2024W AML 2103 2 Visualization for AI and ML

Assignment 1

Jayachandhran Saravanan (C0910392)

Inferernce and Results:

Q1:

- In this task, the function to create a quarterly summary for the given feature variable is created
- Extensive details on the transaction count, average, mean, and median values are represented
- Each service area performance is observed and the highest value of sum is "Children's Family Services 2016.0 Q4"
- most of the plots were unstable

Q2:

- the year 2016 saw the highest spike ever
- Regional Enterprise (service area has more than 5 spikes in the given 4 years)
- 12 out of 25 service areas have permanent (non-spike behaviour)
- percentage difference and z score on consecutive transactions are calculated

Q3:

- "AMAZON UK MARKETPLACE" creditor has the highest level of repeatation in account over 25 instances
- there are 700+ creditors mapped bettern 2-3 accounts

Q4:

- Using the spending behavior (transaction sum, average and count) the service area (25 in numbers) are clustered
- K means gave 3 clusters with unstable distribution with cluster1-3 cluster2-18 cluster3-2
- Using HAC, the data points are branched using the mixed linkage method, the clusters are displayed below

Q5:

- using simple standard deviation, the data points against time(week | consecutive) are checked for any abnormal behavior or sudden difference in patterns
- those are potential for observation to check for anomaly transaction
- Education service has more number of anomalies present

- The highest of 800% surge is given in an account --> which has a mean of 400(CAD/USD)
- various experimentations on the fraud/anomaly is carried out and attached in appendix

*** 12 reproducible functions are created for easy customization and automation**

*** Data set 3 is experimented but the analysis is not adding proper value so the null and non-useful values are removed**

*** below the script and outputs are attached**

since PDF function creates 170+ pages, tables are trauncated for easy go through. Plots and visualization are kept as per instruction

Importing required packages

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display, HTML
from matplotlib.ticker import FuncFormatter
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
```

loading the data | Pandas

In [2]:
```
#loading the dataset given
df1=pd.read_csv('data_1.csv')
df3=pd.read_csv('data_3.csv')
df4=pd.read_csv('data_4.csv')
df2 = pd.read_csv("data_2.txt")
# df3=pd.read_('data_3.txt')
```

In [3]:
```
#checking the given data

data=['df1','df2','df3','df4']
colm={}
```

```
for i,col in enumerate([df1,df2,df3,df4]):
    colm[data[i]]=set(col.columns)
```

Checking column names and data preprocessing experiments

In [4]:
```
colm
```

Out[4]:
```
{'df1': {'Account Description',
  'Creditor',
  'JV Date',
  'JV Reference',
  'JV Value',
  'Service Area',
  'Transaction Date'},
 'df2': {'Account Description',
  'Creditor',
  'Journal Date',
  'Journal Reference',
  'Service Area',
  'Total'},
 'df3': {'Account Description',
  'Creditor',
  'Journal Date',
  'Journal Reference',
  'Service Area',
  'Total'},
 'df4': {'FIN.INET CONVERSION',
  'FIN.ORIGINAL CURRENCY AMOUNT',
  'FIN.ORIGINAL ISO CURRENCY CODE SYMBOL',
  'FIN.POSTING DATE',
  'FIN.TRANSACTION AMOUNT',
  'FIN.TRANSACTION DATE',
  'MCH.CITY NAME',
  'MCH.MERCHANT NAME'}}
```

In [5]:
```
#since we can remove the JV Date from the first data
df1.drop(columns=['JV Date'],inplace=True)
```

In [6]:
```
df1.head()
```

Out[6]:

	Service Area	Account Description	Creditor	Transaction Date	JV Reference	JV Value
0	Childrens Services	IT Services	123-REG.CO.UK	23/04/2014	93	143.81
1	Childrens Services	Other Services	ACCESS EXPEDITIONS	03/04/2014	111	6,000.00
2	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	02/04/2014	6	309.38
3	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	02/04/2014	7	218.76
4	Childrens Services	Building Repairs & Maintenance	ALLSOP & FRANCIS	15/04/2014	381	306

In [7]:
```
df2.head()
```

Out[7]:

	Service Area	Account Description	Creditor	Journal Date	Journal Reference	Total
0	Assurance	Miscellaneous Expenses	43033820 COSTA COFFEE	18/08/2015	5043.0	2
1	Children's Family Services	Miscellaneous Expenses	99 PLUS DISCOUNT MART	08/06/2015	4184.0	29.97
2	Children's Family Services	E19 - Learning Resources	99P STORES LTD	07/12/2015	6278.0	34.65
3	Children's Family Services	Equipment and Materials Purcha	99P STORES LTD	18/08/2015	5041.0	10.72
4	Children's Family Services	Subsistence	CHOPSTIX00000000000	21/05/2015	5750.0	33.7

In [8]:
```
column_names=['Service Area','Account Description',
  'Creditor',
  'Transaction Date',
  'JV Reference',
  'JV Value' ]
```

In [9]:
```
for i,col in enumerate([df1,df2,df3]):
    col.set_axis(column_names,axis=1,inplace=True)
```

In [10]:
```
data=['df1','df2','df3']
colm={}

for i,col in enumerate([df1,df2,df3]):
    colm[data[i]]=set(col.columns)
```

In [11]:
```
set.intersection(*colm.values())#all the columns are common here
```

Out[11]:
```
{'Account Description',
 'Creditor',
 'JV Reference',
 'JV Value',
 'Service Area',
 'Transaction Date'}
```

In [12]:
```
df1.head()
```

Out[12]:

	Service Area	Account Description	Creditor	Transaction Date	JV Reference	JV Value
0	Childrens Services	IT Services	123-REG.CO.UK	23/04/2014	93	143.81
1	Childrens Services	Other Services	ACCESS EXPEDITIONS	03/04/2014	111	6,000.00
2	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	02/04/2014	6	309.38
3	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	02/04/2014	7	218.76
4	Childrens Services	Building Repairs & Maintenance	ALLSOP & FRANCIS	15/04/2014	381	306

In [13]:
```
df2.head()
```

Out[13]:

	Service Area	Account Description	Creditor	Transaction Date	JV Reference	JV Value
0	Assurance	Miscellaneous Expenses	43033820 COSTA COFFEE	18/08/2015	5043.0	2
1	Children's Family Services	Miscellaneous Expenses	99 PLUS DISCOUNT MART	08/06/2015	4184.0	29.97
2	Children's Family Services	E19 - Learning Resources	99P STORES LTD	07/12/2015	6278.0	34.65
3	Children's Family Services	Equipment and Materials Purcha	99P STORES LTD	18/08/2015	5041.0	10.72
4	Children's Family Services	Subsistence	CHOPSTIX00000000000	21/05/2015	5750.0	33.7

In [14]:
```
df=pd.concat([df1,df2,df3],ignore_index=True)
```

In [15]:
```
df.head()
```

Out[15]:

	Service Area	Account Description	Creditor	Transaction Date	JV Reference	JV Value
0	Childrens Services	IT Services	123-REG.CO.UK	23/04/2014	93.0	143.81
1	Childrens Services	Other Services	ACCESS EXPEDITIONS	03/04/2014	111.0	6,000.00
2	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	02/04/2014	6.0	309.38
3	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	02/04/2014	7.0	218.76
4	Childrens Services	Building Repairs & Maintenance	ALLSOP & FRANCIS	15/04/2014	381.0	306

In [16]:
```
df.shape[0]==sum([df1.shape[0],df2.shape[0],df3.shape[0]])
```

Out[16]:
```
True
```

EDA

In [17]:
```
#getting the details of the combined data set

def get_data_info(data):
    print("shape of the dataframe: ",data.shape,"\n","="*50)
    print("size of the data: ",data.size,"\n","="*50)
    print("duplicate values present: ",data.duplicated().sum(),"\n","="*50)
    print("missing values %: \n",data.isnull().mean()*100,"\n","="*50)
    print("unique values present in the data: \n",data.nunique(),"\n","="*50)
```

In [18]:
```
get_data_info(df)
```

```
shape of the dataframe:  (12589, 6)
 ================================================
size of the data:  75534
 ================================================
duplicate values present:  0
 ================================================
missing values %:
 Service Area           0.007943
Account Description    0.015887
Creditor               0.015887
Transaction Date       0.015887
JV Reference           0.015887
JV Value               0.000000
dtype: float64
 ================================================
unique values present in the data:
 Service Area              25
Account Description       67
Creditor                1936
Transaction Date         739
JV Reference           10742
JV Value                6292
dtype: int64
 ================================================
```

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12589 entries, 0 to 12588
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Service Area         12588 non-null  object 
 1   Account Description  12587 non-null  object 
 2   Creditor             12587 non-null  object 
 3   Transaction Date     12587 non-null  object 
 4   JV Reference         12587 non-null  float64
 5   JV Value             12589 non-null  object 
dtypes: float64(1), object(5)
memory usage: 590.2+ KB
```

In [20]:
```
#the values of jv value is in object/string type

df['JV Value']=df['JV Value'].apply(lambda x:(''.join(re.findall(r"\d+",x)) ))
df['JV Value']=df['JV Value'].astype(float)
```

In [21]:
```
df['JV Value'].dtype #mission success
```

Out[21]: dtype('float64')

In [22]:
```
df['Transaction Date']=pd.to_datetime(df['Transaction Date'])
df['date']=df['Transaction Date'].dt.day
```

In [23]:
```
df['month']=df['Transaction Date'].dt.month
```

In [24]:
```
df['year']=df['Transaction Date'].dt.year
df['QT']=df['Transaction Date'].dt.quarter
df['QT_validation']=df['Transaction Date'].dt.to_period('Q')

df['final_qr'] = pd.PeriodIndex(df['Transaction Date'], freq='Q')
```

In [25]:
```
df['Quarter'] = df['QT'].map(lambda x: "Q{}".format(x)).str[:-2]
```

In [26]:
```
df.head()
```

Out[26]:

	Service Area	Account Description	Creditor	Transaction Date	JV Reference	JV Value	date	month	year	QT	QT_validation	final_qr	Quarter
0	Childrens Services	IT Services	123-REG.CO.UK	2014-04-23	93.0	14381.0	23.0	4.0	2014.0	2.0	2014Q2	2014Q2	Q2
1	Childrens Services	Other Services	ACCESS EXPEDITIONS	2014-03-04	111.0	600000.0	4.0	3.0	2014.0	1.0	2014Q1	2014Q1	Q1
2	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	2014-02-04	6.0	30938.0	4.0	2.0	2014.0	1.0	2014Q1	2014Q1	Q1
3	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	2014-02-04	7.0	21876.0	4.0	2.0	2014.0	1.0	2014Q1	2014Q1	Q1
4	Childrens Services	Building Repairs & Maintenance	ALLSOP & FRANCIS	2014-04-15	381.0	306.0	15.0	4.0	2014.0	2.0	2014Q2	2014Q2	Q2

In [27]:
```
df.Quarter.value_counts()
```

Out[27]:
```
Q1    3272
Q3    3216
Q4    3168
Q2    2931
Qn       2
Name: Quarter, dtype: int64
```

In [28]:
```
get_data_info(df)
```

```
shape of the dataframe:  (12589, 13)
 ================================================
size of the data:  163657
 ================================================
duplicate values present:  0
 ================================================
missing values %:
 Service Area           0.007943
Account Description    0.015887
Creditor               0.015887
Transaction Date       0.015887
JV Reference           0.015887
JV Value               0.000000
date                   0.015887
month                  0.015887
year                   0.015887
QT                     0.015887
QT_validation          0.015887
final_qr               0.015887
Quarter                0.000000
dtype: float64
 ================================================
unique values present in the data:
 Service Area              25
Account Description       67
Creditor                1936
Transaction Date         739
JV Reference           10742
JV Value                5353
date                      31
month                     12
year                       4
QT                         4
QT_validation             16
final_qr                  16
Quarter                    5
dtype: int64
 ================================================
```

In [29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12589 entries, 0 to 12588
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Service Area         12588 non-null  object
 1   Account Description  12587 non-null  object
 2   Creditor             12587 non-null  object
 3   Transaction Date     12587 non-null  datetime64[ns]
 4   JV Reference         12587 non-null  float64
 5   JV Value             12589 non-null  float64
 6   date                 12587 non-null  float64
 7   month                12587 non-null  float64
 8   year                 12587 non-null  float64
 9   QT                   12587 non-null  float64
 10  QT_validation        12587 non-null  period[Q-DEC]
 11  final_qr             12587 non-null  period[Q-DEC]
 12  Quarter              12589 non-null  object
dtypes: datetime64[ns](1), float64(6), object(4), period[Q-DEC](2)
memory usage: 1.2+ MB
```

In [30]:
```
df.describe().T
```

Out[30]:

	count	mean	std	min	25%	50%	75%	max
JV Reference	12587.0	6126.846906	3248.967457	1.0	3943.5	5795.0	8847.5	12136.0
JV Value	12589.0	14626.548495	541065.815394	0.0	298.0	1600.0	5825.0	47104401.0
date	12587.0	15.853976	8.456099	1.0	9.0	16.0	23.0	31.0
month	12587.0	6.467625	3.489941	1.0	3.0	7.0	10.0	12.0
year	12587.0	2015.268610	0.950082	2014.0	2014.0	2015.0	2016.0	2017.0
QT	12587.0	2.498927	1.128441	1.0	1.0	3.0	4.0	4.0

In [31]:
```
df['QT_validation'].value_counts()
```

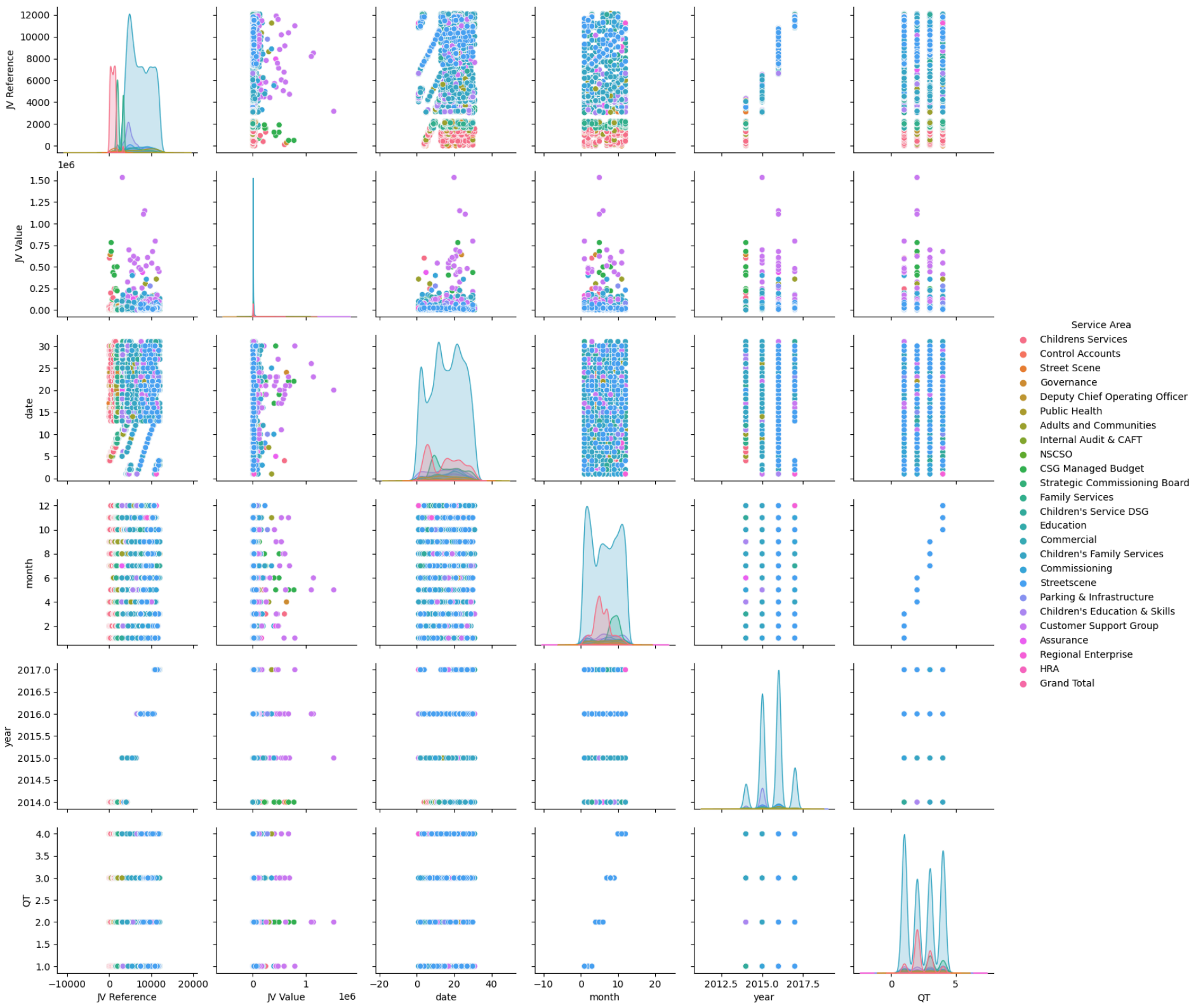
Out[31]:
```
2016Q3    1098
2016Q1    1091
2015Q3    1087
2016Q4    1072
2014Q4    1057
2016Q2    1006
2015Q4     974
2014Q2     933
2014Q3     923
2017Q1     919
2015Q1     917
2015Q2     898
2014Q1     345
2017Q3     108
2017Q2      94
2017Q4      65
Freq: Q-DEC, Name: QT_validation, dtype: int64
```

In [32]:
```
df['QT_validation'].unique()
```

Out[32]:
```
<PeriodArray>
['2014Q2', '2014Q1', '2014Q4', '2014Q3', '2015Q1', '2015Q3', '2015Q2',
 '2015Q4', '2016Q1', '2016Q3', '2016Q4', '2016Q2',    'NaT', '2017Q1',
 '2017Q4', '2017Q2', '2017Q3']
Length: 17, dtype: period[Q-DEC]
```

understanding the data distribution

In [33]:
```
#seperate the plot granular by passing the values clearly
sns.pairplot(df,hue='Service Area')
plt.show()
```


JV Reference
JV Value
1e6
date
month
year
QT
Service Area
Childrens Services
Control Accounts
Street Scene
Governance
Deputy Chief Operating Officer
Public Health
Adults and Communities
Internal Audit & CAFT
NSCSO
CSG Managed Budget
Strategic Commissioning Board
Family Services
Children's Service DSG
Education
Commercial
Children's Family Services
Commissioning
Streetscene
Parking & Infrastructure
Children's Education & Skills
Customer Support Group
Assurance
Regional Enterprise
HRA
Grand Total

TASK 1

In [ ]:
```
#understanding the service area presentin the data

display(df['Service Area'].unique())
print("total_unique service area",df['Service Area'].nunique())
display(HTML("<hr>"))
display(df['Service Area'].value_counts())
```

In [ ]:
```
temp = df.groupby('Service Area')
for i in temp:
    display(i)
```

group data for total,avg,most common value of the transaction by service area, then accounts (year,quater)

In [36]:
```
def summary_table(data,group,value):
    """
    arg: data--> dataframe given
    arg: group--> list of columns to be group by
    arg: value--> column where stats are need to be calcualted

    func: grouping the df by the columns mentioned

    return: dataframe confined by the parameters passed by
    """
    output=df.groupby(group)[value].agg(['count', 'mean', 'median','sum']).reset_index()
    print("created a dataframe with columns grouped by: {}\n values are aggregated by: {}".format(group,value))

    return output
```

In [37]:
```
sa_year_qrt=summary_table(df,list(('Service Area', 'year', 'Quarter')),'JV Value')
sa_year=summary_table(df,list(('Service Area', 'year')),'JV Value')
```

```
created a dataframe with columns grouped by: ['Service Area', 'year', 'Quarter']
 values are aggregated by: JV Value
created a dataframe with columns grouped by: ['Service Area', 'year']
 values are aggregated by: JV Value
```

In [38]:
```
ad_year_qt=summary_table(df,list(('Account Description', 'year', 'Quarter')),'JV Value')
ad_year=summary_table(df,list(('Account Description', 'year')),'JV Value')
```

