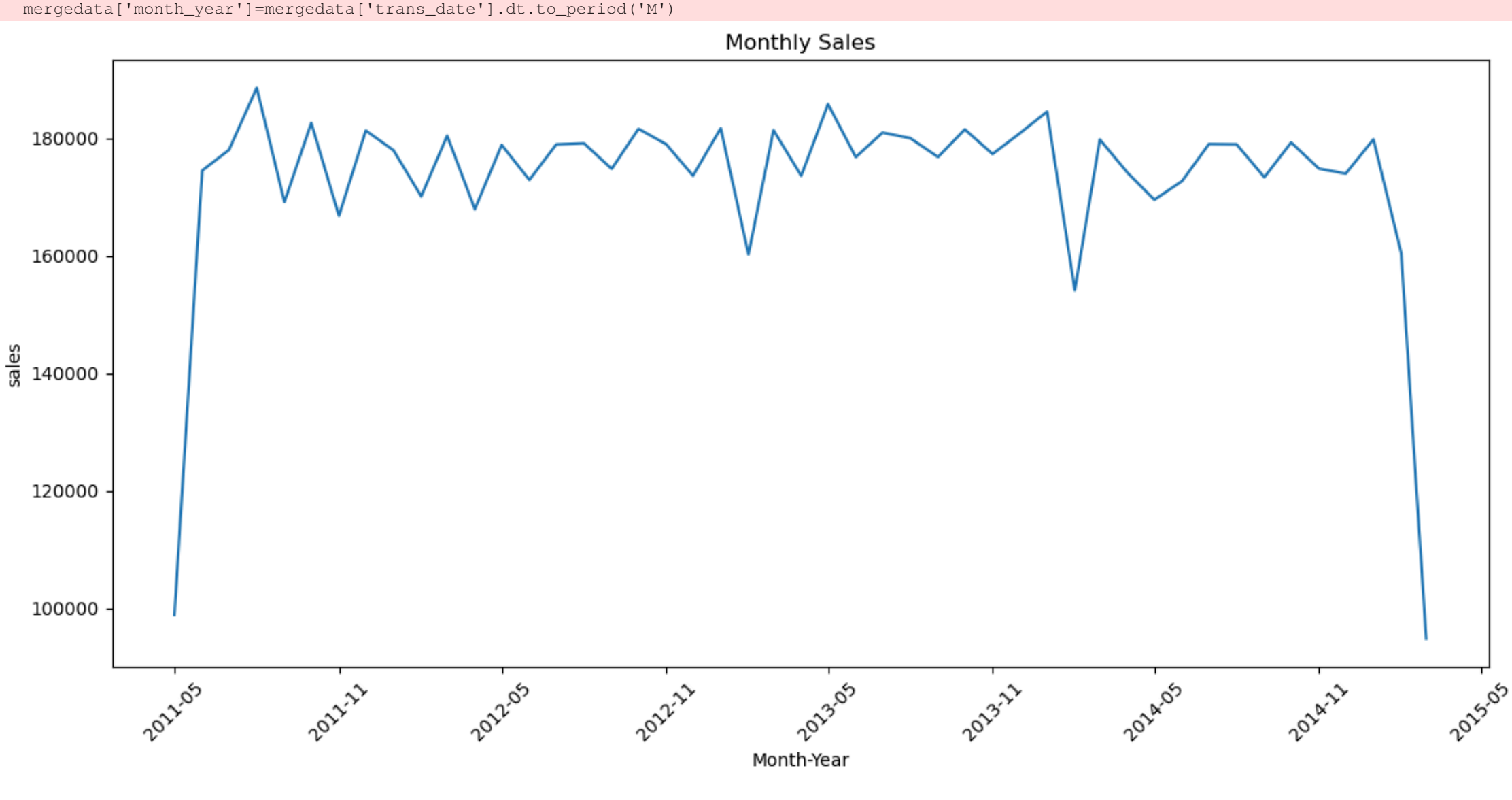
In [25]:
```python
#ploting barplot
sns.barplot(x='customer_id', y='tran_amount', data=top5_sales)
```

Out[25]:
```
<Axes: xlabel='customer_id', ylabel='tran_amount'>
```



In [26]:
```python
#Advanced analytics

#Time series analysis
import matplotlib.dates as mdates
mergedata['month_year']=mergedata['trans_date'].dt.to_period('M')
monthly_sales=mergedata.groupby('month_year')['tran_amount'].sum()

monthly_sales.index=monthly_sales.index.to_timestamp()

plt.figure(figsize=(12,6))
plt.plot(monthly_sales.index, monthly_sales.values)

plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=6))
plt.xlabel('Month-Year')
plt.ylabel('sales')
plt.title('Monthly Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
C:\Users\test\AppData\Local\Temp\ipykernel_15344\1271101556.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mergedata['month_year']=mergedata['trans_date'].dt.to_period('M')
```



In [27]:
```python
#Cohort segementation

#recency
recency=mergedata.groupby('customer_id')['trans_date'].max()

#frequency
frequency=mergedata.groupby('customer_id')['trans_date'].count()

#Monetory
monetory=mergedata.groupby('customer_id')['tran_amount'].sum()

#Combine
rfm= pd.DataFrame({'recency':recency, 'frequency':frequency, 'monetory':monetory})

rfm
```

Out[27]:

	recency	frequency	monetory
customer_id			
CS1112	2015-01-14	15	1012
CS1113	2015-02-09	20	1490
CS1114	2015-02-12	19	1432
CS1115	2015-03-05	22	1659
CS1116	2014-08-25	13	857
...	...	...	...
CS8996	2014-12-09	13	582
CS8997	2014-06-28	14	543
CS8998	2014-12-22	13	624
CS8999	2014-07-02	12	383
CS9000	2015-02-28	13	533

6884 rows × 3 columns

In [28]:
```python
def segment_customer(row):
    if row['recency'].year>=2012 and row['frequency']>=15 and row['monetory']>1000:
        return 'P0'
    elif (2011<=row['recency'].year<2012) and (10<row['frequency']<15) and(500<=row['monetory']<=1000):
        return 'P1'
    else:
        return 'P2'

rfm['Segment']=rfm.apply(segment_customer, axis=1)

rfm
```

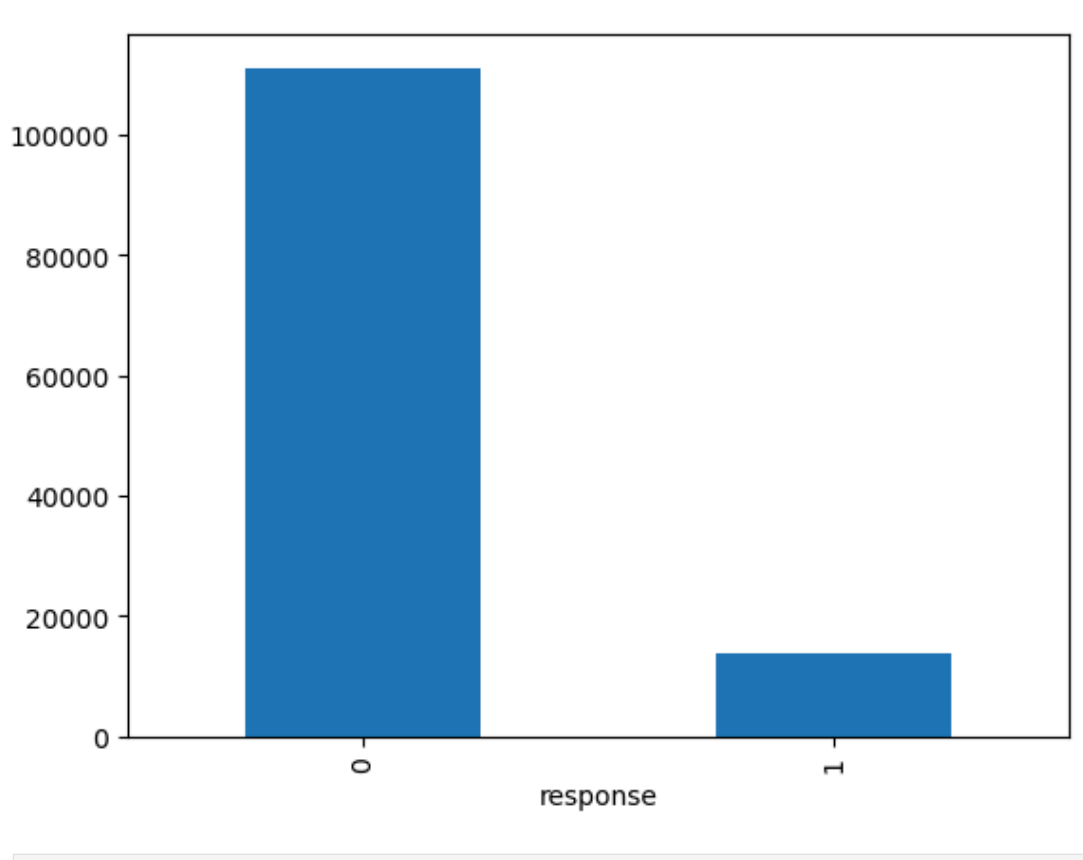
Out[28]:

	recency	frequency	monetory	Segment
customer_id				
CS1112	2015-01-14	15	1012	P0
CS1113	2015-02-09	20	1490	P0
CS1114	2015-02-12	19	1432	P0
CS1115	2015-03-05	22	1659	P0
CS1116	2014-08-25	13	857	P2
...	...	...	...	...
CS8996	2014-12-09	13	582	P2
CS8997	2014-06-28	14	543	P2
CS8998	2014-12-22	13	624	P2
CS8999	2014-07-02	12	383	P2
CS9000	2015-02-28	13	533	P2