```
created a dataframe with columns grouped by: ['Account Description', 'year', 'Quarter']
 values are aggregated by: JV Value
created a dataframe with columns grouped by: ['Account Description', 'year']
 values are aggregated by: JV Value
```

In [39]:
```
#summary view of the transactions for each Service Area
```

In [40]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Question 1 part 2

summary table -I

In [89]:
```
#displaying the created dataframe through groupby function for the second part of the question
sa_year_qrt[:10].T
```

Out[89]:

	0	1	2	3	4	5	6	7	8	
Service Area	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults ar Communiti
year	2014.0	2014.0	2014.0	2014.0	2015.0	2015.0	2015.0	2015.0	2016.0	2016
Quarter	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	C
count	2	15	11	14	7	10	19	17	23	3
mean	225.5	4793.4	228.545455	3502.0	1044.285714	14753.0	3990.526316	1470.0	8437.130435	19754.3421(
median	225.5	207.0	124.0	198.0	268.0	205.0	164.0	1667.0	3247.0	2673
sum	451.0	71901.0	2514.0	49028.0	7310.0	147530.0	75820.0	24990.0	194054.0	750665

due to the pdf pages are going till 200, i have to cut it short to 10 records

Question 1 part 2

summary table -I

In [90]:
```
ad_year_qt[:10].T
```

Out[90]:

	0	1	2	3	4	5	6	7	8	9
Account Description	Advertising	Advertising	Advertising	Advertising	Advertising	Advertising	Advertising	Advertising for staff	Books-CDs-Audio-Video	Books-CDs-Audio-Video
year	2014.0	2014.0	2014.0	2015.0	2015.0	2015.0	2016.0	2016.0	2014.0	2014.0
Quarter	Q2	Q3	Q4	Q1	Q2	Q4	Q4	Q2	Q1	Q2
count	8	3	5	2	2	2	4	1	25	50
mean	4604.0	5825.0	4659.4	5825.0	2927.0	115110.0	47692.5	45000.0	1547.36	5371.68
median	5216.0	5825.0	5825.0	5825.0	2927.0	115110.0	60000.0	45000.0	411.0	1513.5
sum	36832.0	17475.0	23297.0	11650.0	5854.0	230220.0	190770.0	45000.0	38684.0	268584.0

In [43]:
```
#experimenting with the tables to create subset of the dataframe for plotting
```

In [91]:
```
sa_year_qrt[sa_year_qrt['Service Area']=="Adults and Communities"][['year','Quarter','count']].T
```

Out[91]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
year	2014.0	2014.0	2014.0	2014.0	2015.0	2015.0	2015.0	2015.0	2016.0	2016.0	2016.0	2016.0	2017.0	2017.0	2017.0	2017.0
Quarter	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
count	2	15	11	14	7	10	19	17	23	38	33	36	37	6	7	3

In [92]:
```
sa_year_qrt[sa_year_qrt['Service Area']=="Adults and Communities"][['year','Quarter','sum']].T
```

Out[92]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	1
year	2014.0	2014.0	2014.0	2014.0	2015.0	2015.0	2015.0	2015.0	2016.0	2016.0	2016.0	2016.0	2017.0	2017.0	2017
Quarter	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q
sum	451.0	71901.0	2514.0	49028.0	7310.0	147530.0	75820.0	24990.0	194054.0	750665.0	480493.0	667573.0	500621.0	15934.0	48387

In [46]:
```
def plot_q1(data,col,val):
    """
    arg: data --> dataframe passed to get the plot
    arg: col --> column for which individual plots are need to be created
    arg: val --> what kind of observation needed (mean,sum,count)
    fun: plot bar chart
    return None
    """
    count=0
    col_unique_values=data[col].unique()
    for i in col_unique_values:
        plot_df=data[data[col]==i][['year','Quarter',val]]
        count+=1
        plt.figure(figsize=(14,6))
        sns.barplot(data=plot_df,x='year',y=val,hue='Quarter')
        plt.title("{} of Transcation for {}".format(val,col))
        plt.xlabel("YEAR with unstacked Quarters")
        plt.ylabel(val)
        plt.tight_layout()
        plt.show()
```

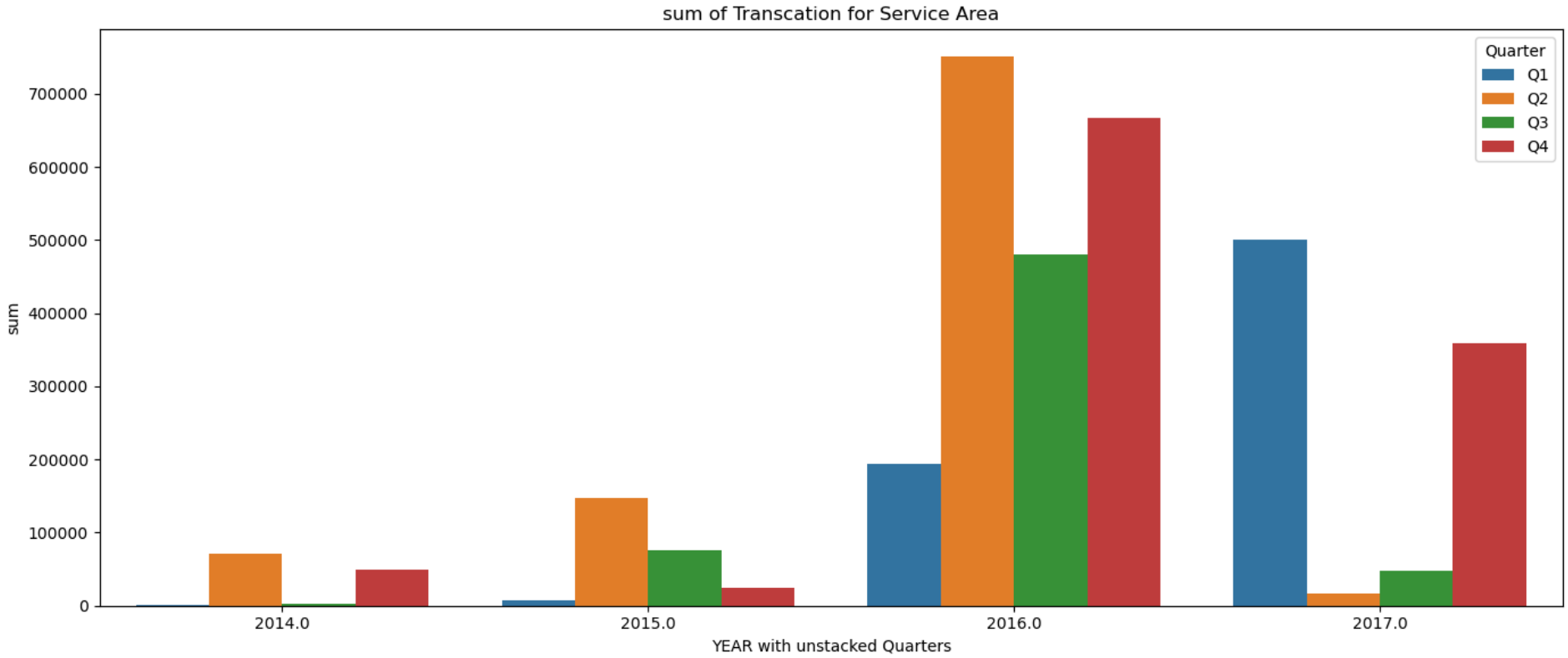
In [47]:
```
def plot_q1a(data,col,val):
    """
    arg: data --> dataframe passed to get the plot
    arg: col --> column for summary creation
    arg: val --> column to apply sum
    fun: plot summary bar chart
    return None
    """
    summary_df = data.groupby([col, 'year', 'Quarter'])[val].sum().reset_index()
    plt.figure(figsize=(12, 8))
    a=sns.barplot(data=summary_df, x='Quarter', y=val, hue='year',errorbar=None)
    plt.title('Summary Plot: {}'.format(col))
    plt.xlabel('Quarters')
    plt.ylabel("summation of {}".format(val))
    a.yaxis.set_major_formatter(FuncFormatter(lambda x, _: format(int(x), ',')))
```

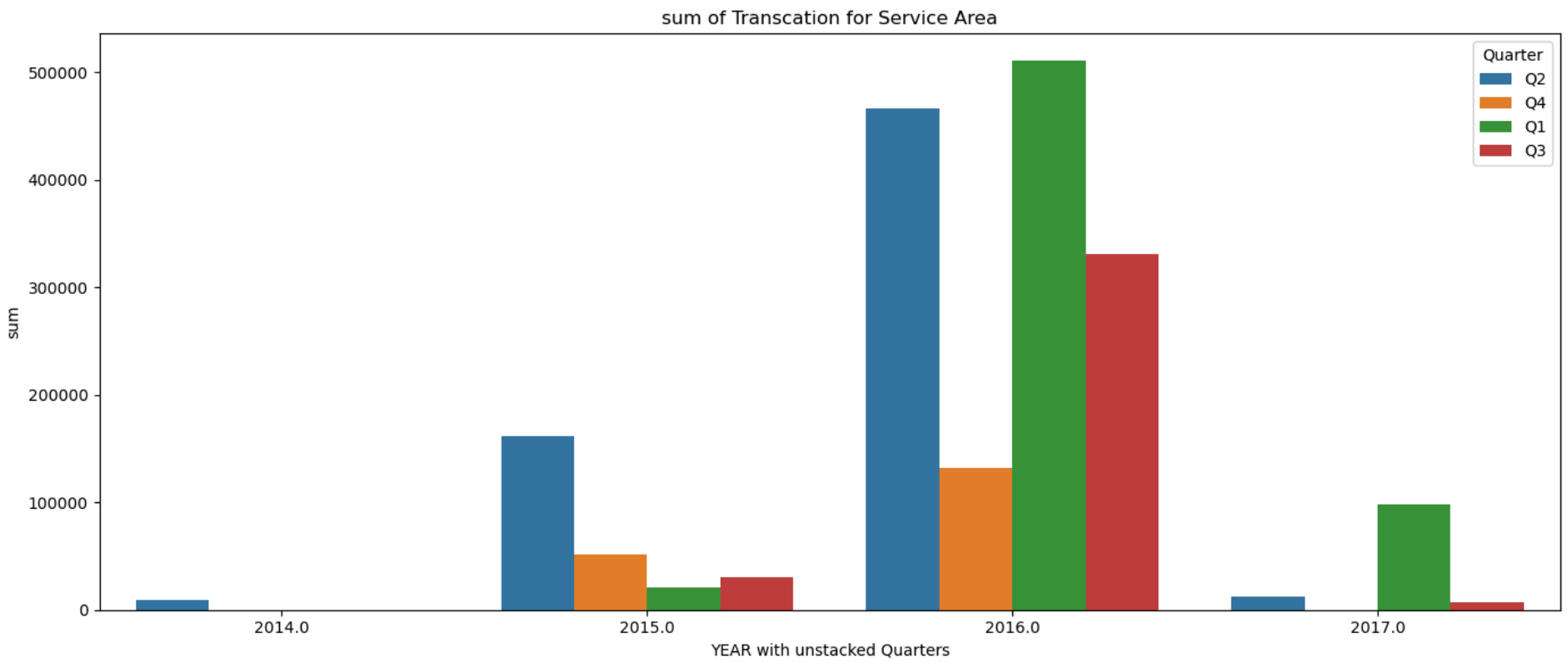
```
    plt.tight_layout()
    plt.show()
```

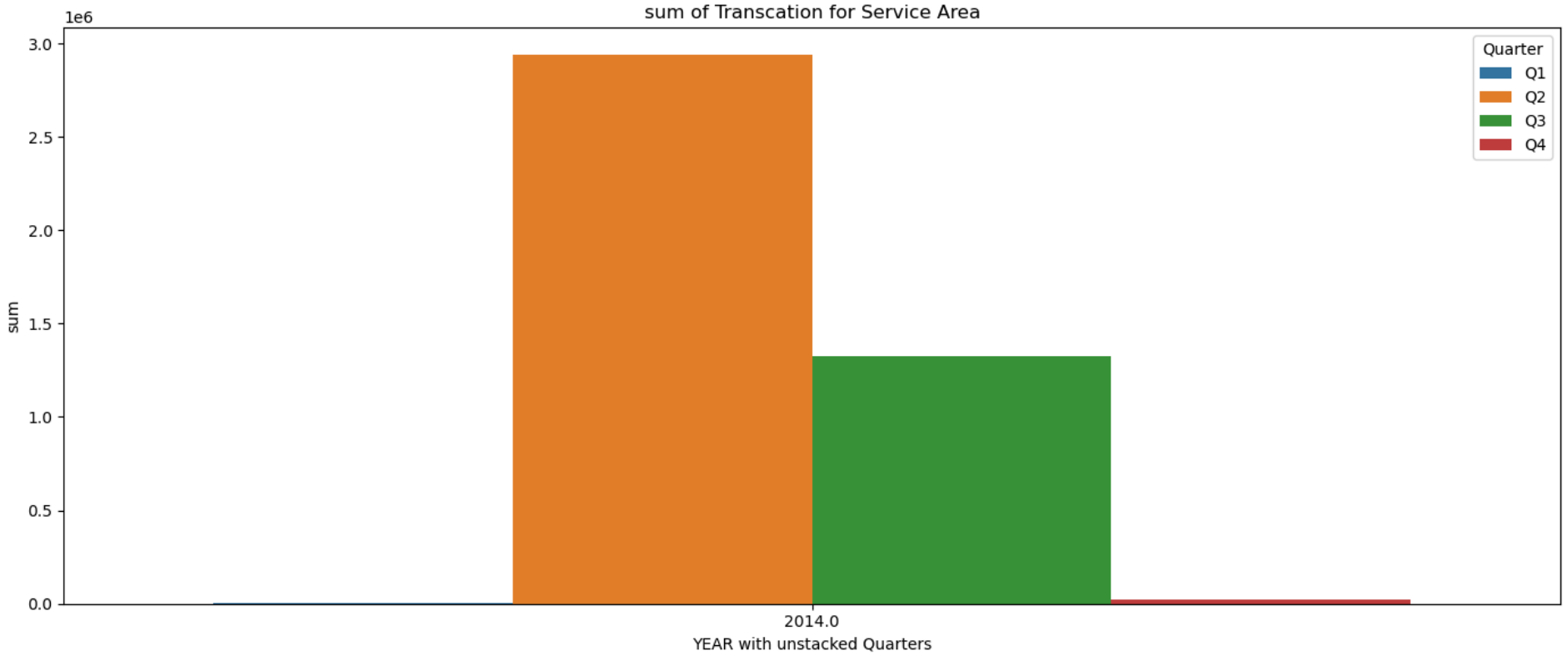
Question 1 part 1

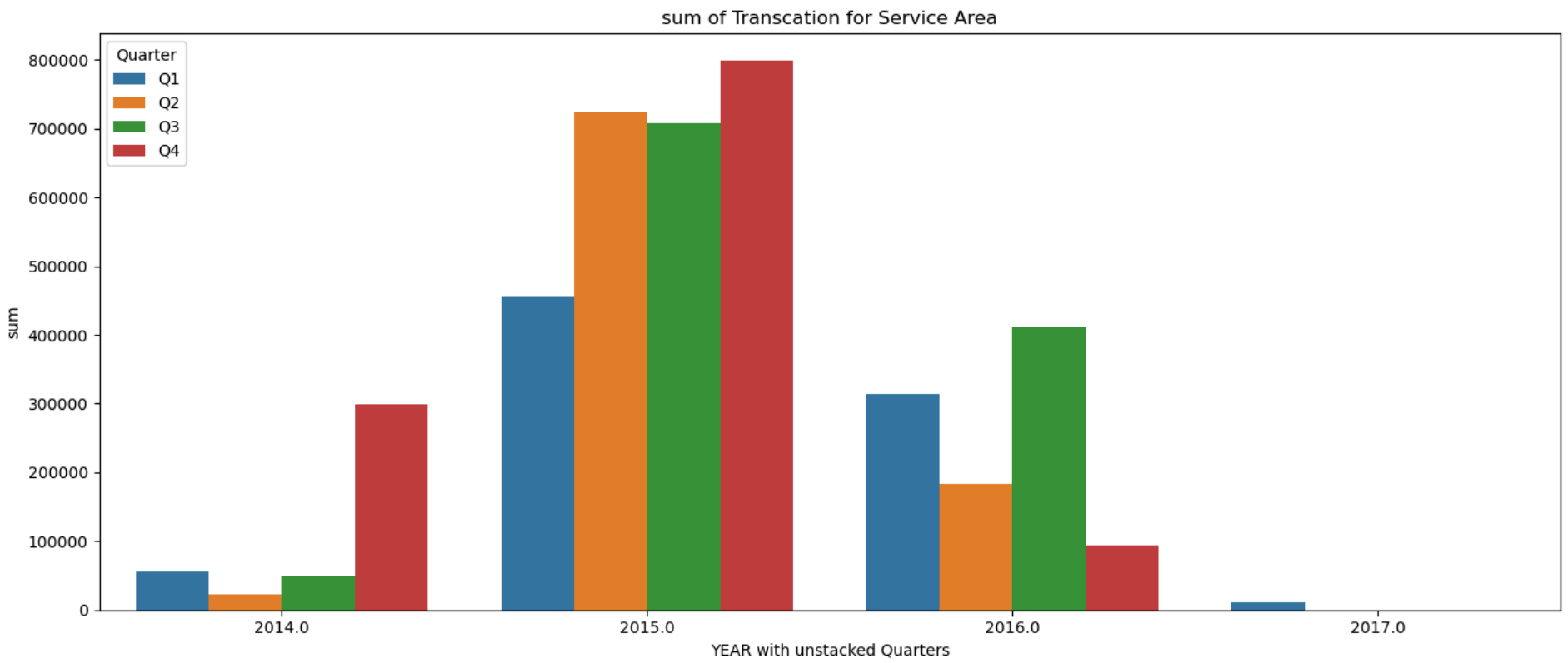
Vizualization of transaction stats for each service area

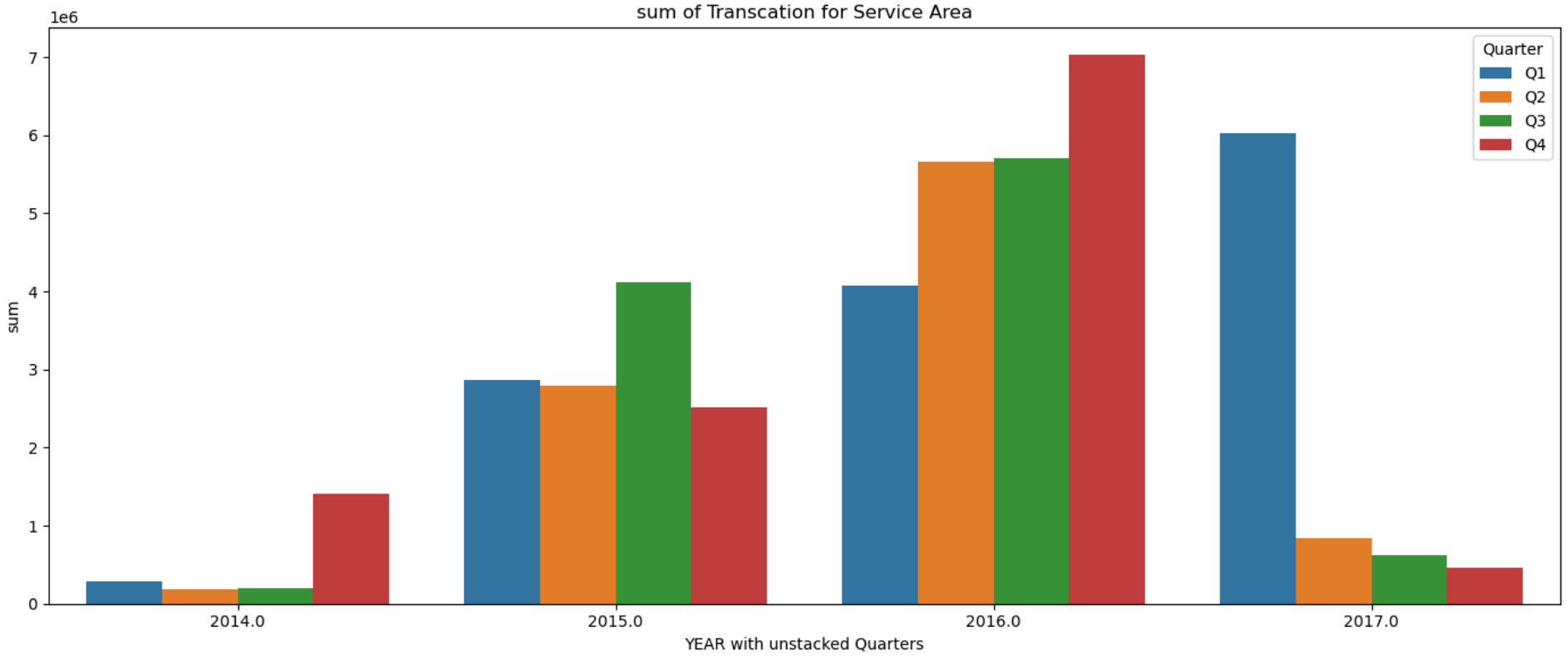
In [48]:
```
plot_q1(sa_year_qrt,'Service Area','sum') #passing service area column for stats