6884 rows × 4 columns

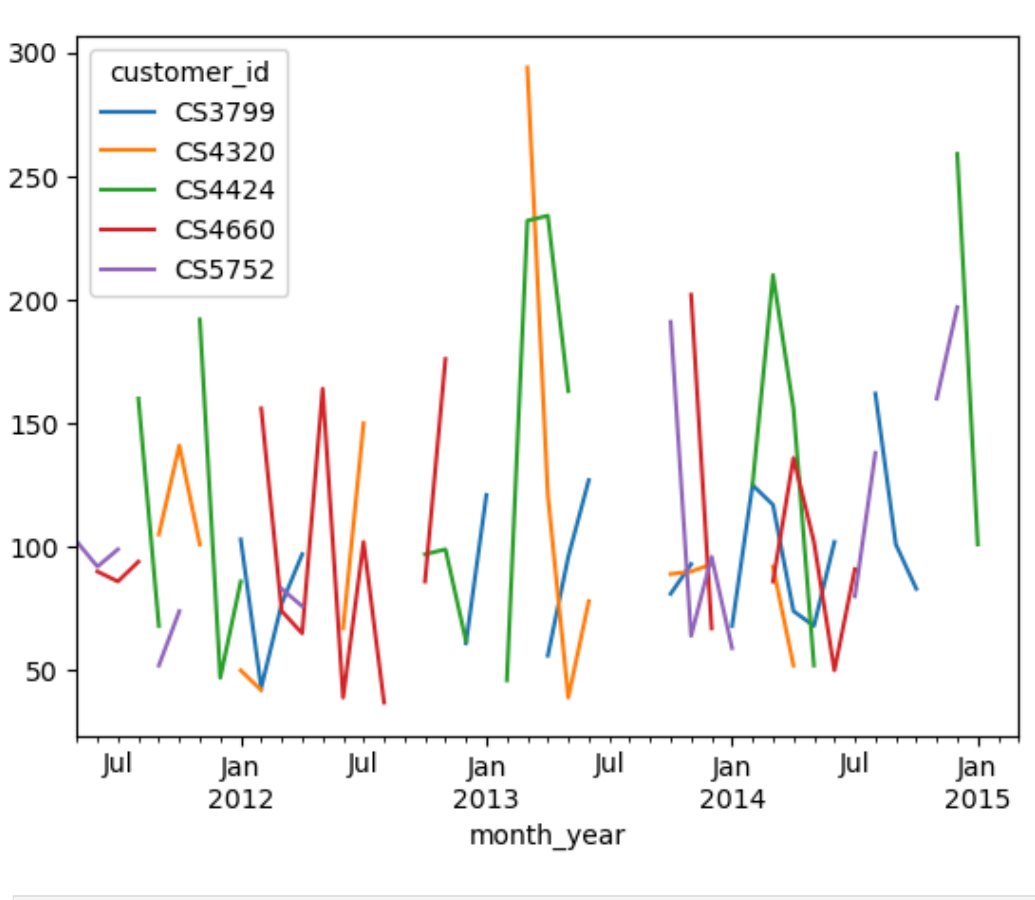
In [29]:
```python
#churn analysis
churn_counts=mergedata['response'].value_counts()
churn_counts.plot(kind='bar')
```

Out[29]:
```
<Axes: xlabel='response'>
```



In [30]:
```python
#analysing top customers

top5_cust=monetory.sort_values(ascending=False).head(5).index
top_customers_mergedata=mergedata[mergedata['customer_id'].isin(top5_cust)]
top_customer_sales=top_customers_mergedata.groupby(['customer_id','month_year'])['tran_amount'].sum().unstack(level=0)
top_customer_sales.plot(kind='line')
```

Out[30]:
```
<Axes: xlabel='month_year'>
```



In [31]:
```python
mergedata.to_csv('MainData.csv')
```

In [32]:
```python
rfm.to_csv('AddAnalyse.csv')
```