```

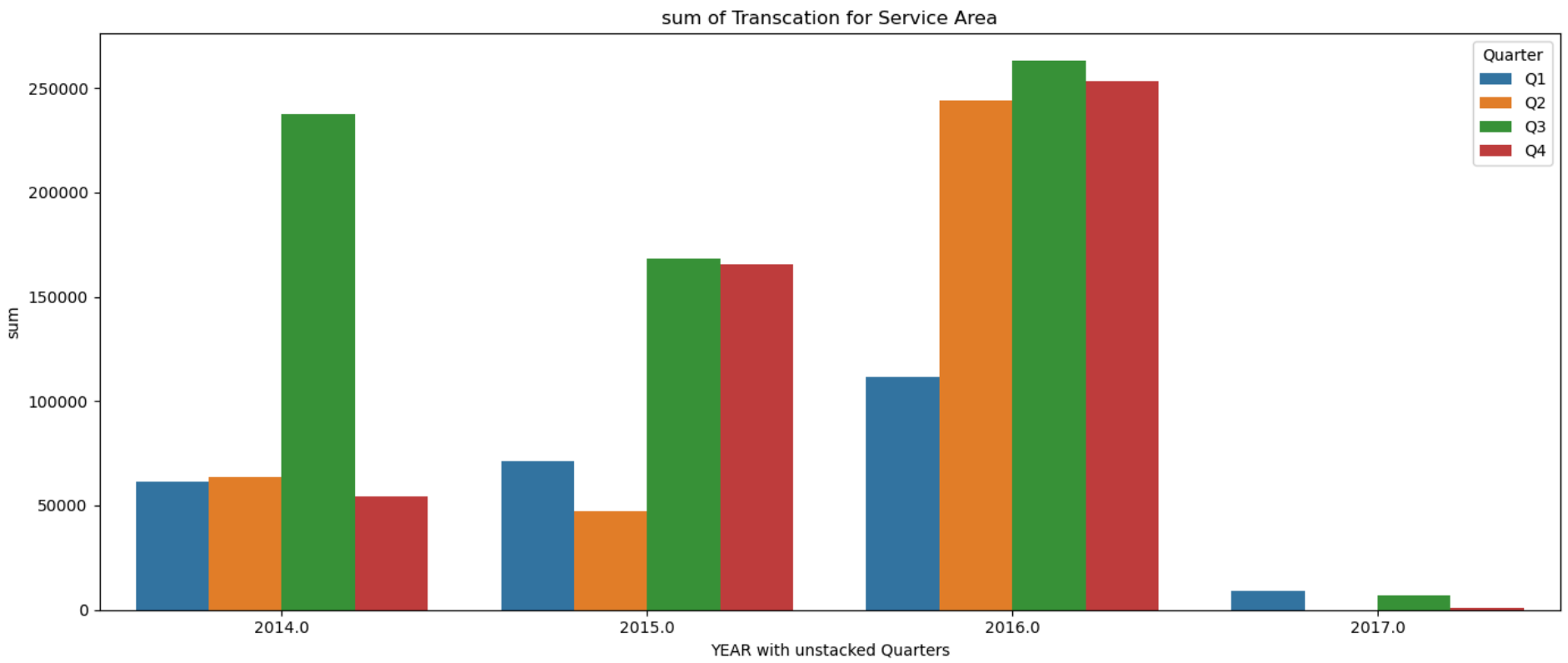


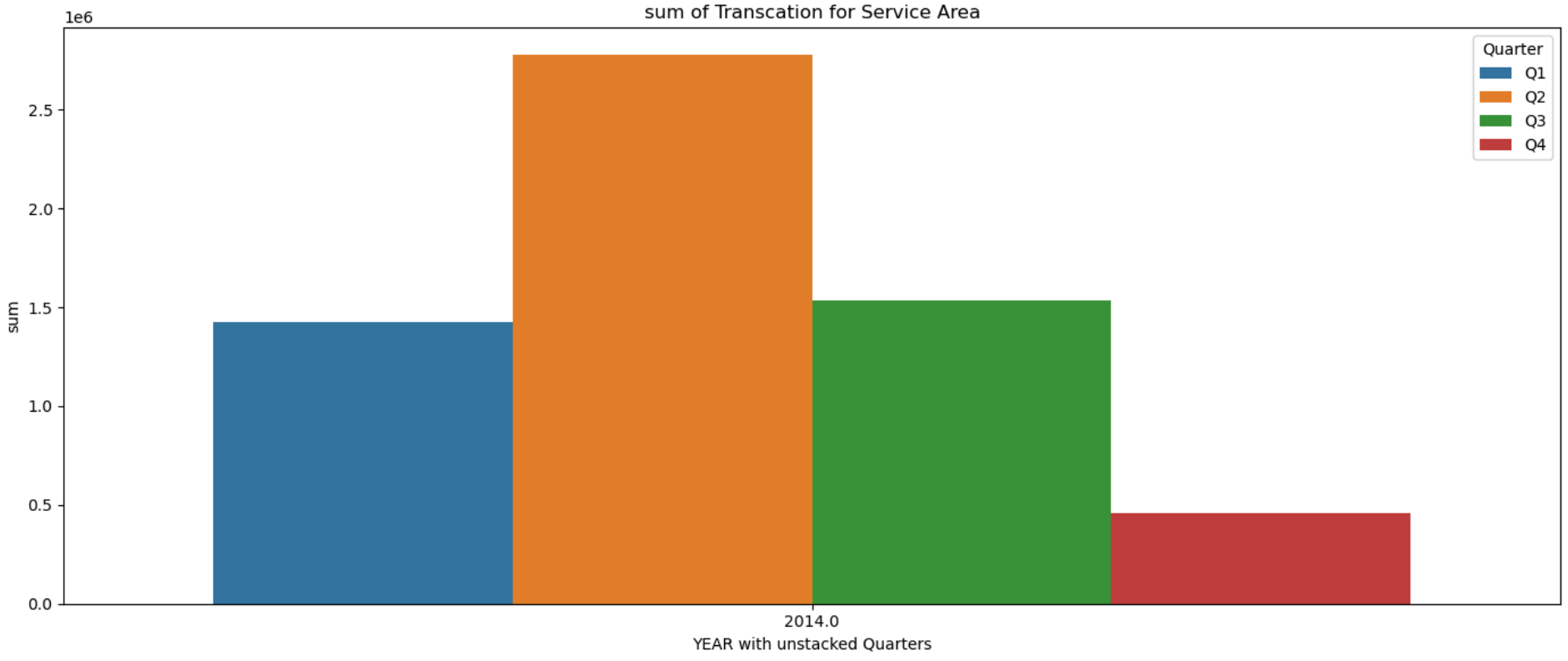

sum of Transcation for Service Area
Quarter
Q2
Q4
Q1
Q3
500000
400000
300000
200000
100000
0
sum
2014.0
2015.0
2016.0
2017.0
YEAR with unstacked Quarters

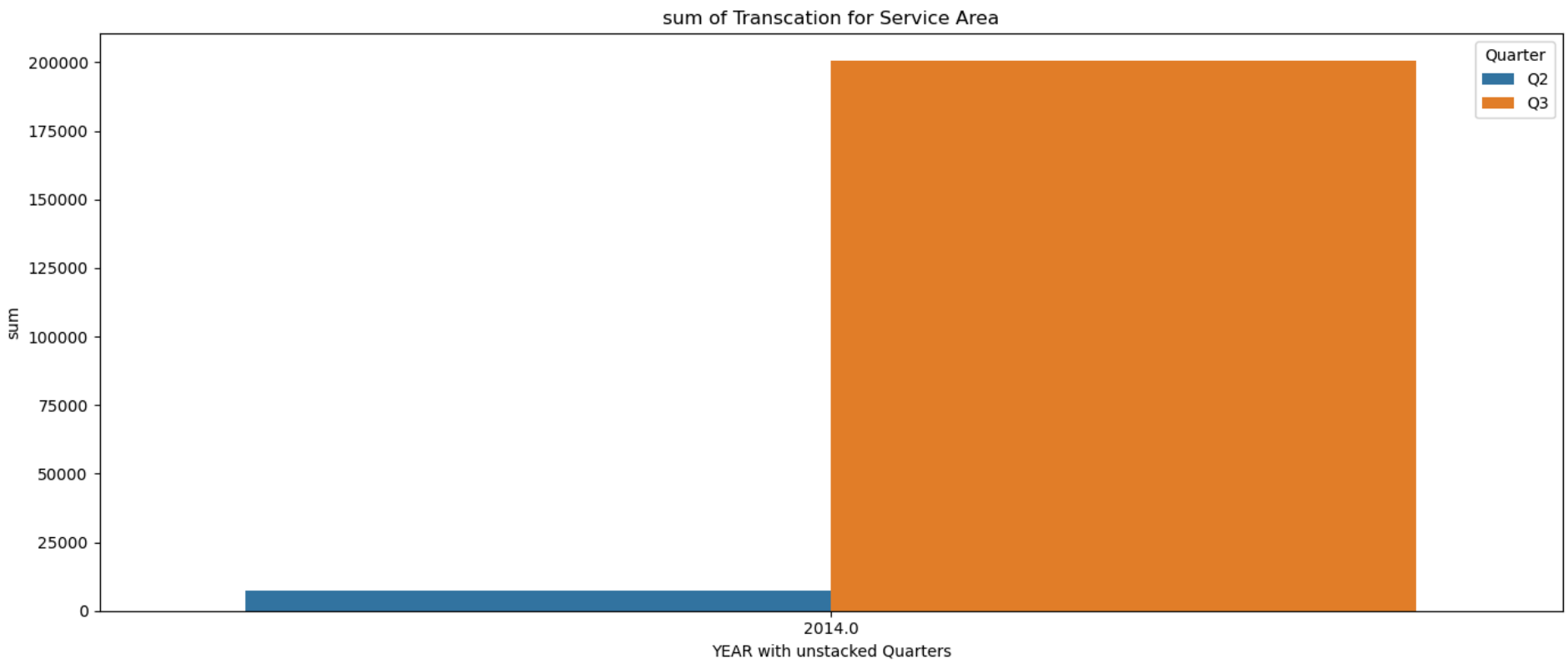

sum of Transcation for Service Area
1e6
Quarter
Q1
Q2
Q3
Q4
3.0
2.5
2.0
1.5
1.0
0.5
0.0
sum
2014.0
YEAR with unstacked Quarters

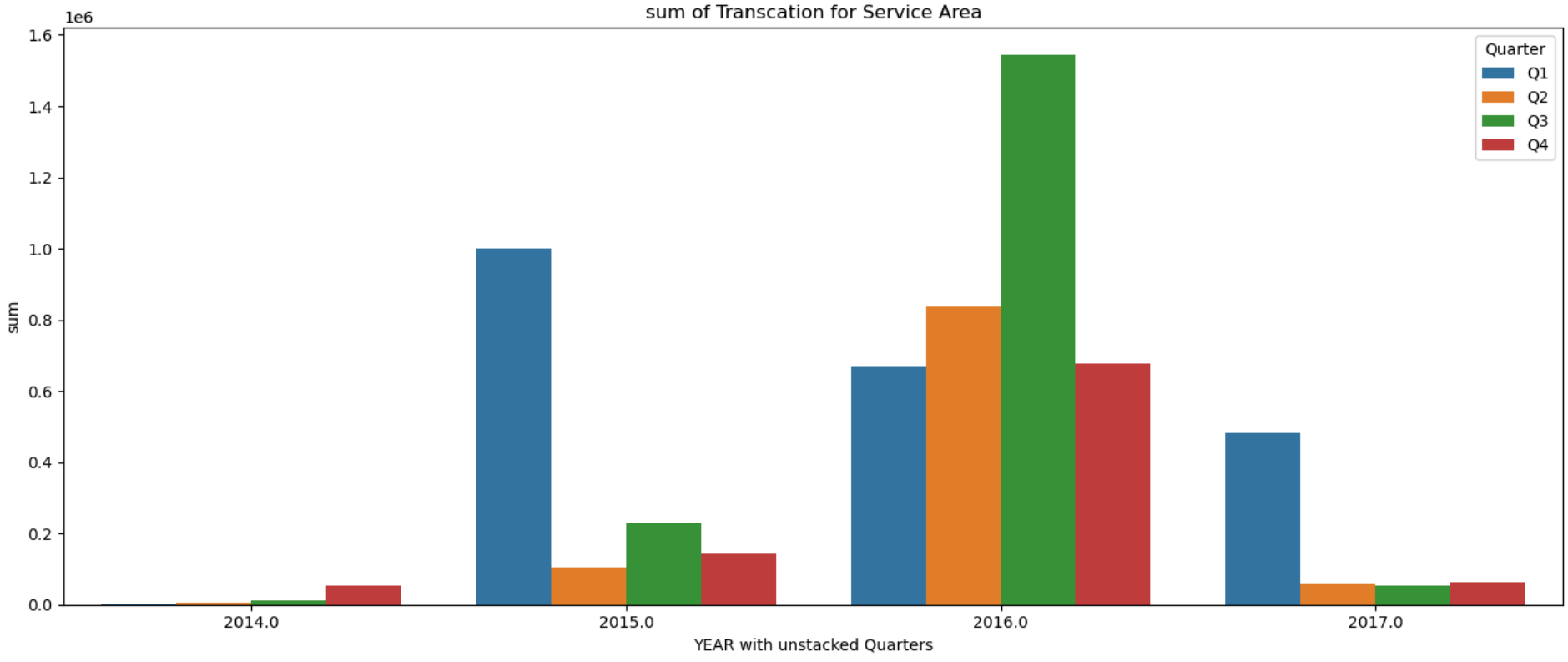

sum of Transcation for Service Area
Quarter
Q1
Q2
Q3
Q4
800000
700000
600000
500000
400000
300000
200000
100000
0
sum
2014.0
2015.0
2016.0
2017.0
YEAR with unstacked Quarters

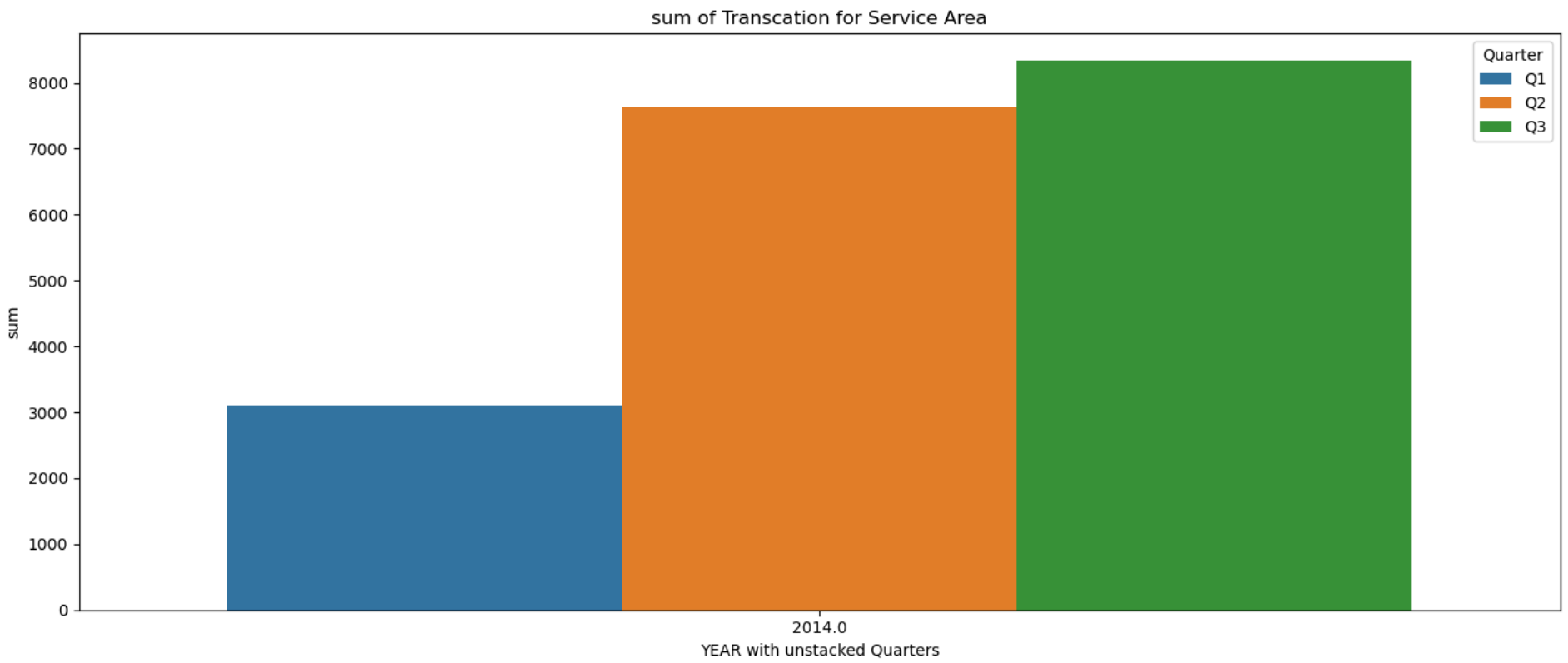

sum of Transcation for Service Area
1e6
Quarter
Q1
Q2
Q3
Q4
7
6
5
4
3
2
1
0
sum
2014.0
2015.0
2016.0
2017.0
YEAR with unstacked Quarters

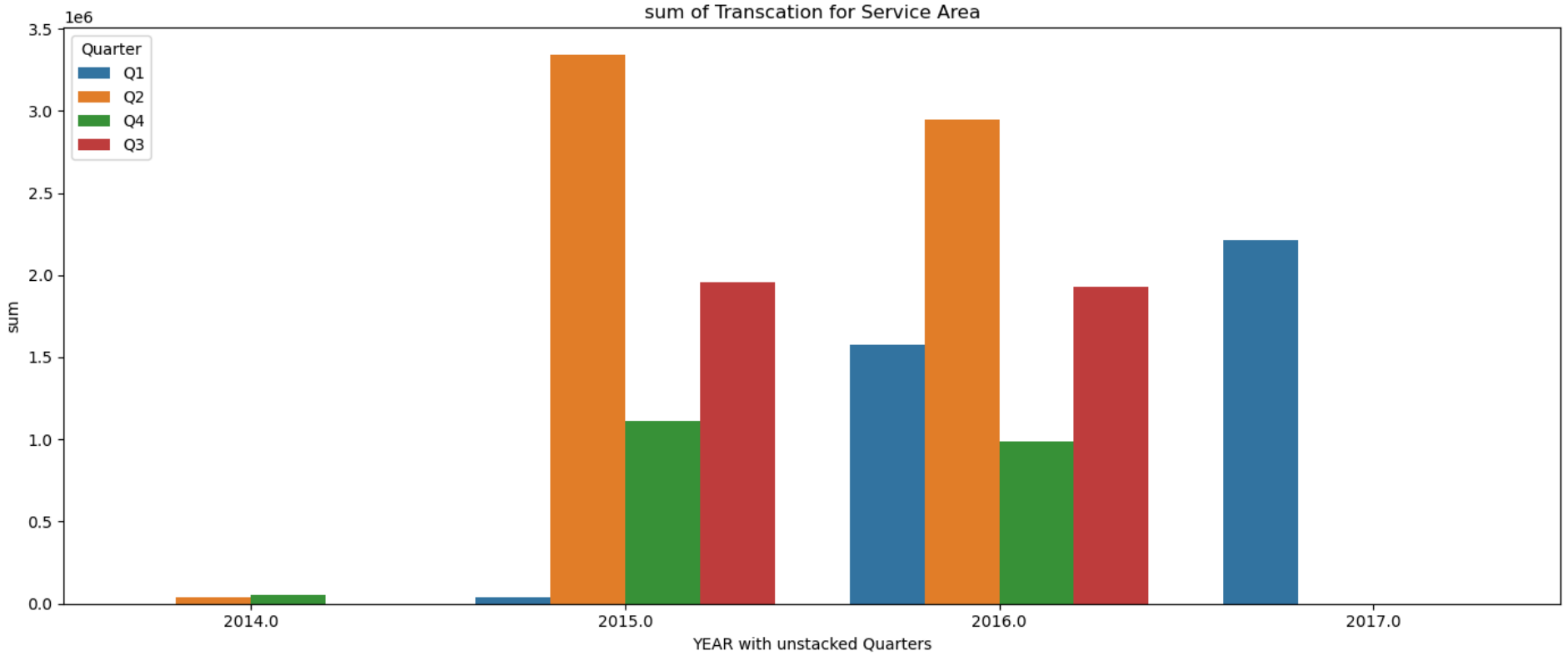

sum of Transcation for Service Area
Quarter
Q1
Q2
Q3
Q4
250000
200000
150000
100000
50000
0
sum
2014.0
2015.0
2016.0
2017.0
YEAR with unstacked Quarters

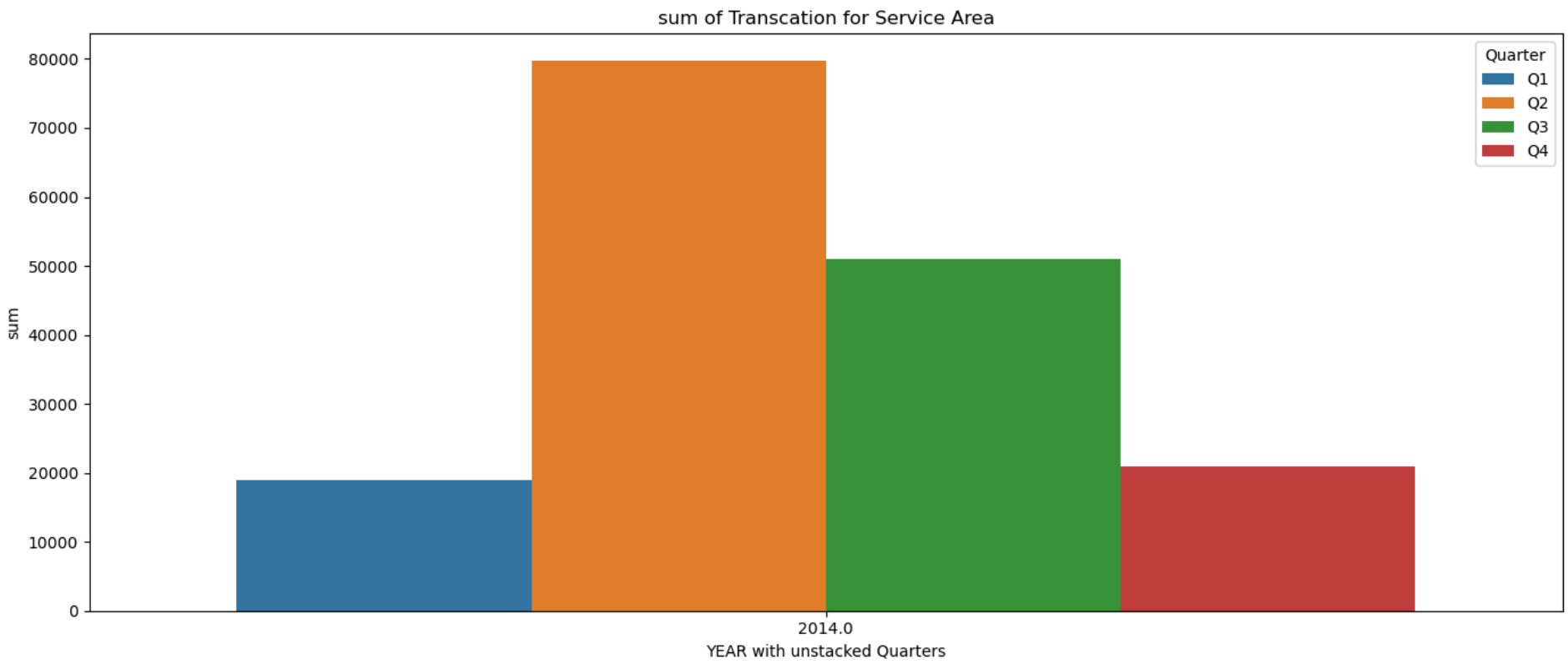

sum of Transcation for Service Area
1e6
Quarter
Q1
Q2
Q3
Q4
2.5
2.0
1.5
1.0
0.5
0.0
sum
2014.0
YEAR with unstacked Quarters

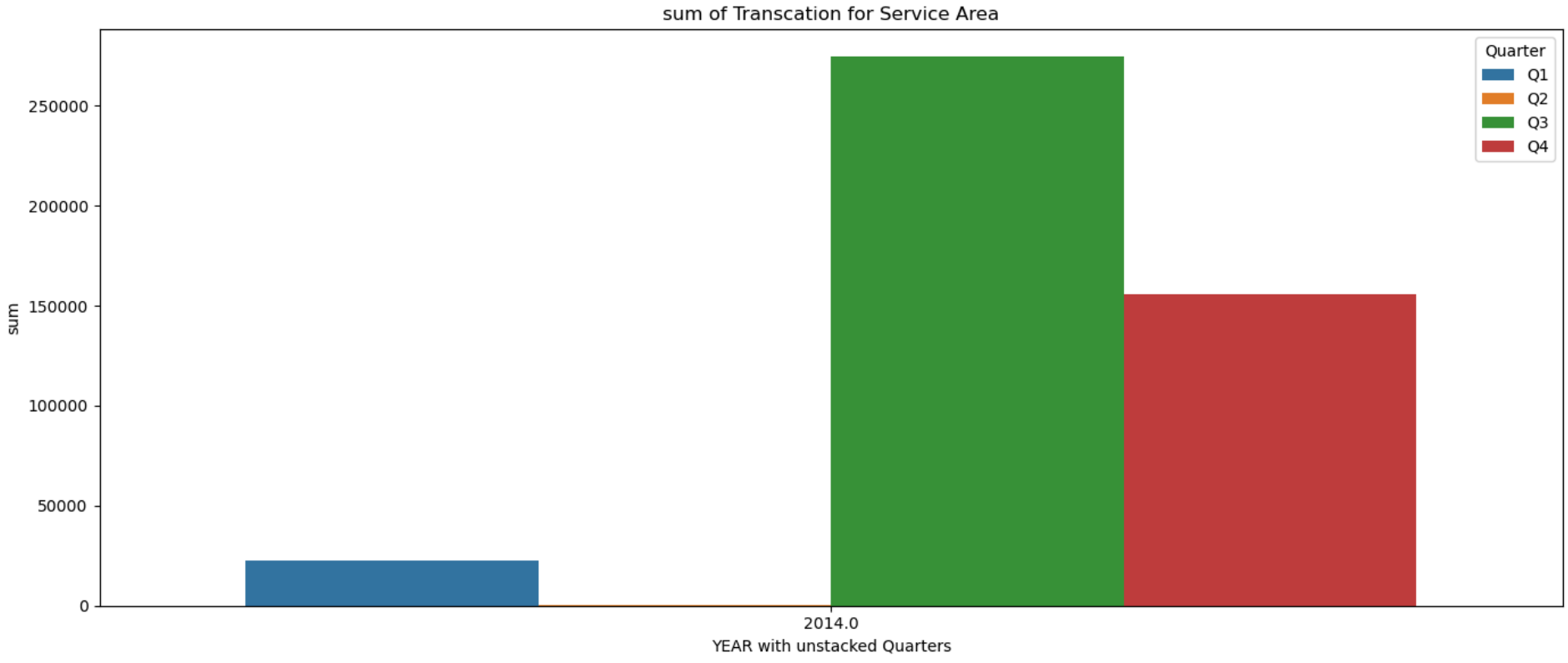

sum of Transcation for Service Area
Quarter
Q2
Q3
200000
175000
150000
125000
100000
75000
50000
25000
0
sum
2014.0
YEAR with unstacked Quarters

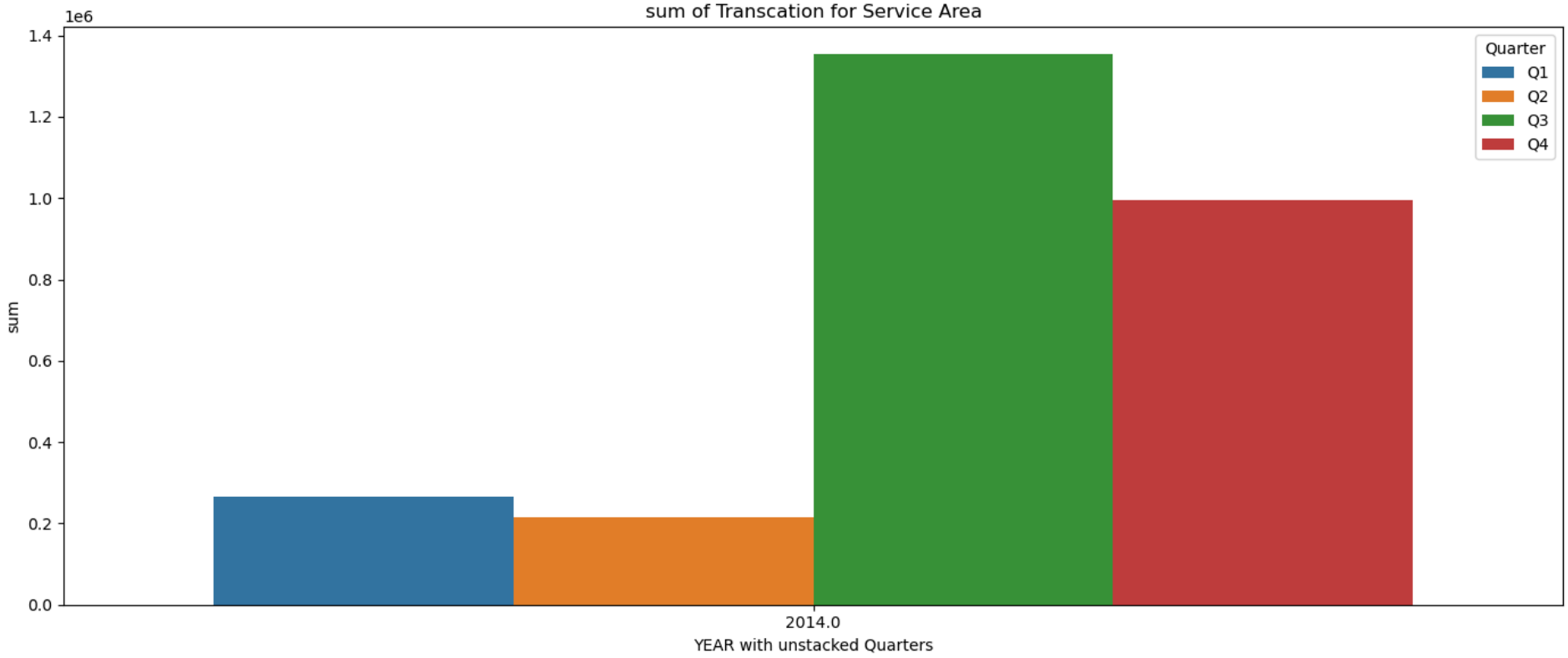

sum of Transcation for Service Area
1e6
Quarter
Q1
Q2
Q3
Q4
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
sum
2014.0
2015.0
2016.0
2017.0
YEAR with unstacked Quarters

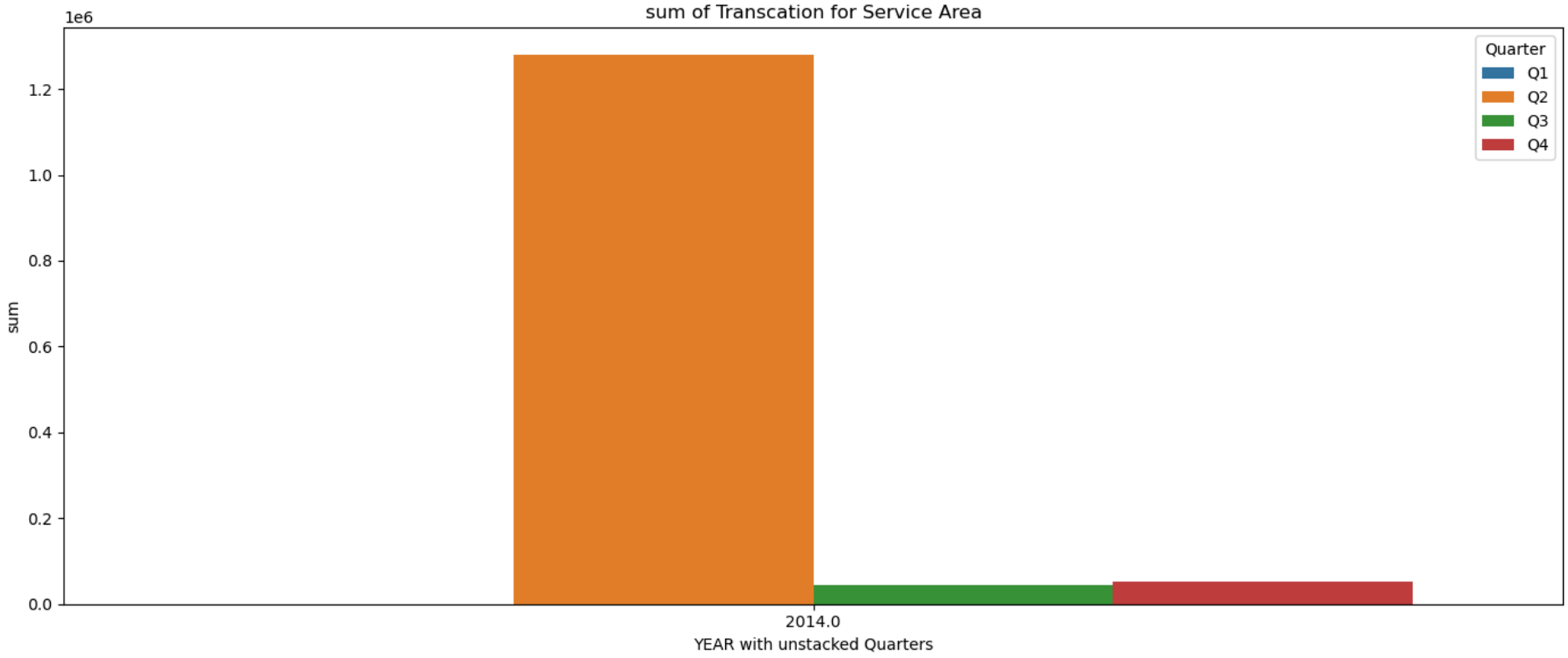

sum of Transcation for Service Area
Quarter
Q1
Q2
Q3
8000
7000
6000
5000
4000
3000
2000
1000
0
sum
2014.0
YEAR with unstacked Quarters

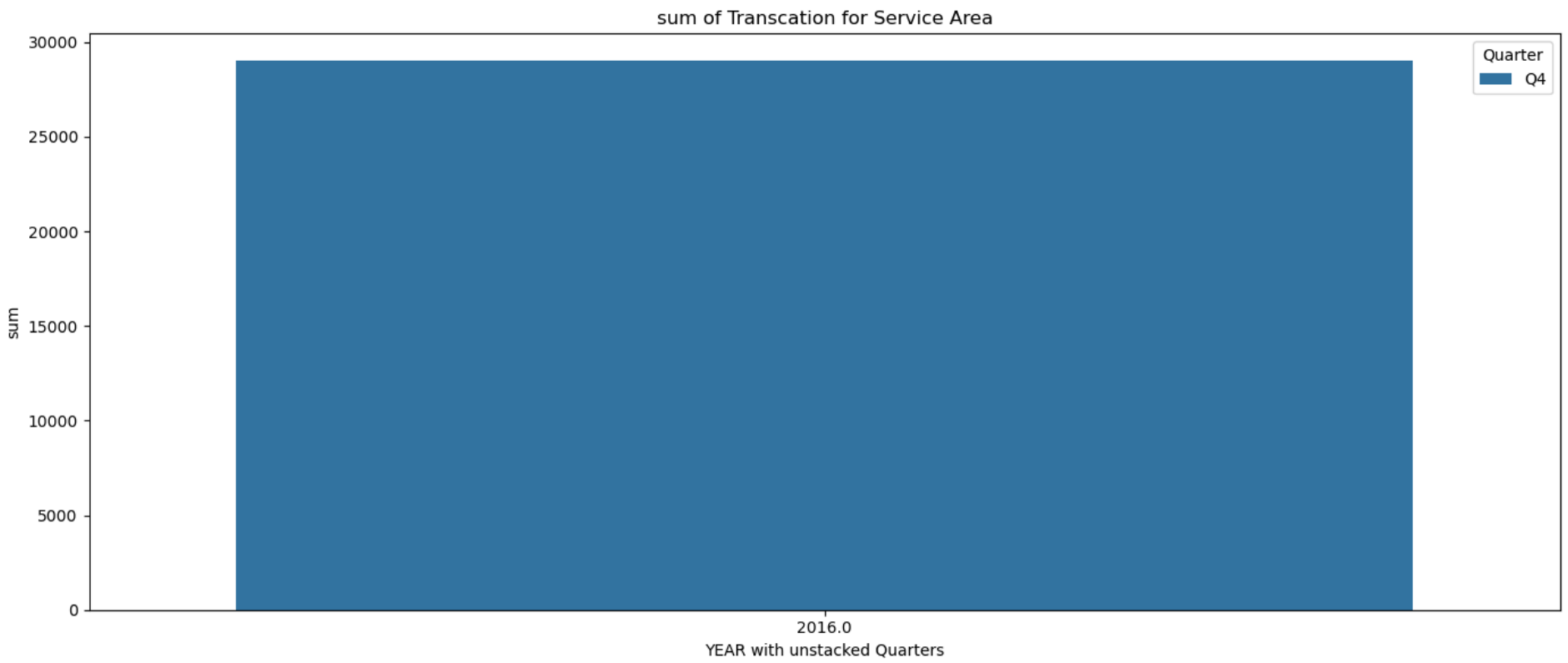

sum of Transcation for Service Area
1e6
Quarter
Q1
Q2
Q4
Q3
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
sum
2014.0
2015.0
2016.0
2017.0
YEAR with unstacked Quarters

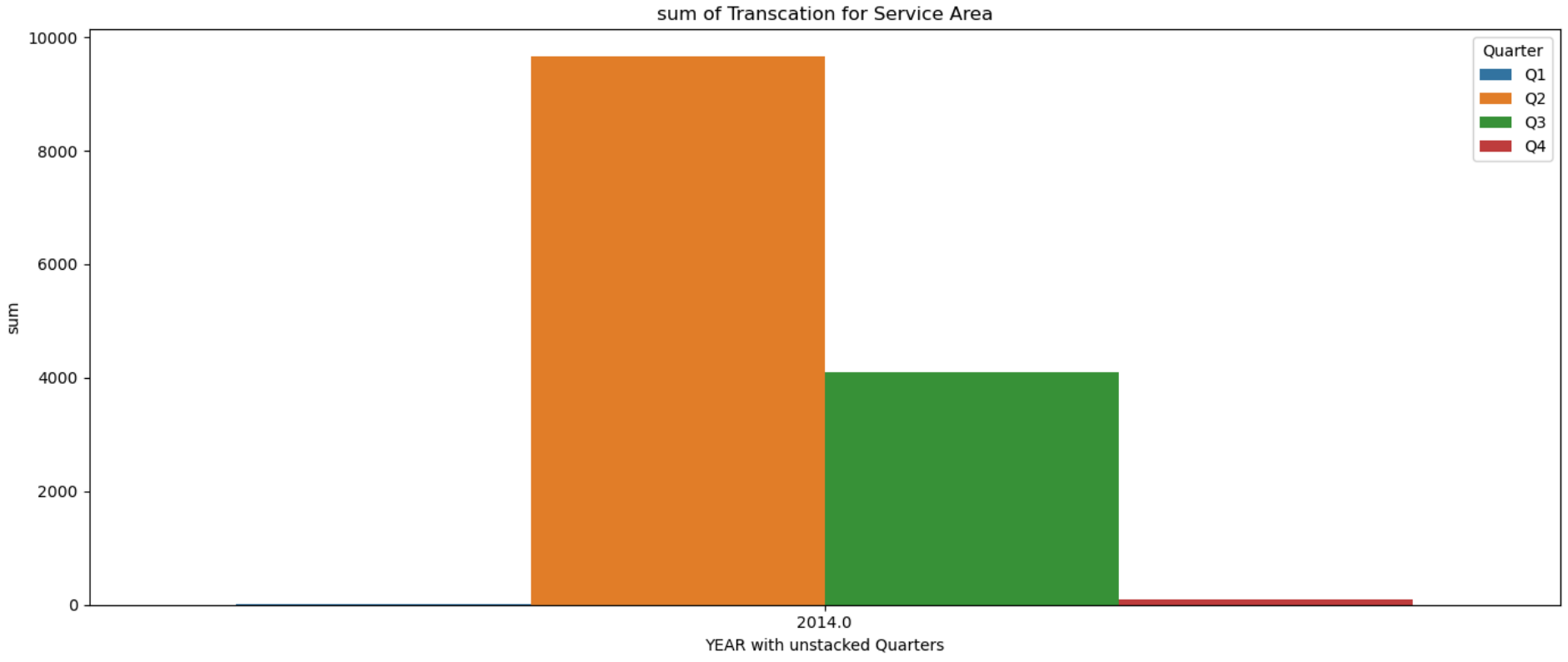

sum of Transcation for Service Area
Quarter
Q1
Q2
Q3
Q4
sum
80000
70000
60000
50000
40000
30000
20000
10000
0
2014.0
YEAR with unstacked Quarters

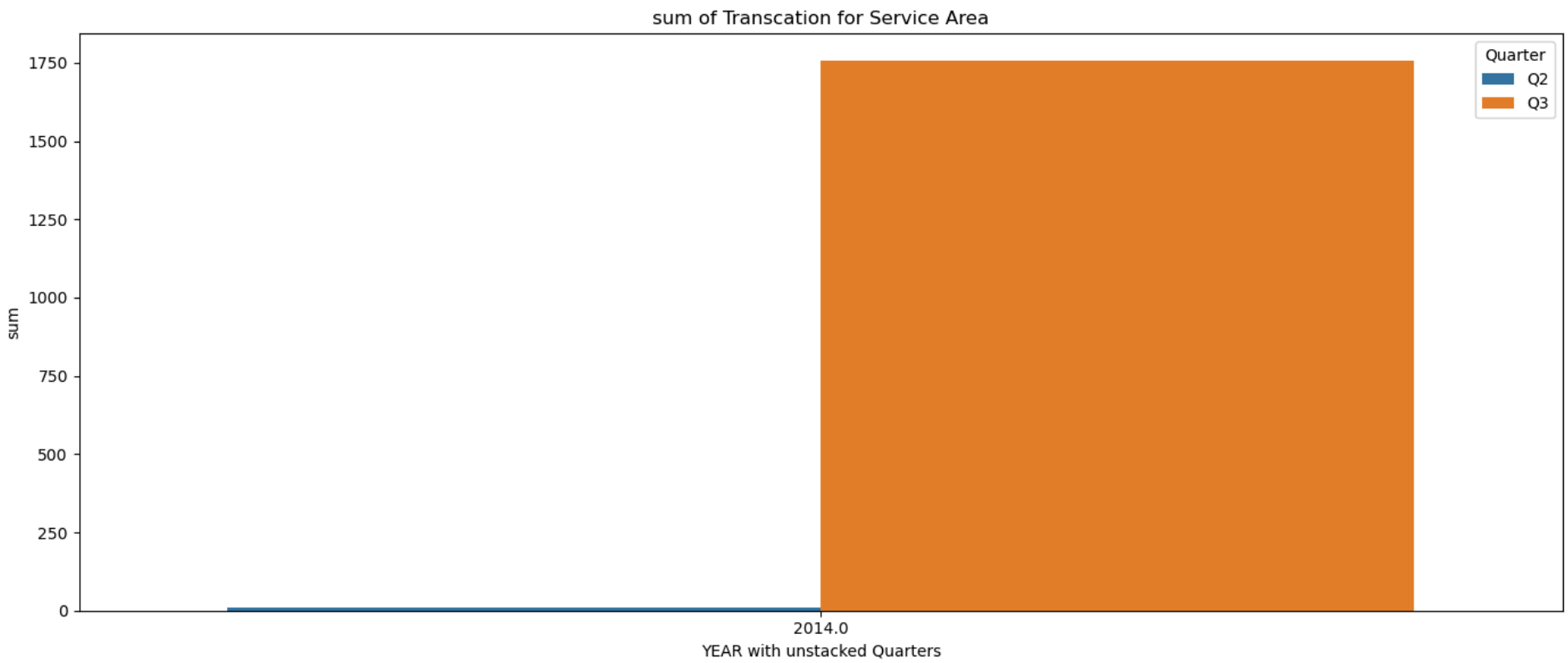

sum of Transcation for Service Area
Quarter
Q1
Q2
Q3
Q4
sum
250000
200000
150000
100000
50000
0
2014.0
YEAR with unstacked Quarters

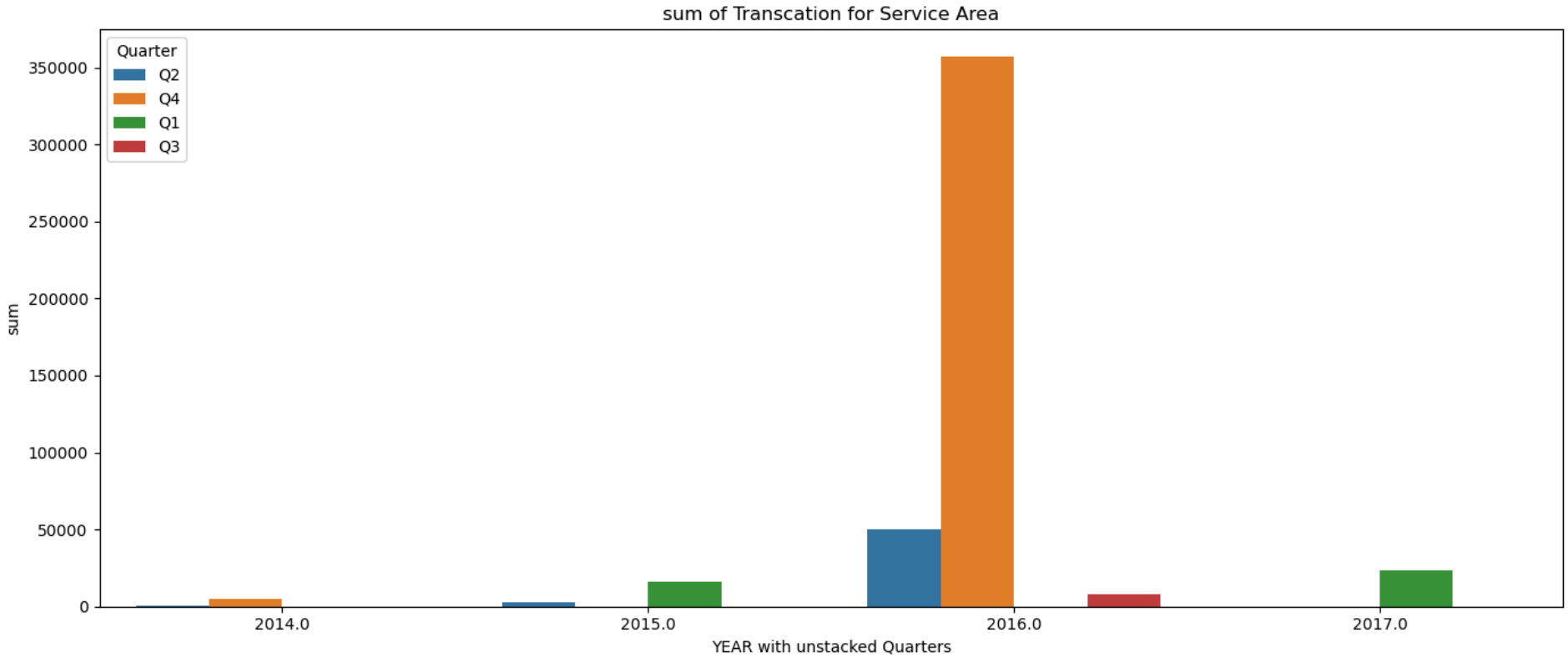

sum of Transcation for Service Area
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
sum
Quarter
Q1
Q2
Q3
Q4
2014.0
YEAR with unstacked Quarters

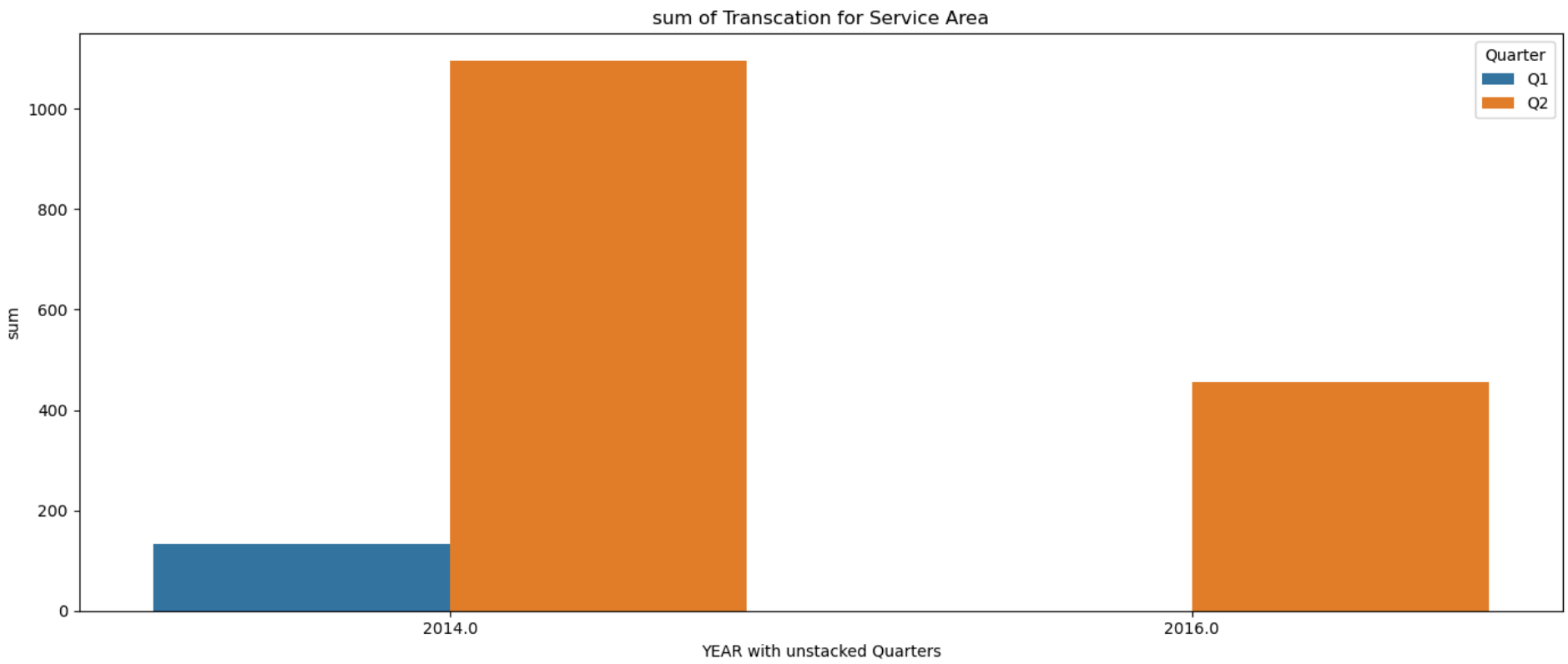

sum of Transcation for Service Area
1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
sum
Quarter
Q1
Q2
Q3
Q4
2014.0
YEAR with unstacked Quarters

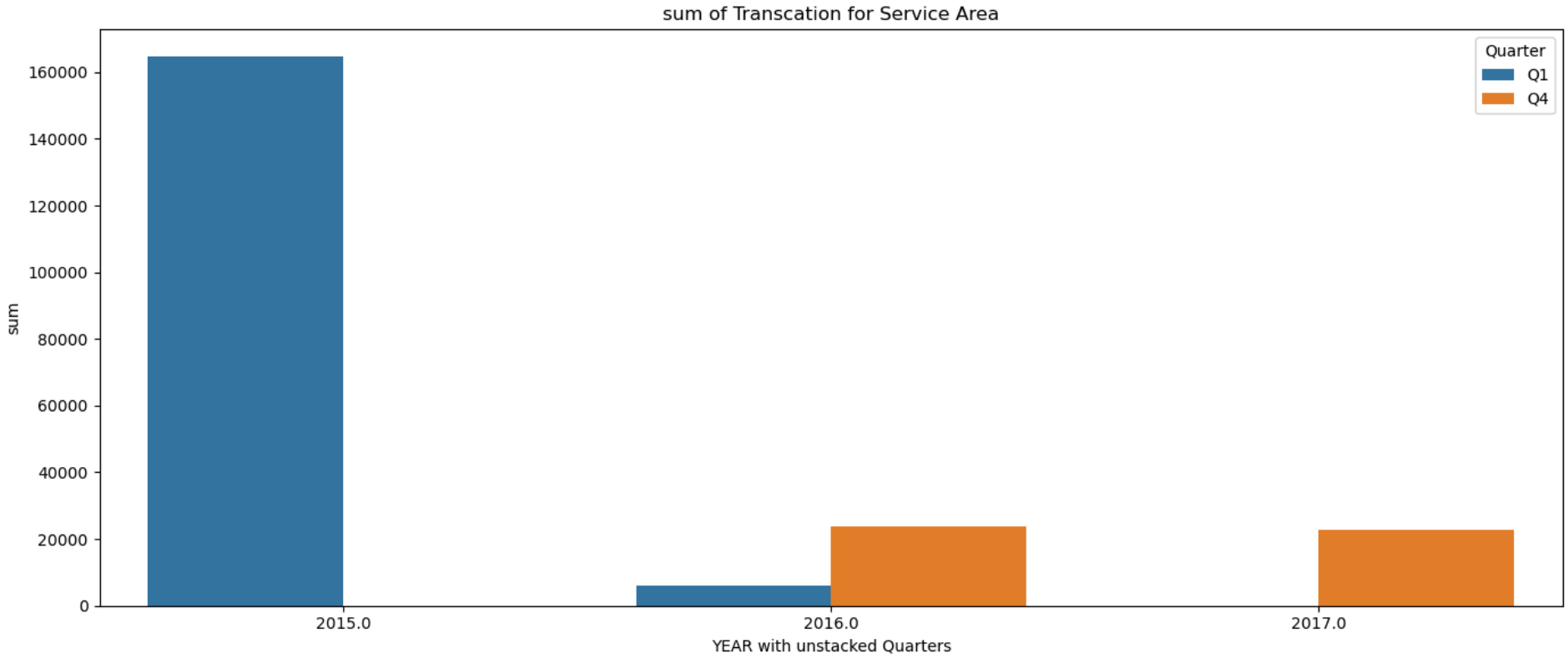

sum of Transcation for Service Area
30000
25000
20000
15000
10000
5000
0
sum
Quarter
Q4
2016.0
YEAR with unstacked Quarters

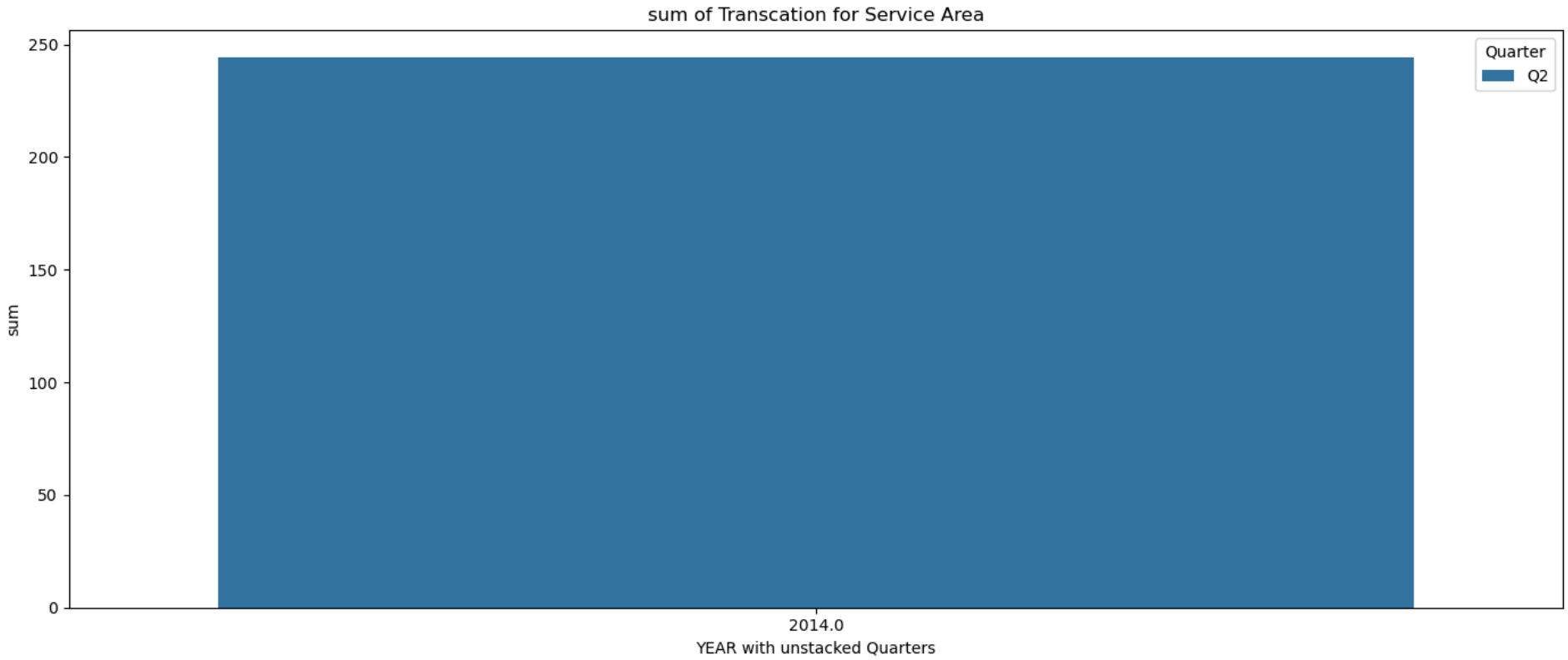

sum of Transcation for Service Area
10000
8000
6000
4000
2000
0
sum
Quarter
Q1
Q2
Q3
Q4
2014.0
YEAR with unstacked Quarters

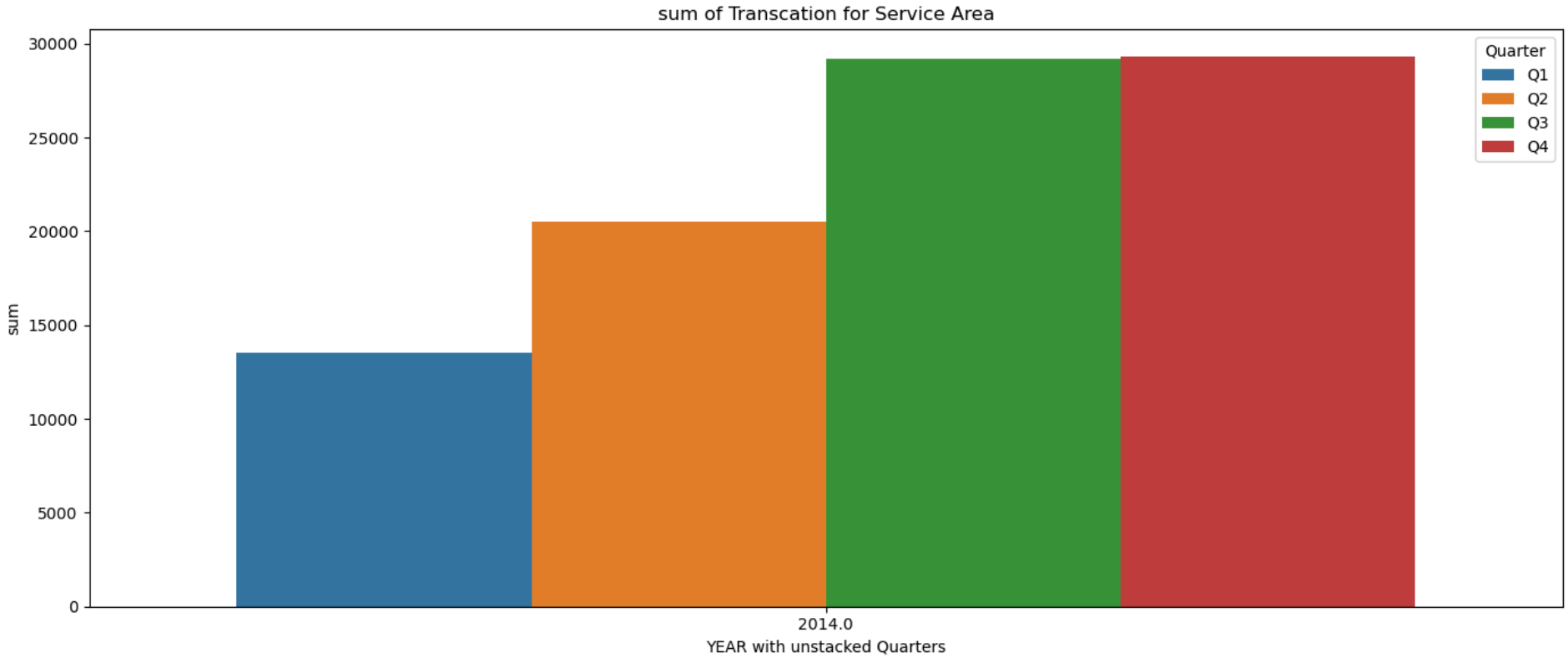

sum of Transcation for Service Area
Quarter
Q2
Q3
1750
1500
1250
1000
750
500
250
0
sum
2014.0
YEAR with unstacked Quarters

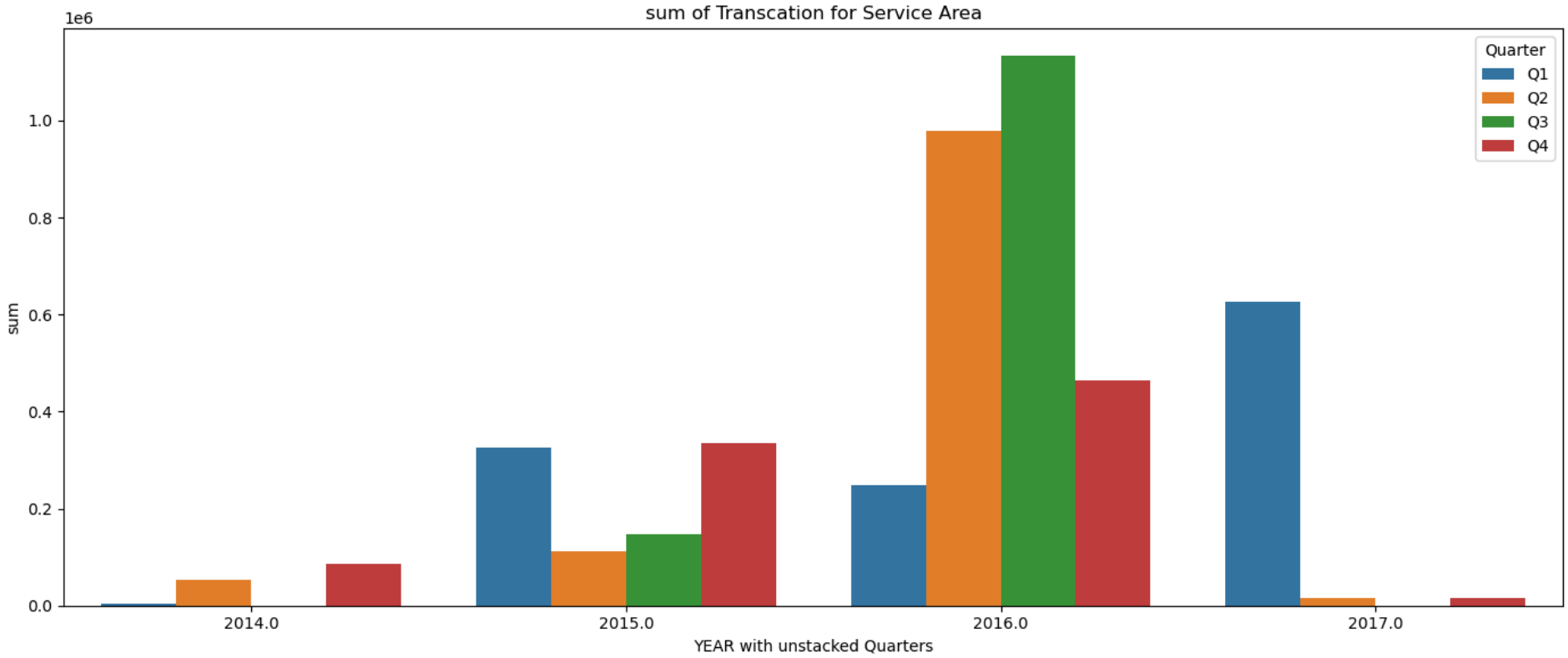

sum of Transcation for Service Area
Quarter
Q2
Q4
Q1
Q3
350000
300000
250000
200000
150000
100000
50000
0
sum
2014.0
2015.0
2016.0
2017.0
YEAR with unstacked Quarters


sum of Transcation for Service Area
Quarter
Q1
Q2
1000
800
600
400
200
0
sum
2014.0
2016.0
YEAR with unstacked Quarters


sum of Transcation for Service Area
Quarter
Q1
Q4
160000
140000
120000
100000
80000
60000
40000
20000
0
sum
2015.0
2016.0
2017.0
YEAR with unstacked Quarters


sum of Transcation for Service Area
Quarter
Q2
250
200
150
100
50
0
sum
2014.0
YEAR with unstacked Quarters


sum of Transcation for Service Area
Quarter
Q1
Q2
Q3
Q4
30000
25000
20000
15000
10000
5000
0
sum
2014.0
YEAR with unstacked Quarters



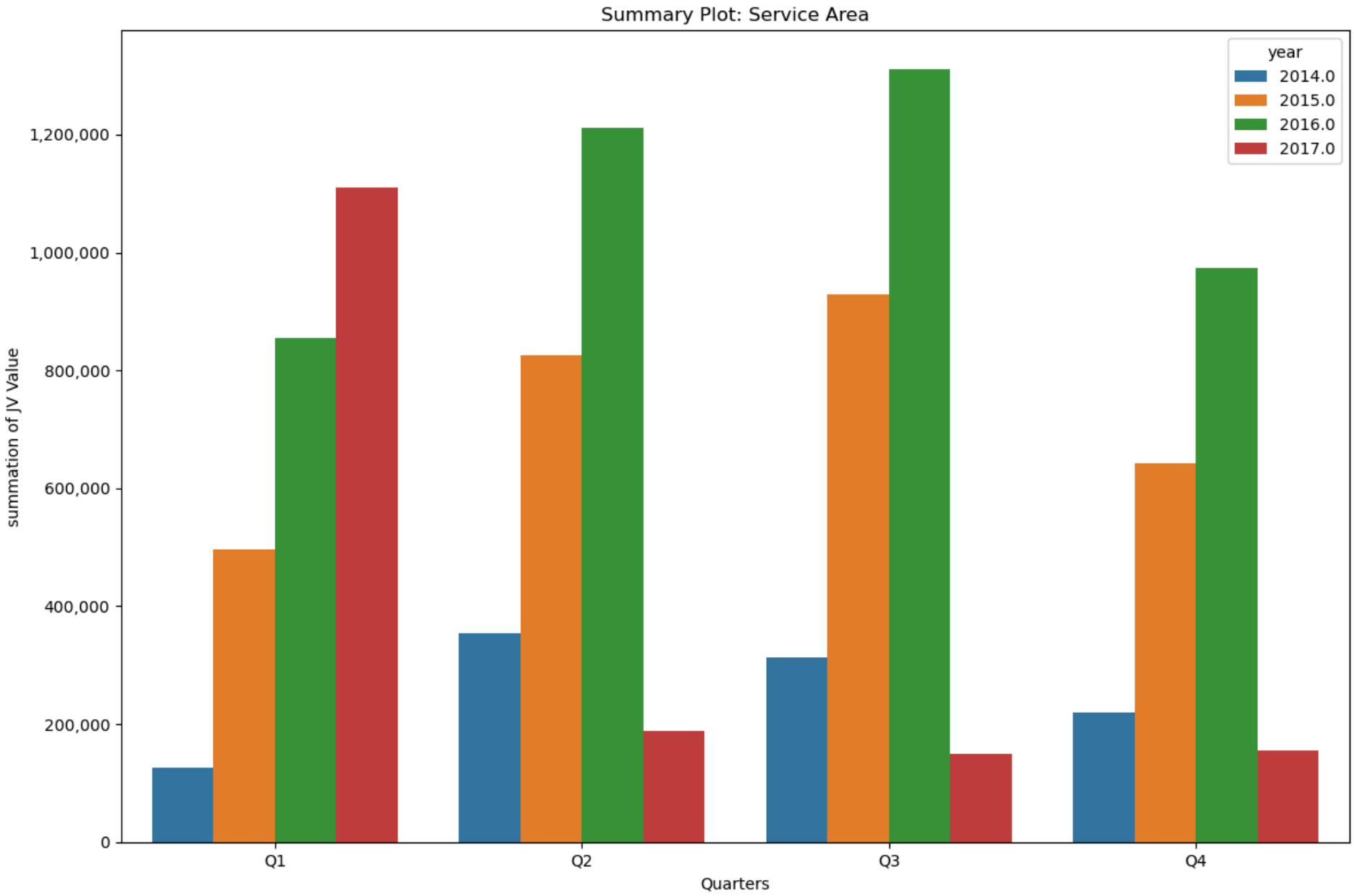
Question 1 part 1

Vizualization of transaction stats for each accounts

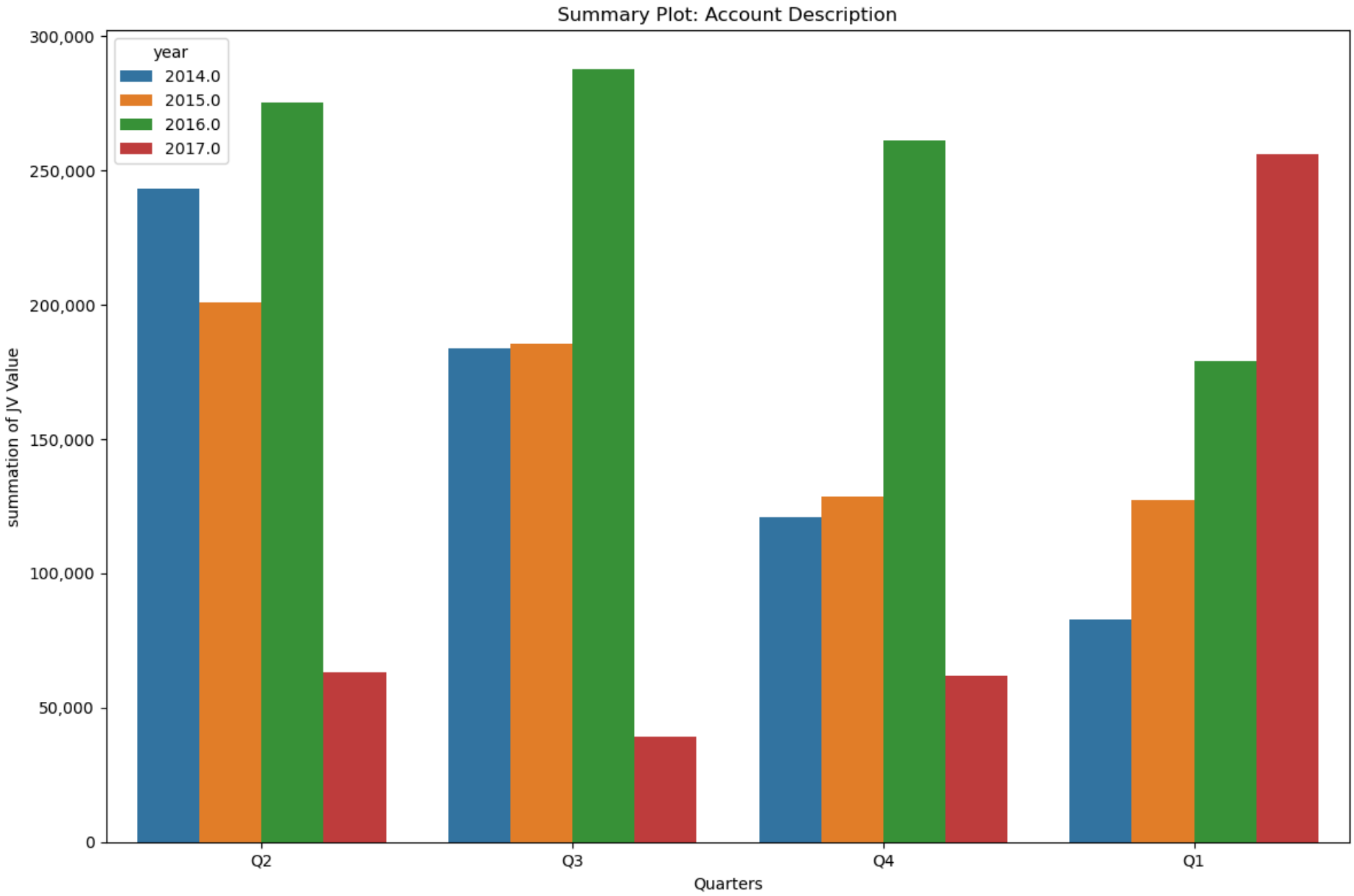
In [ ]:
```
plot_q1(ad_year_qt,'Account Description','sum') #passing account column for stats
```

Question 1 final summary

In [50]:
```
plot_q1a(df,'Service Area','JV Value')
```



In [51]:
```
plot_q1a(df,'Account Description','JV Value')
```



++

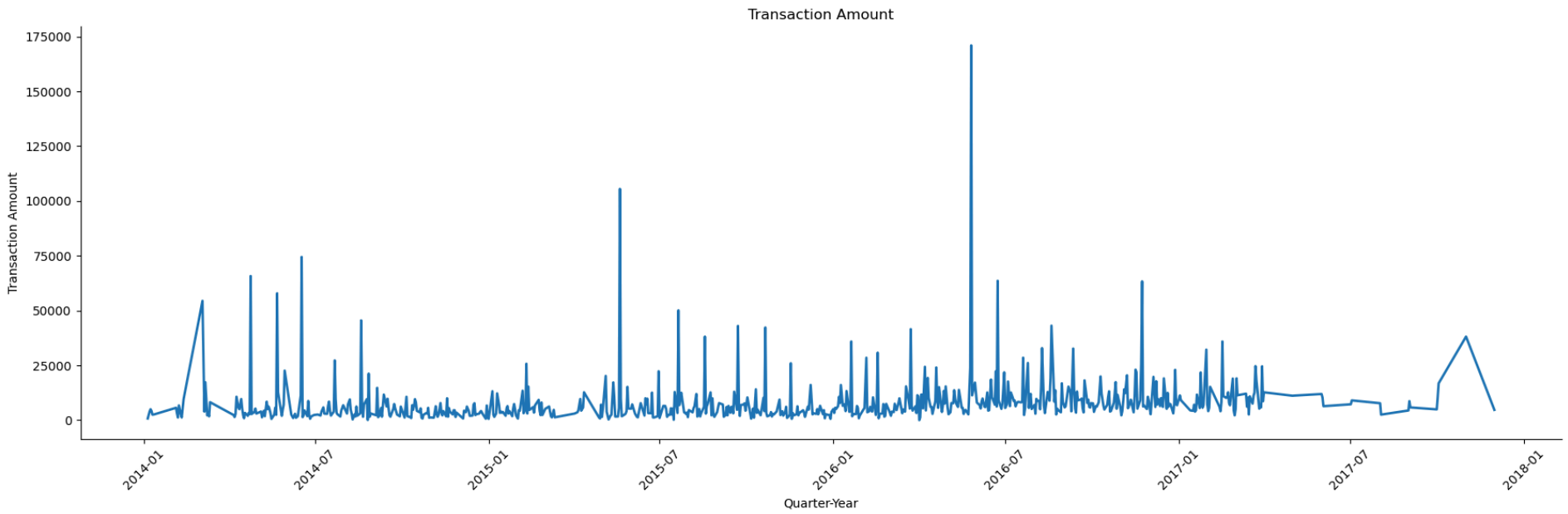
Question 2 part 1

creating plot and observing spike on various time sectors

In [52]:
```
#created few more tables for easy plotting

df_filtered = df.dropna(subset=['QT_validation'])
df_filtered['Quarter'] = df_filtered['Transaction Date'].dt.to_period('Q').astype(str) #the formart was creating issue so converte
## custom list of year-quarter
quarter_order = ['2014Q1', '2014Q2', '2014Q3', '2014Q4','2015Q1', '2015Q2', '2015Q3', '2015Q4','2016Q1', '2016Q2', '2016Q3', '2016
df_filtered['Quarter'] = pd.Categorical(df_filtered['Quarter'], categories=quarter_order, ordered=True) #to control the order issu
```

In [53]:
```
#plotting the transaction amount

def spike_plot(data,x,y,col_temp=None):
    """
    arg:data--> dataframe
    arg:xaxis value
    arg:yaxis value
    col_temp: col parameter for relplot
    """
    if col_temp:
        g = sns.relplot(data=data, x=x, y=y, kind='line', col=col_temp,
                col_wrap=2, aspect=2, height=3, linewidth=2)
        g.set_titles('{col_name}')
        g.set_axis_labels('Quarter', 'Transaction Amount')

        # Adjust x-ticks for all subplots
        for ax in g.axes.flatten():
            ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

        plt.tight_layout()
        plt.show()
    else:
        sns.relplot(data=data, x=x, y=y, kind='line',height=6, linewidth=2,aspect=3,ci=None)
        plt.title('Transaction Amount')
        plt.xlabel('Quarter-Year')
        plt.ylabel('Transaction Amount')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()
```

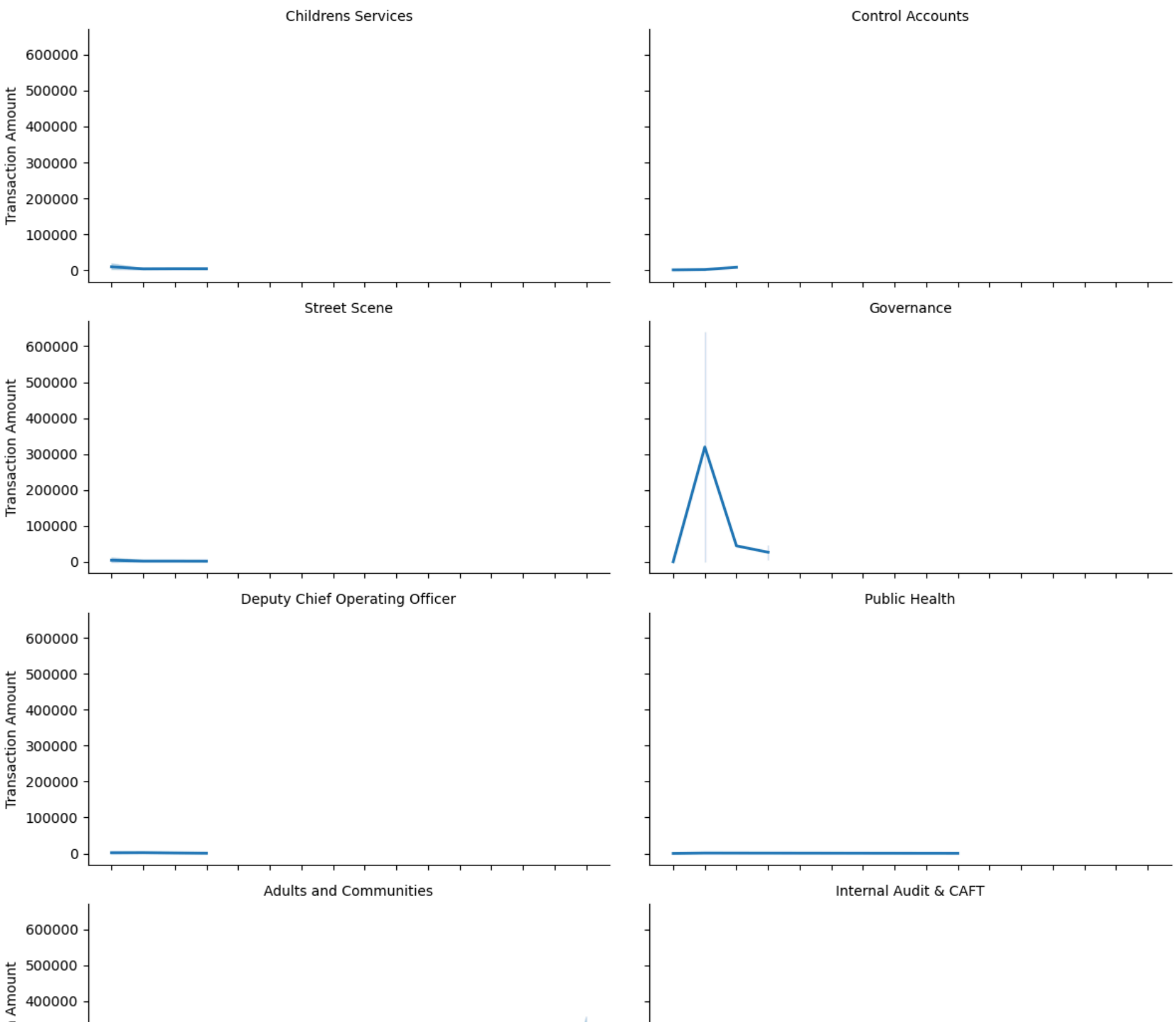
In [54]:
```
spike_plot(df,'Transaction Date','JV Value')
```

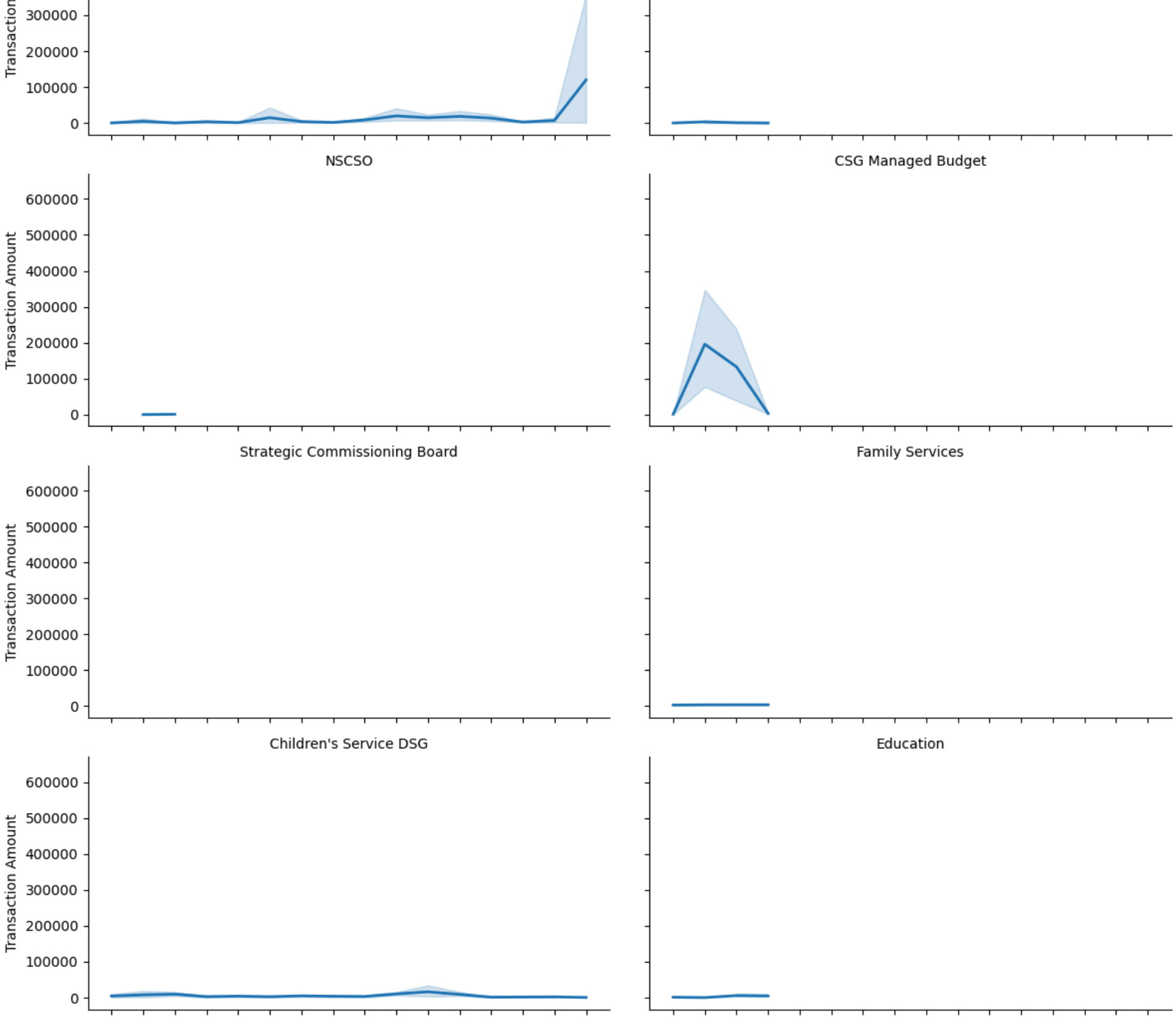


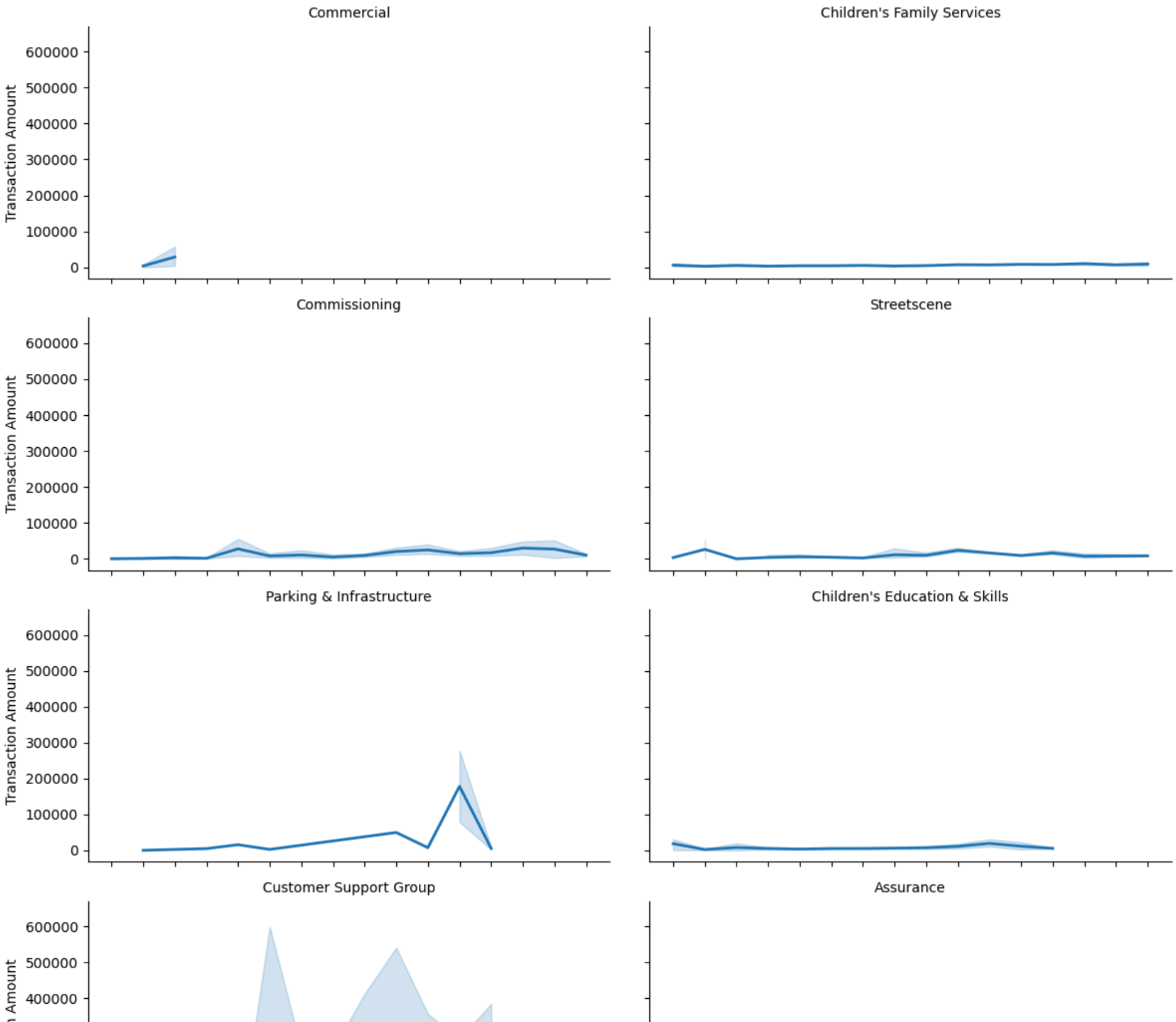
Question 2 part 1

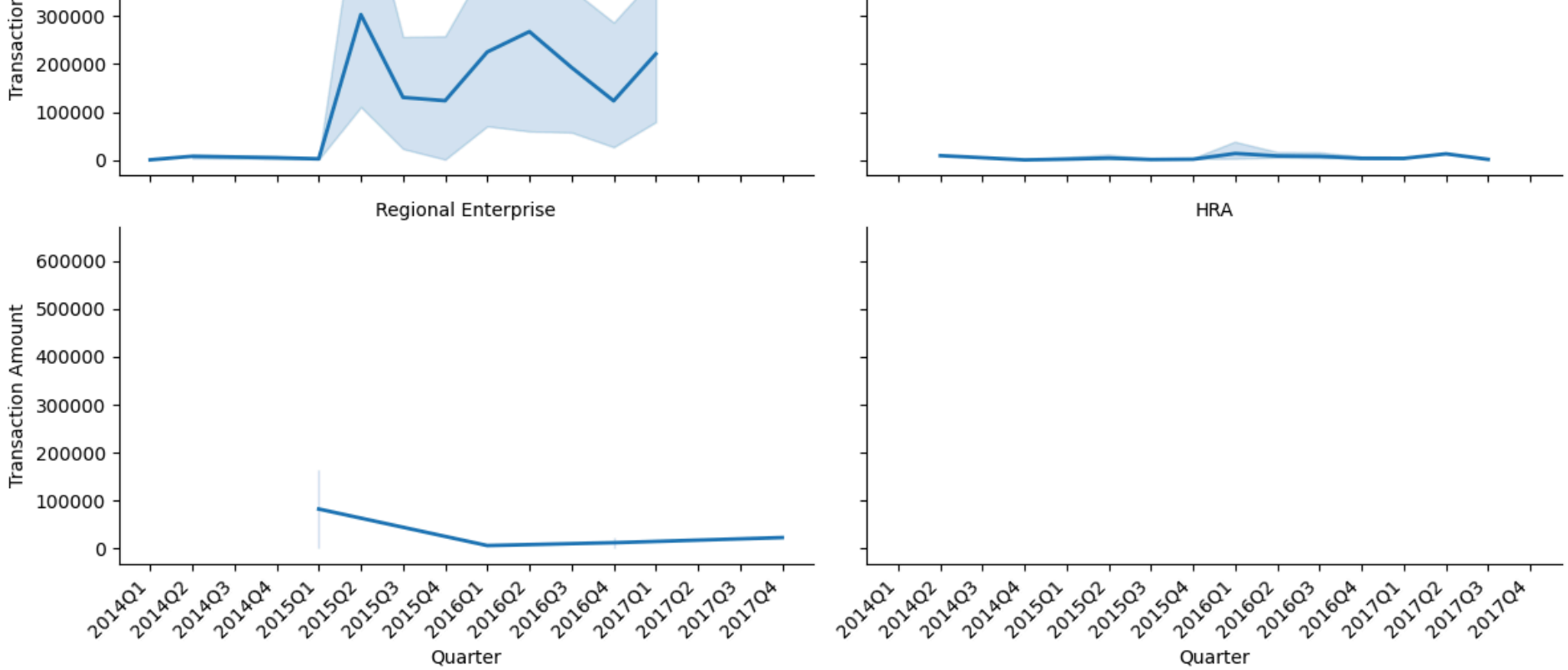
Service Area

In [55]:
```
spike_plot(df_filtered,'Quarter','JV Value',col_temp='Service Area')
```

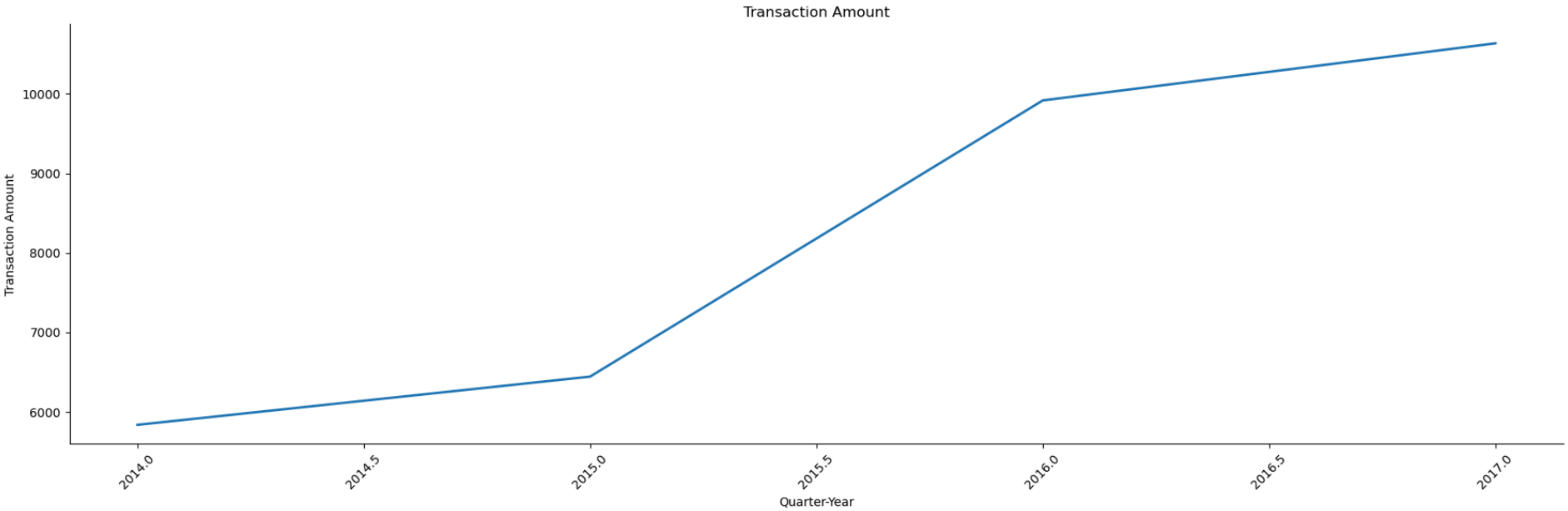

Childrens Services
Control Accounts
Street Scene
Governance
Deputy Chief Operating Officer
Public Health
Adults and Communities
Internal Audit & CAFT
Transaction Amount
600000
500000
400000
300000
200000
100000
0


Transaction Amount
300000
200000
100000
0
NSCSO
CSG Managed Budget
Transaction Amount
600000
500000
400000
300000
200000
100000
0
Strategic Commissioning Board
Family Services
Transaction Amount
600000
500000
400000
300000
200000
100000
0
Children's Service DSG
Education
Transaction Amount
600000
500000
400000
300000
200000
100000
0

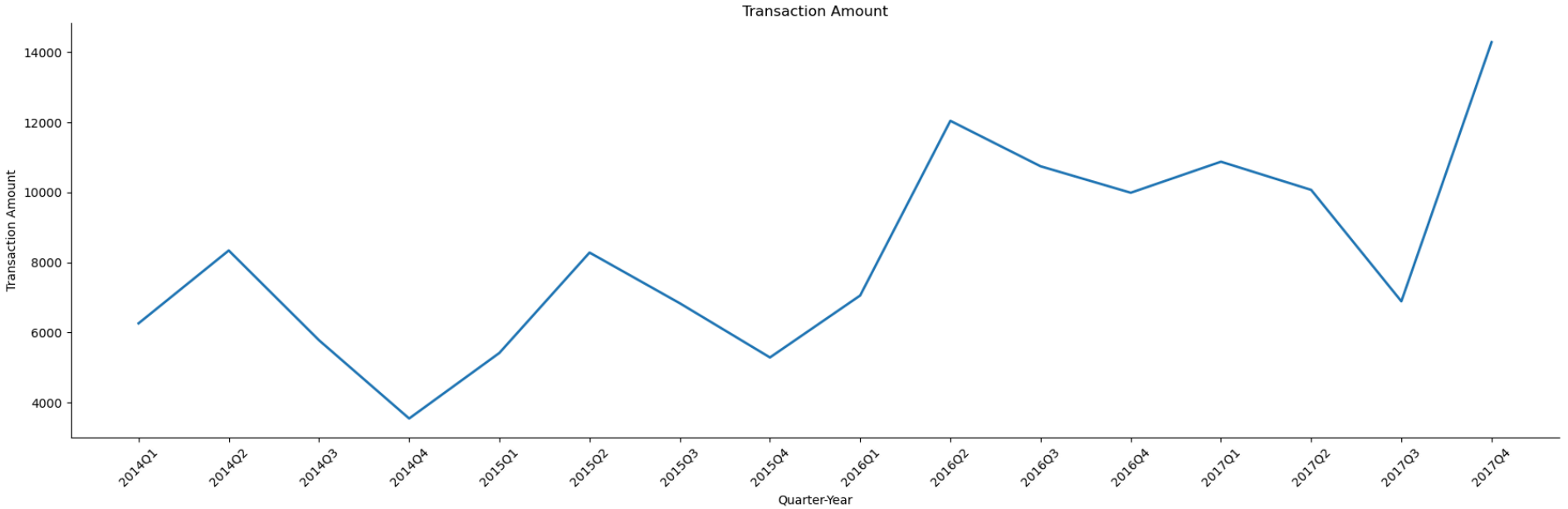

Commercial
Children's Family Services
Commissioning
Streetscene
Parking & Infrastructure
Children's Education & Skills
Customer Support Group
Assurance
Transaction Amount
600000
500000
400000
300000
200000
100000
0



In [56]:
```
spike_plot(df_filtered,'year','JV Value')
```



In [57]:
```
spike_plot(df_filtered,'Quarter','JV Value')
```



Question 2 part 1

Account

In [ ]:
```
spike_plot(df_filtered,'Quarter','JV Value',col_temp='Account Description')
```

In [59]:
```
#observation through z-score and percentage differnce in the transaction amount
```

In [60]:
```
sp_const=5
per_const=2
```

In [61]:
```
df['zscore'] = df.groupby(['Service Area', 'Account Description', 'QT_validation'])['JV Value'].transform(lambda x: (x - x.mean())
#artbitary value of 5 as treshold
z_score_spike=df[df['zscore']>sp_const]
z_score_non_spike=df[df['zscore']<per_const]

#using percentage differnce bwt consecutive values
#first date is ordered in ascending
df_sorted = df.groupby(['Service Area', 'Account Description']).apply(lambda x: x.sort_values('Transaction Date')).reset_index(dro
df_sorted['%change'] = df_sorted.groupby(['Service Area', 'Account Description'])['JV Value'].pct_change()


percent_spike=df_sorted[df_sorted['%change']>sp_const]
percent_non_spike=df_sorted[df_sorted['%change']<per_const]
```

Question 2 part 2

segmenting spike and permanent changes

In [62]:
```
z_score_spike[z_score_spike["Service Area"]=="Childrens Services"][['JV Value','Service Area','zscore']].head()
```

Out[62]:

	JV Value	Service Area	zscore
322	32943.0	Childrens Services	5.207716
494	68664.0	Childrens Services	5.493995
500	196000.0	Childrens Services	6.986634
673	12999.0	Childrens Services	7.431291
758	45672.0	Childrens Services	6.695405

In [63]:
```
percent_spike[percent_spike["Service Area"]=="Childrens Services"][['JV Value','Service Area','%change']].head()
```

Out[63]:

	JV Value	Service Area	%change
9278	5997.0	Childrens Services	855.714286
9280	2274.0	Childrens Services	5.497143
9285	433.0	Childrens Services	47.111111
9286	6944.0	Childrens Services	15.036952
9290	29789.0	Childrens Services	11.991278

In [64]:
```
percent_non_spike[percent_non_spike["Service Area"]=="Childrens Services"][['JV Value','Service Area','%change']].head()
```

Out[64]:

	JV Value	Service Area	%change
9275	222.0	Childrens Services	-0.555110
9277	7.0	Childrens Services	-0.989781
9279	350.0	Childrens Services	-0.941637
9281	249.0	Childrens Services	-0.890501
9282	252.0	Childrens Services	0.012048

In [65]:
```
z_score_non_spike[z_score_non_spike["Service Area"]=="Childrens Services"][['JV Value','Service Area','zscore']].head()
```

Out[65]:

	JV Value	Service Area	zscore
1	600000.0	Childrens Services	1.788670
2	30938.0	Childrens Services	0.707107
3	21876.0	Childrens Services	-0.707107
4	306.0	Childrens Services	-0.356084
5	732.0	Childrens Services	-0.471575

++

Question 4

similarity I

In [66]:
```
#creating pipeline for clustering the service area by transaction count and total
```

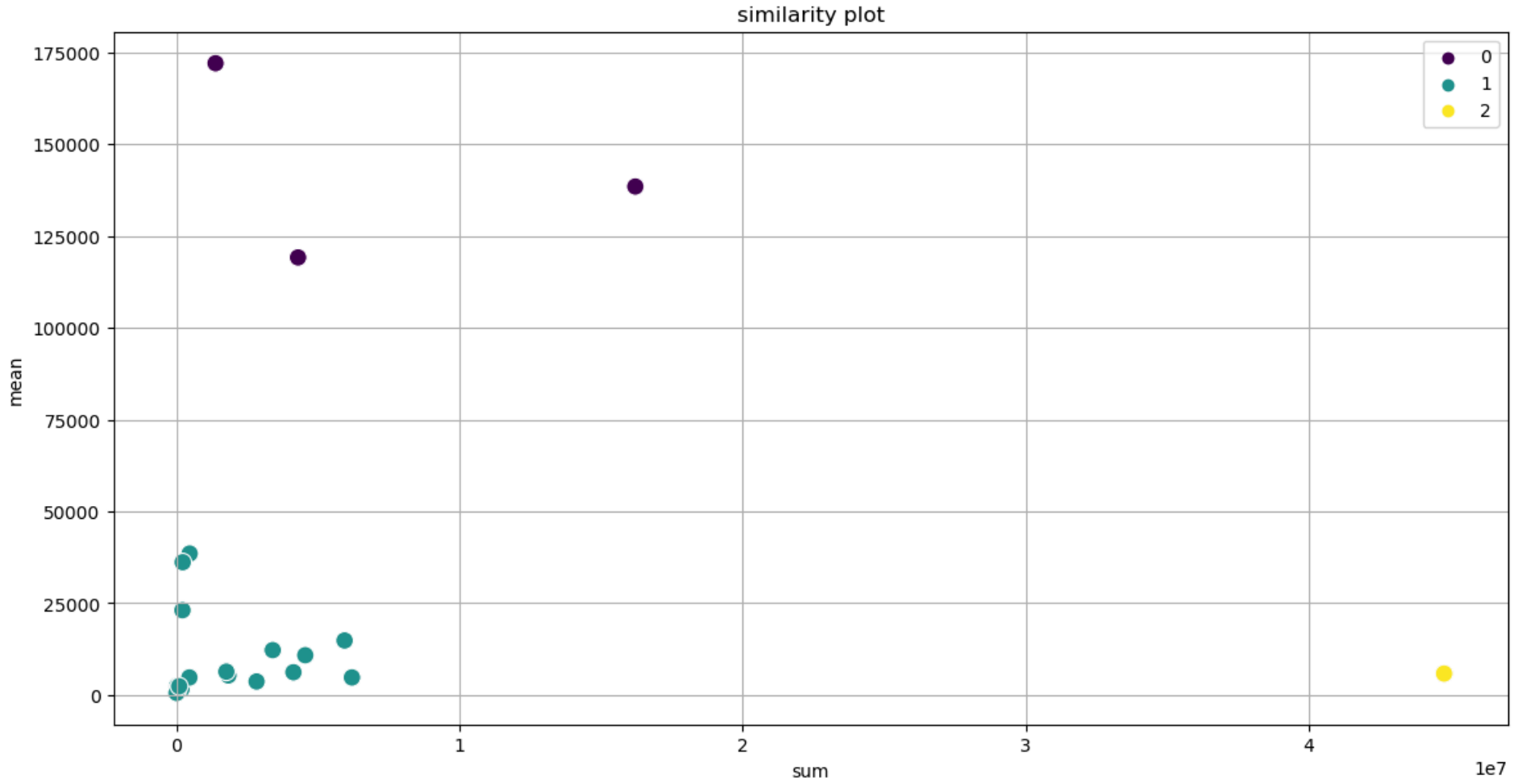
In [67]:
```
def similarity(data,col,val,k):
    """
    arg:data--> dataframe
    arg:col--> column which has to be clustered
    arg:val--> to be clustered of
    arg:k--> cluster number for kmeans
    function: to create a clustering model
    returns dataframe with culster number
    """
    temp=data.groupby(col)[val].agg(['count', 'sum', 'mean', 'median', 'std']).reset_index()
    temp=temp.dropna()
#     temp = temp.replace([np.inf, -np.inf], np.nan).dropna()
    std= StandardScaler()
    scaled=std.fit_transform(temp.drop(col, axis=1))
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(scaled)
    temp['cluster_id']=kmeans.labels_

    plt.figure(figsize=(14,7))
    sns.scatterplot(data=temp,x='sum',y='mean',hue='cluster_id',palette='viridis',s=100)
    plt.legend()
    plt.title("similarity plot")
```

```
    plt.grid(True)
    plt.show()


#     plt.figure(figsize=(10, 6))
# sns.scatterplot(x='sum', y='sum', hue='cluster_id', data=a, palette='viridis', s=100)
# plt.title('Service Areas Grouped by Spending Behavior')
# plt.xlabel('Average Transaction Amount')
# plt.ylabel('Total Transaction Amount')
# plt.legend(title='Cluster')
# plt.grid(True)
# plt.show()
    return temp,scaled
```

In [68]:
```
def hac(data,scale,link,col):
    """
    arg: data--> dataframe
    arg: scaled--> scaled numeric value
    arg: linkage method
    arg:leaves of dendogram by which column
    functon: creating HAC plot
    return: None
    """
    cal_temp=linkage(scale,method=link)
    plt.figure(figsize=(14,7))
    dendrogram(cal_temp,labels=data[col].values,leaf_rotation=90)
    plt.title('HAC')
    plt.xlabel(col)
    plt.show()
```

In [ ]:
```

```

In [69]:
```
clus_analysis,scaled=similarity(df,'Service Area','JV Value',3)
```



In [94]:
```
clus_analysis[:10].T
```

Out[94]:

	0	1	2	3	4	5	6	7	8
Service Area	Adults and Communities	Assurance	CSG Managed Budget	Children's Education & Skills	Children's Family Services	Children's Service DSG	Childrens Services	Commercial	Commissioning
count	278	344	36	667	7672	277	1297	9	400
sum	3396782.0	1829415.0	4290803.0	4128956.0	44773136.0	1757715.0	6194487.0	207786.0	5934935.0
mean	12218.640288	5318.06686	119188.972222	6190.338831	5835.914494	6345.541516	4776.011565	23087.333333	14837.3375
median	1831.0	900.0	4564.0	1385.0	1797.5	1894.0	675.0	594.0	4612.0
std	36217.416042	27137.891992	213468.294515	12512.61528	12028.966756	13725.713997	20421.416626	37508.583144	37451.003617
cluster_id	1	1	0	1	2	1	1	1	1

In [71]:
```
clus_analysis[clus_analysis['cluster_id']==0]
```

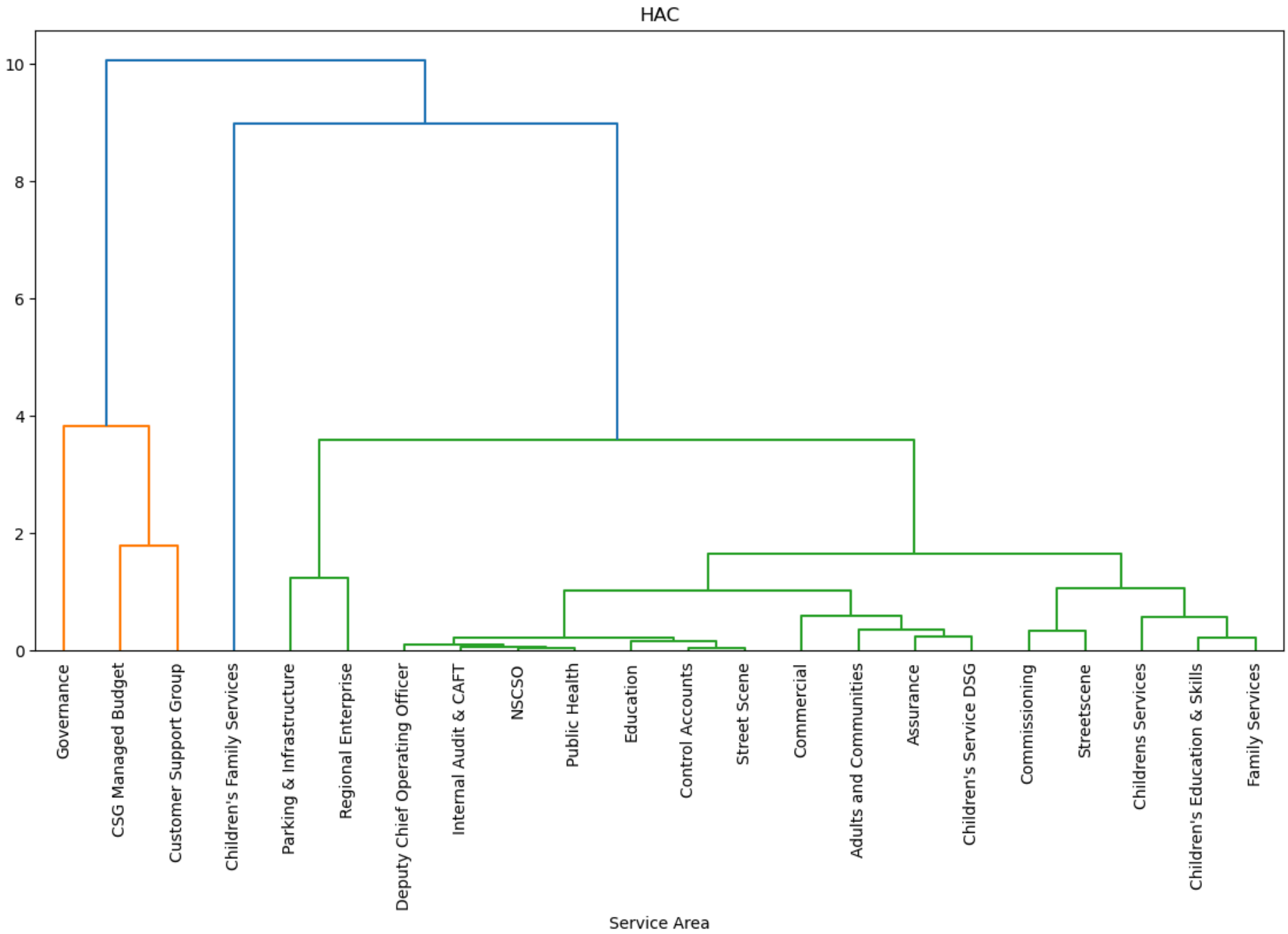
Out[71]:

	Service Area	count	sum	mean	median	std	cluster_id
2	CSG Managed Budget	36	4290803.0	119188.972222	4564.0	213468.294515	0
10	Customer Support Group	117	16204913.0	138503.529915	10399.0	267847.540772	0
14	Governance	8	1376694.0	172086.750000	24943.5	288737.362988	0

Question 4

similarity II

In [72]:
```
hac(clus_analysis,scaled,'ward','Service Area')
```

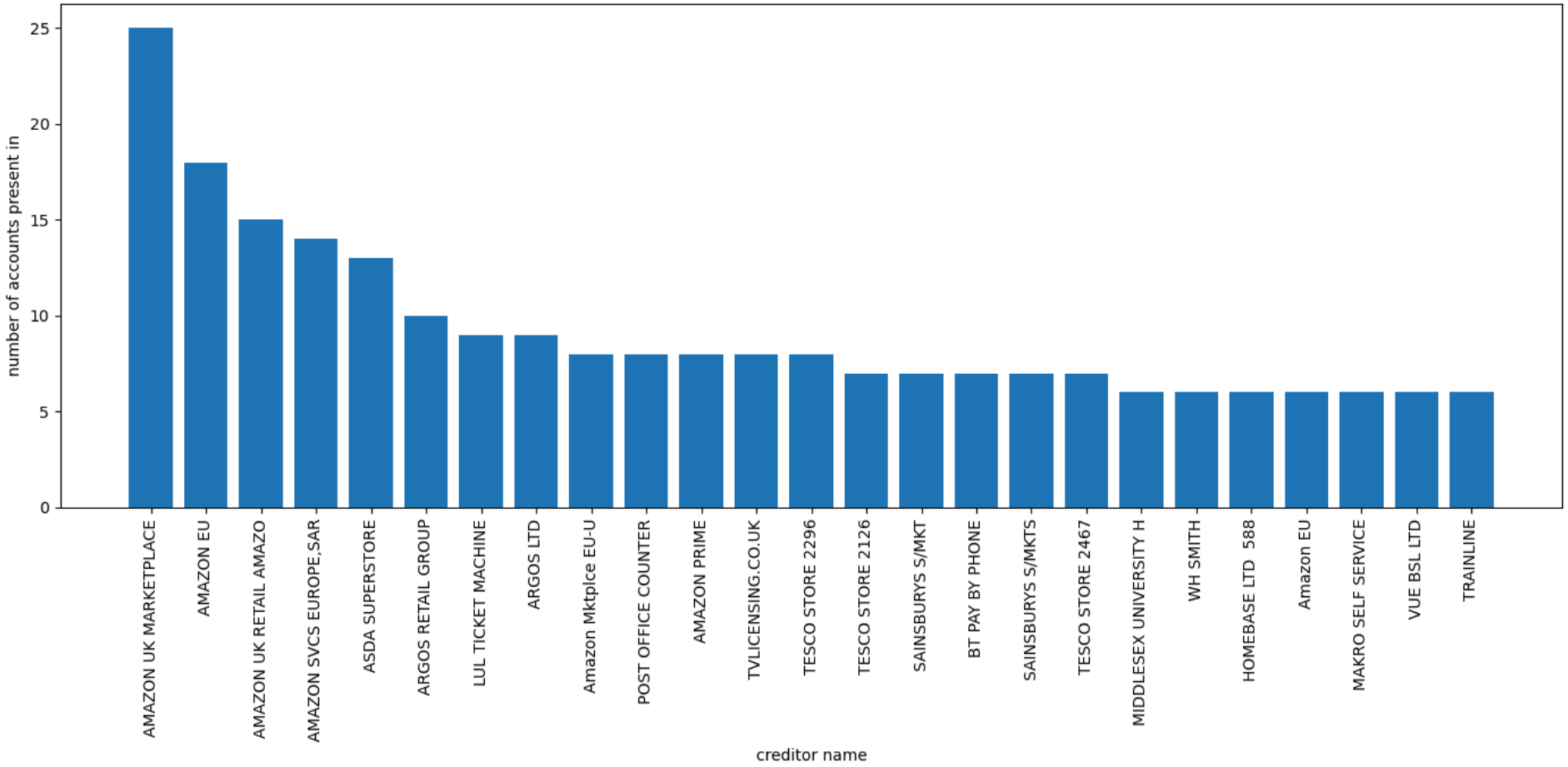

HAC
10
8
6
4
2
0
Governance
CSG Managed Budget
Customer Support Group
Children's Family Services
Parking & Infrastructure
Regional Enterprise
Deputy Chief Operating Officer
Internal Audit & CAFT
NSCSO
Public Health
Education
Control Accounts
Street Scene
Commercial
Adults and Communities
Assurance
Children's Service DSG
Commissioning
Streetscene
Childrens Services
Children's Education & Skills
Family Services
Service Area

++

Question 3

In [73]:
```
def repeating_values(data,check_col,on_col,repeat):
    """
    arg:data-->dataframe
    arg:check_col-->column to be looked for misclassifications
    arg:on_col-->against which column
    arg:repeat--> repetation allowed
    function: calculating the frequency of the column values
    return: dataframe
    """
    grp=data.groupby([check_col,on_col]).size().reset_index(name="count")
    frq=grp.groupby(check_col)[on_col].count()
    repeating_cond = frq[frq > 1]
    top_inst=repeating_cond.sort_values(ascending=False)[:25]

    plt.figure(figsize=(14,7))
    plt.bar(top_inst.index,top_inst.values)
    plt.xlabel('creditor name')
    plt.ylabel('number of accounts present in')
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()

    return top_inst,repeating_cond
```

In [74]:
```
top_list,whole_list=repeating_values(df,'Creditor','Account Description',2)
```



In [75]:
```
top_list.head()
```

Out[75]:
```
Creditor
AMAZON UK MARKETPLACE     25
AMAZON EU                 18
AMAZON UK RETAIL AMAZO    15
AMAZON SVCS EUROPE,SAR    14
ASDA SUPERSTORE           13
Name: Account Description, dtype: int64
```

In [76]:
```
whole_list.shape[0] #total of 474 creditors have been mapped into 3 or more accounts
```

Out[76]:
```
474
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

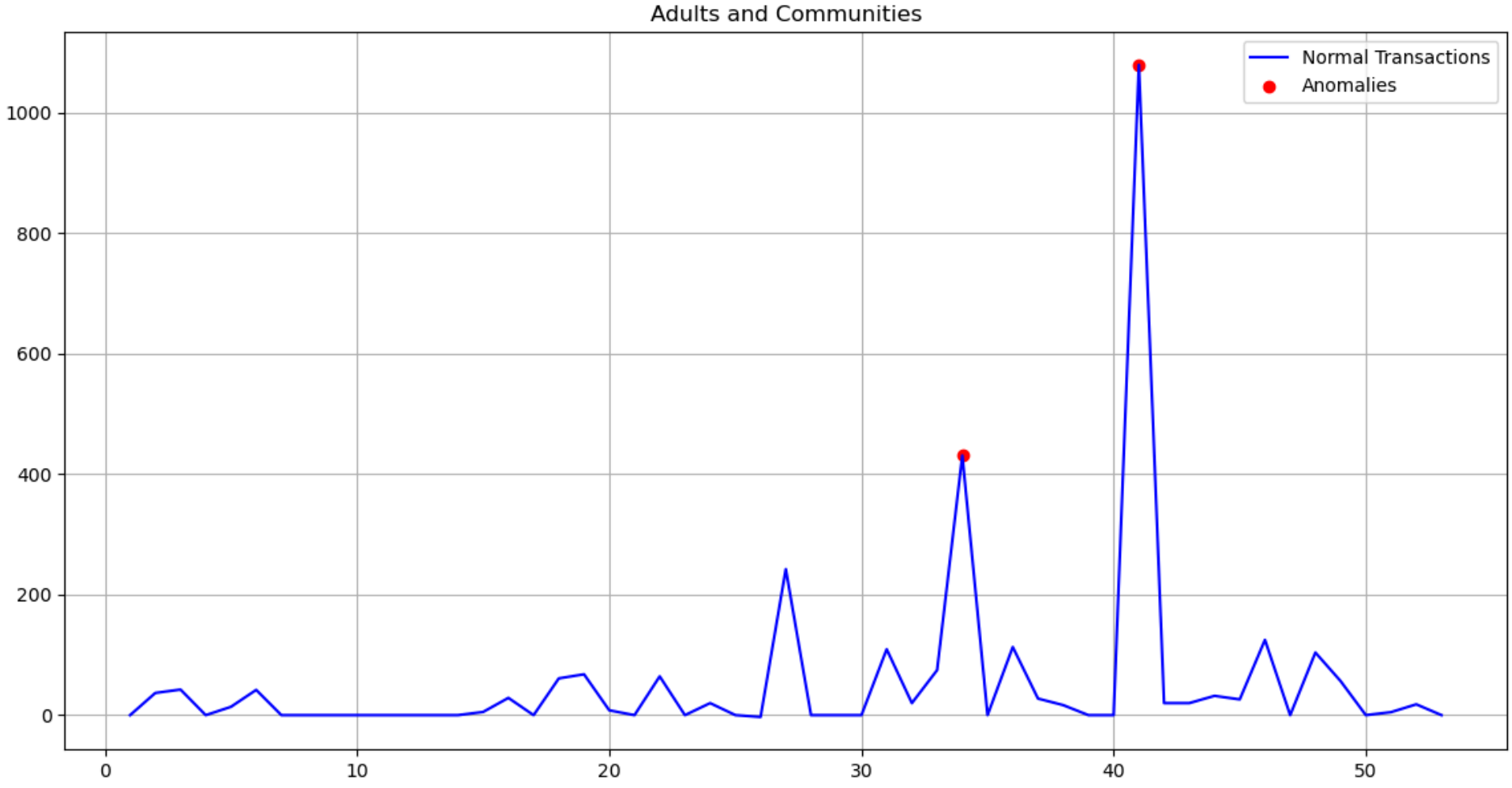
In [77]:
```
##anomaly detection
```

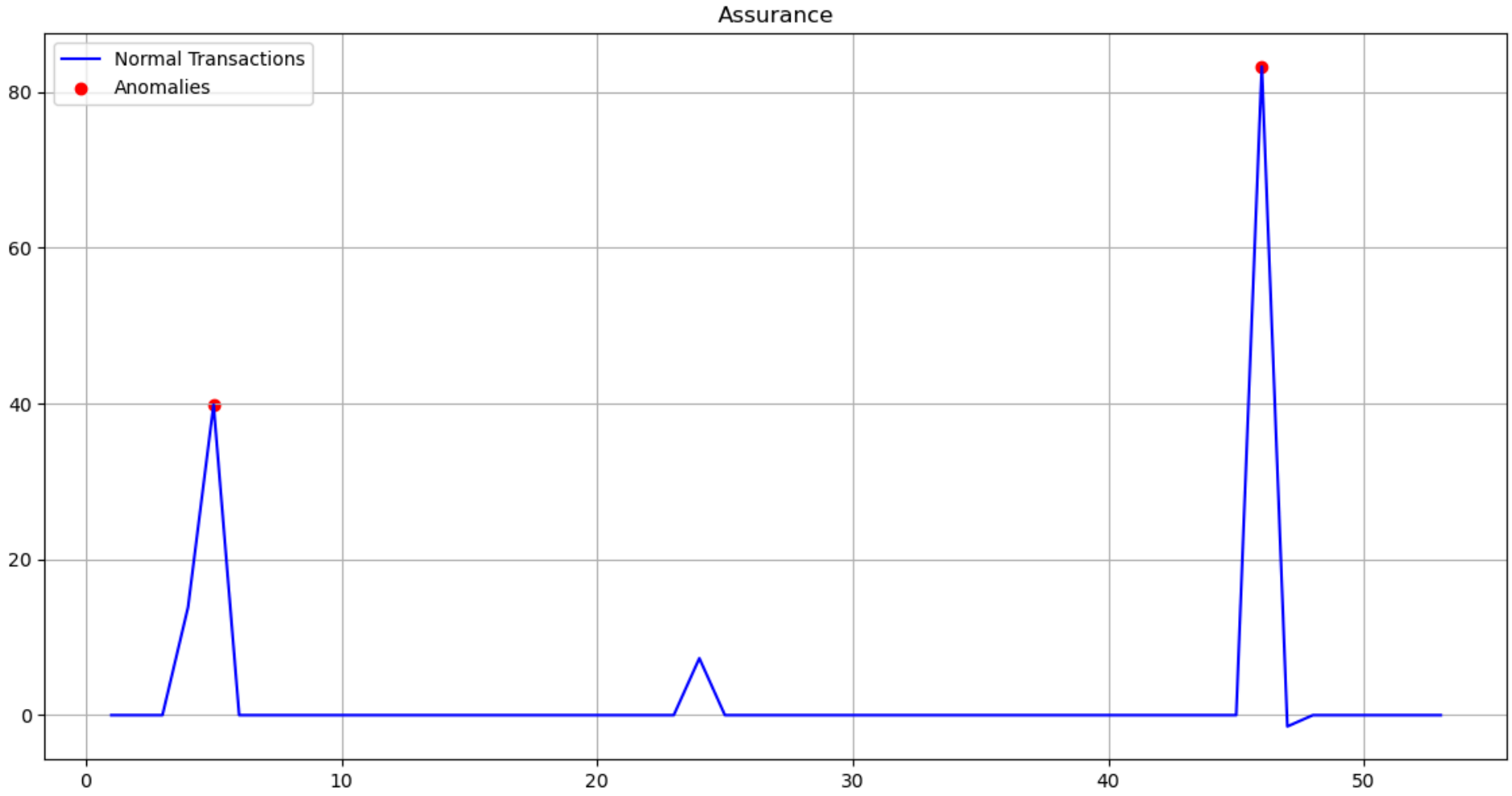
In [78]:
```
#creating dataframe based on weekly transaction sum
def anamoly(data,date_col,grp_col,val_col):
    """
    arg:data--> dataframe to be checked for anomaly
    arg:date_col--> considering anomaly has strong correlation with time ; passing the required data
    arg:grp_col-->which column is checked for the anomaly
    arg:val_col--> which column is used to check the anomaly
    func: basic plots of the anomaly in the transactions weekly
    return: supported dataframe
    """
    data[date_col] = pd.to_datetime(data[date_col])
    data[val_col]=pd.to_numeric(df2[val_col],errors='coerce')
    temp_grp=data.groupby([grp_col,data[date_col].dt.week])
    weekly_values=temp_grp[val_col].sum().reset_index() #
    treshold=2 * weekly_values.groupby(grp_col)[val_col].transform('std') #
    week_anomaly=weekly_values.copy()
    anomaly=week_anomaly[week_anomaly[val_col] > treshold]
    week_anomaly['Anomaly']=week_anomaly[val_col]>treshold#
#     week_anomaly['Anomaly']=week_anomaly[val_col]>threshold#

    for service_area,grp in week_anomaly.groupby(grp_col):
        an_temp=grp[val_col]>treshold[grp.index]
        plt.figure(figsize=(14,7))
        plt.plot(grp[date_col], grp[val_col], label='Normal Transactions', color='blue')
        plt.scatter(grp.loc[an_temp,date_col],grp.loc[an_temp,val_col],color='red',label='Anomalies')
        plt.title(service_area)
        plt.legend()
        plt.grid(True)
        plt.show()


    return weekly_values,week_anomaly,anomaly,treshold
```

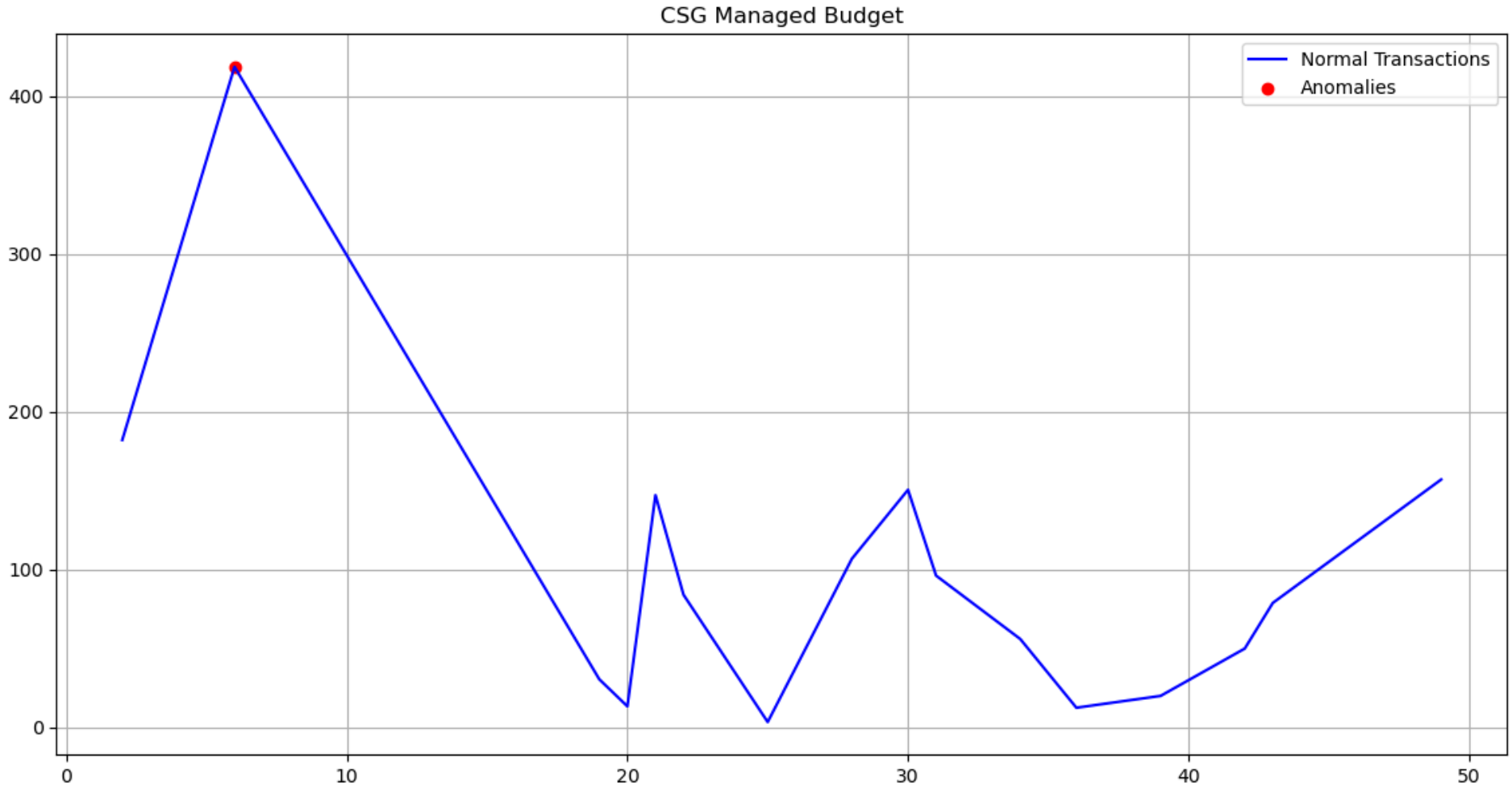
Question 5 part 1

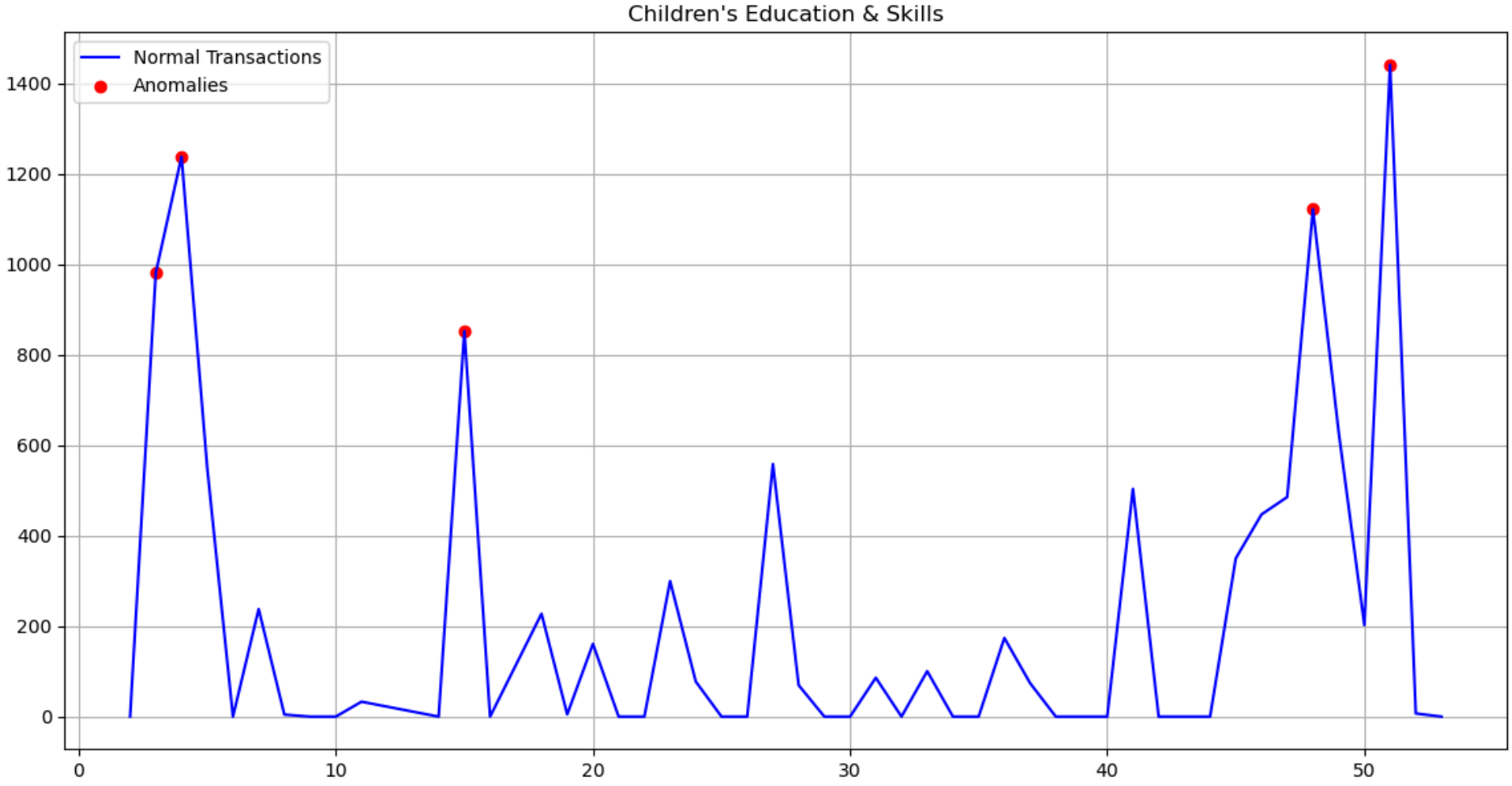
ploting values of abnormal transactions in service area / week

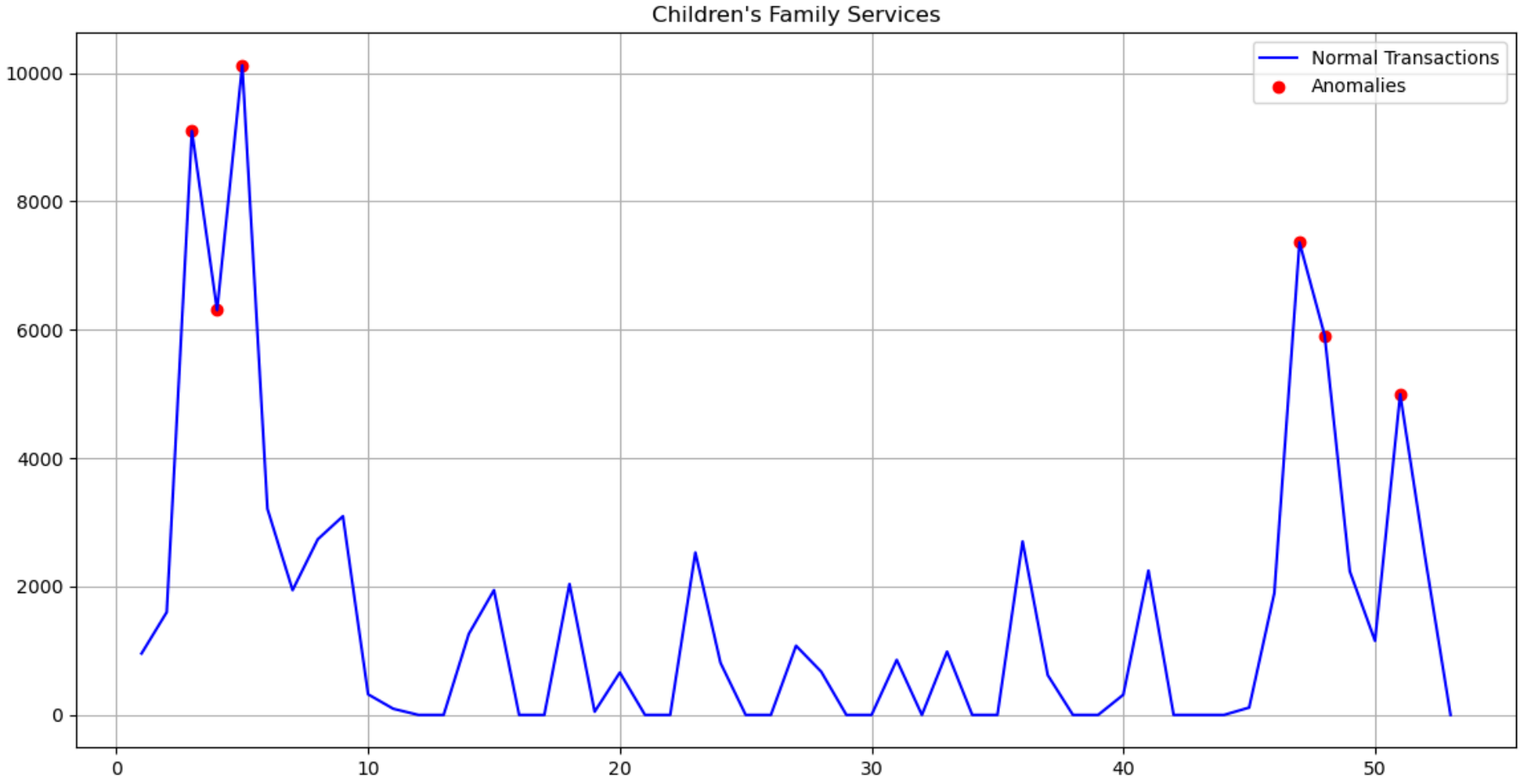
In [79]:
```
weekly_trans_sum,anomaly_encoded_df,only_anomaly,treshold=anamoly(df,'Transaction Date','Service Area','JV Value')
```

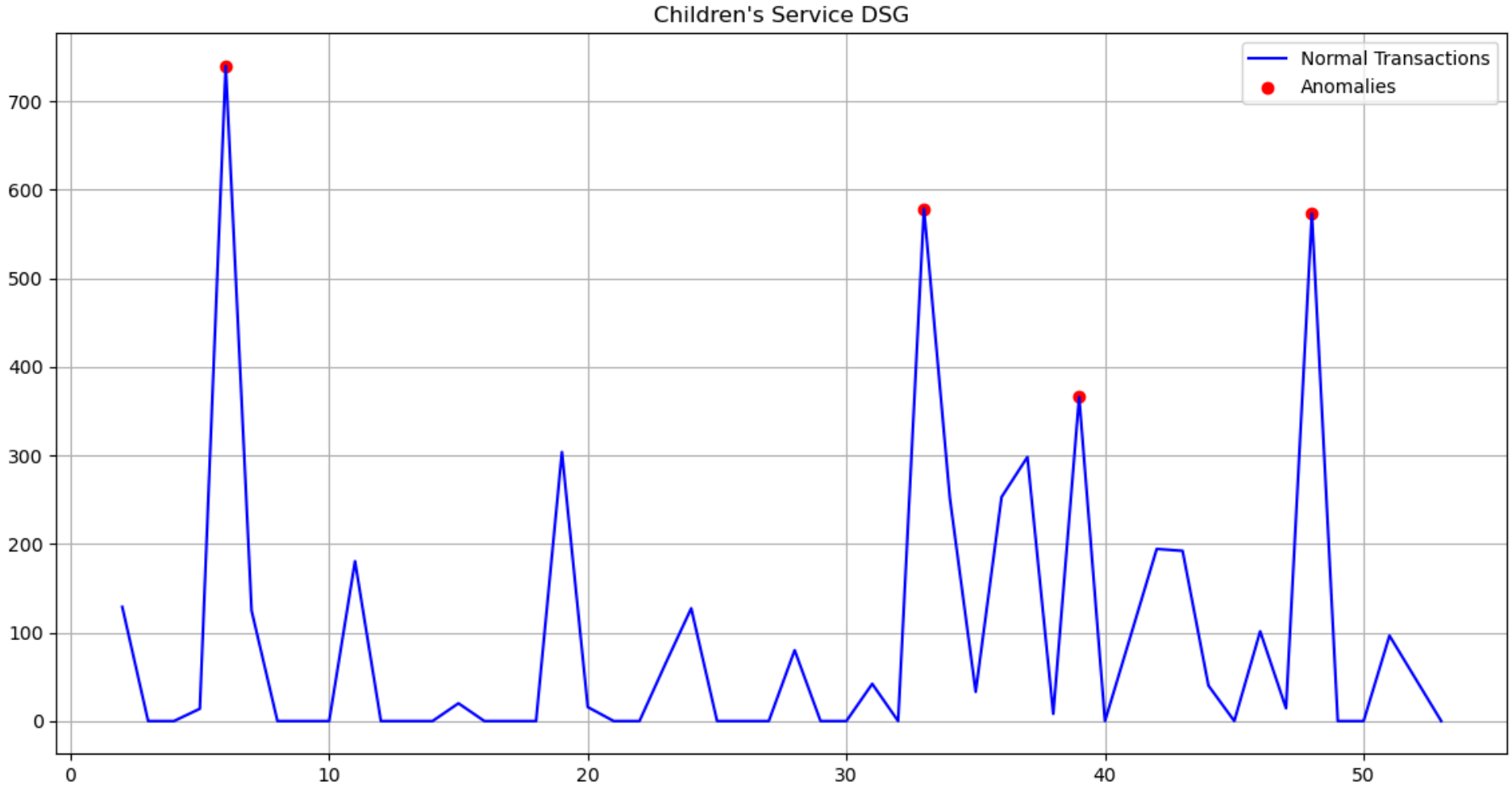

Adults and Communities
Normal Transactions
Anomalies
1000
800
600
400
200
0
0
10
20
30
40
50

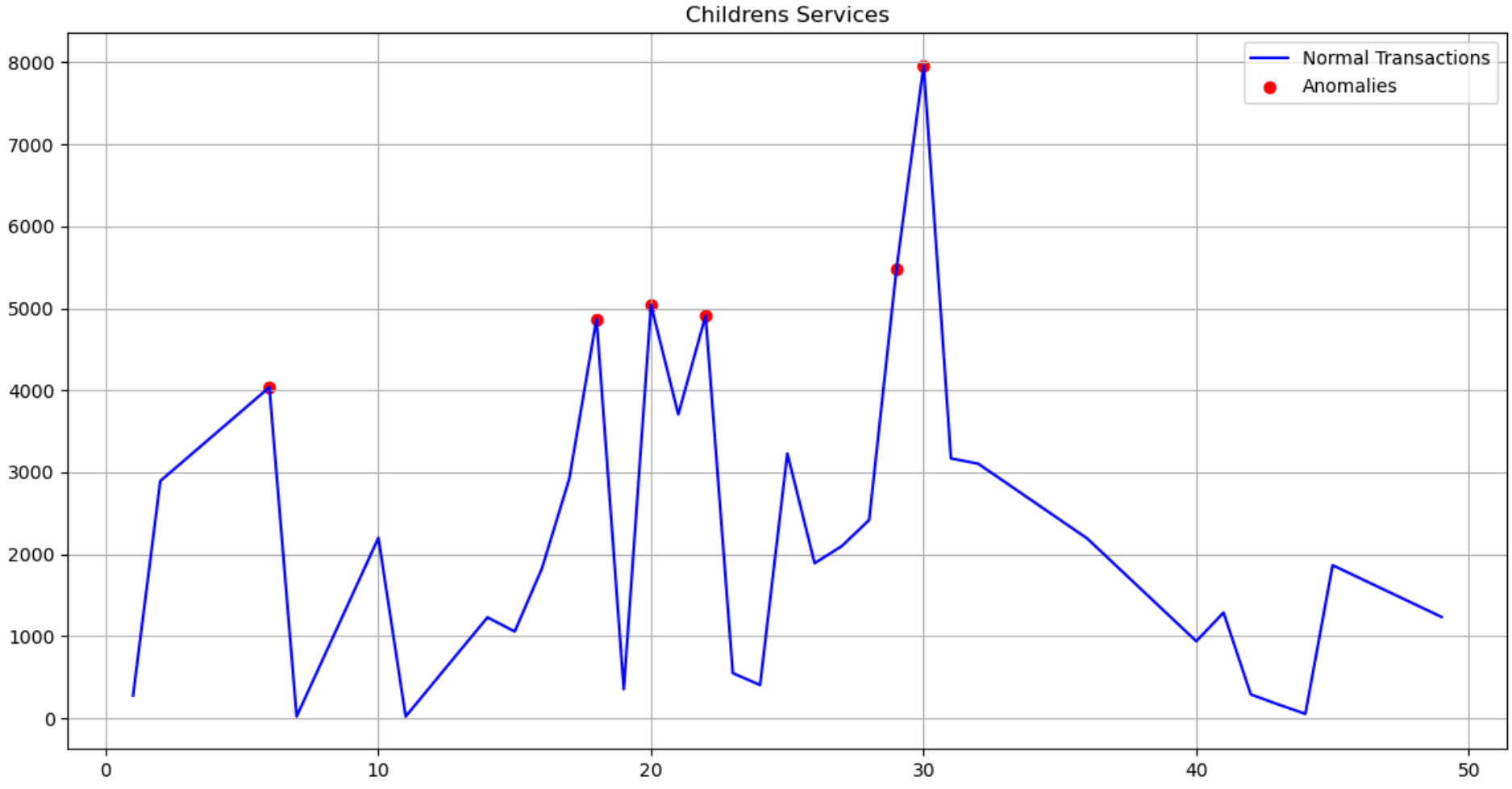

Assurance
Normal Transactions
Anomalies
80
60
40
20
0
0
10
20
30
40
50

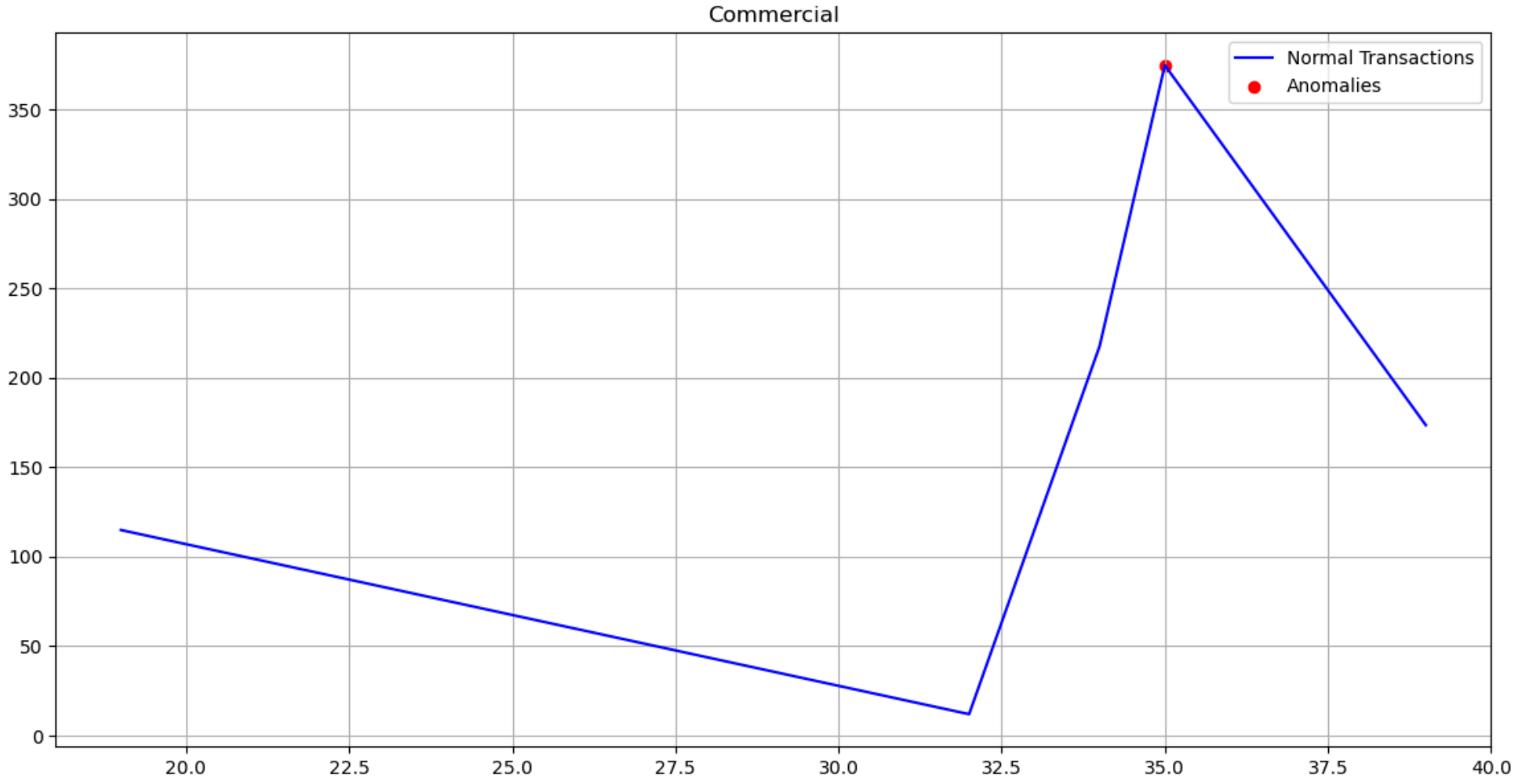

CSG Managed Budget
Normal Transactions
Anomalies
400
300
200
100
0
0
10
20
30
40
50

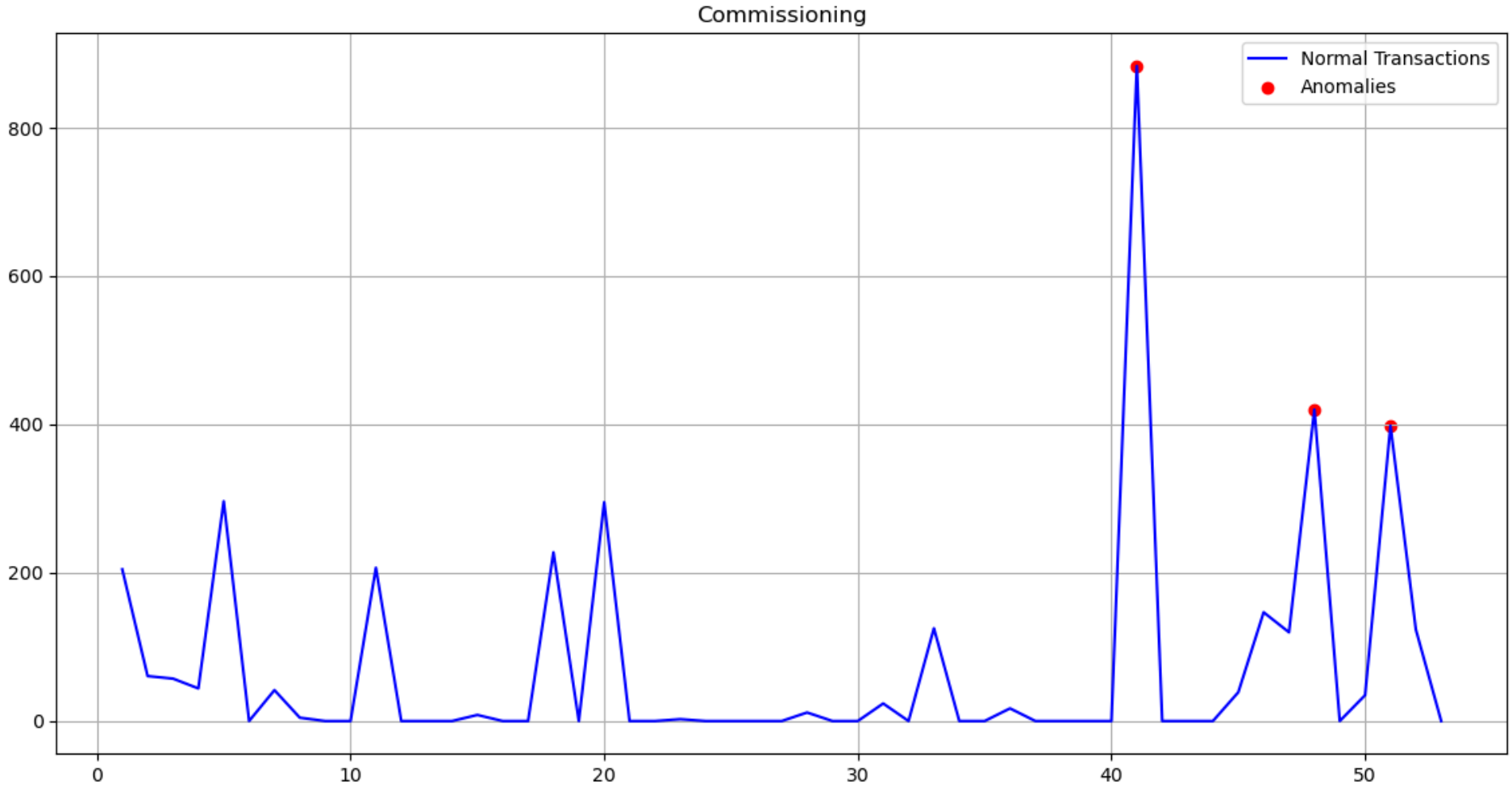

Children's Education & Skills
Normal Transactions
Anomalies
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50

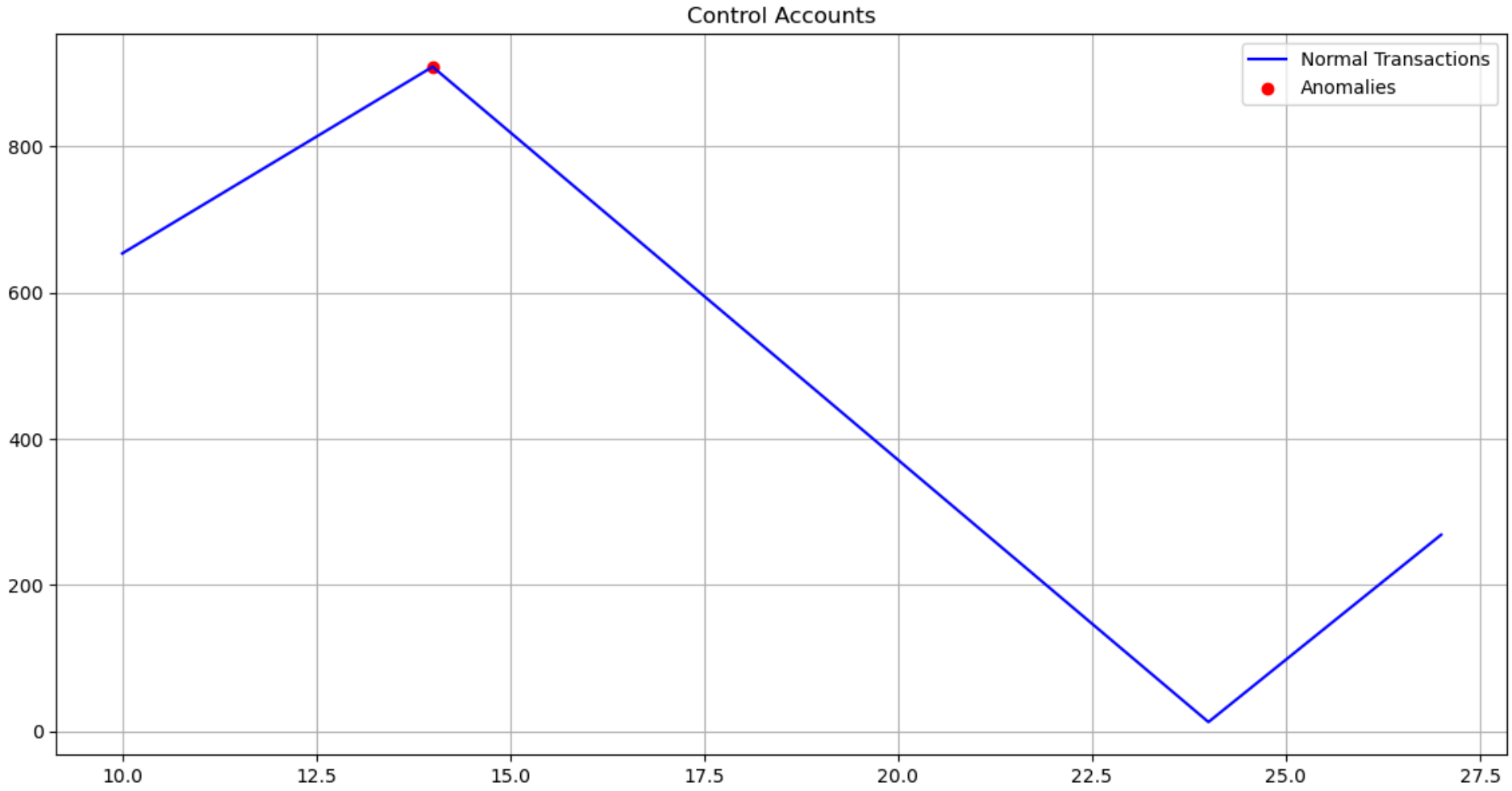

Children's Family Services
Normal Transactions
Anomalies
10000
8000
6000
4000
2000
0
0
10
20
30
40
50

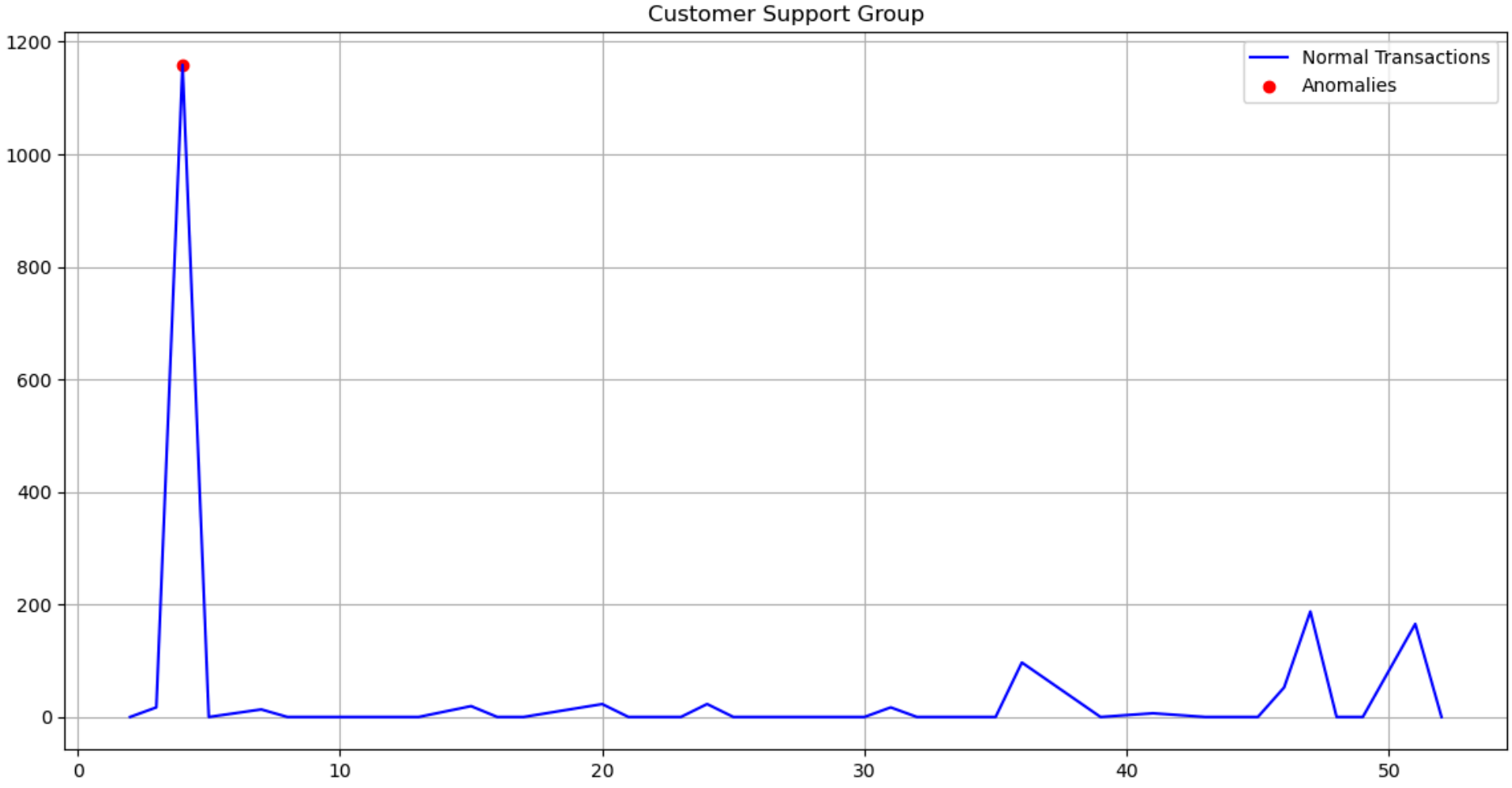

Children's Service DSG
Normal Transactions
Anomalies
700
600
500
400
300
200
100
0
0
10
20
30
40
50

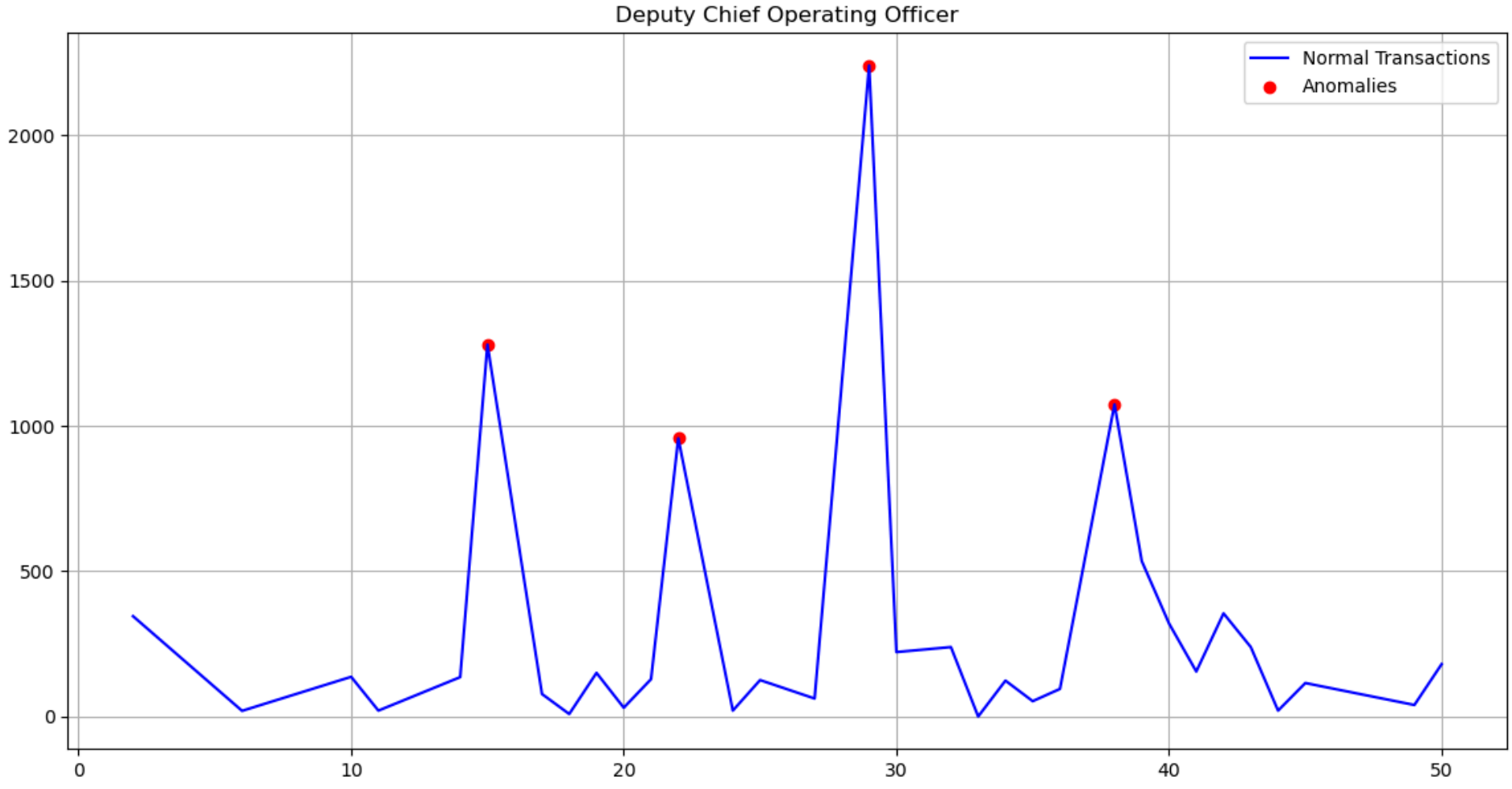

Childrens Services
Normal Transactions
Anomalies
8000
7000
6000
5000
4000
3000
2000
1000
0
0
10
20
30
40
50

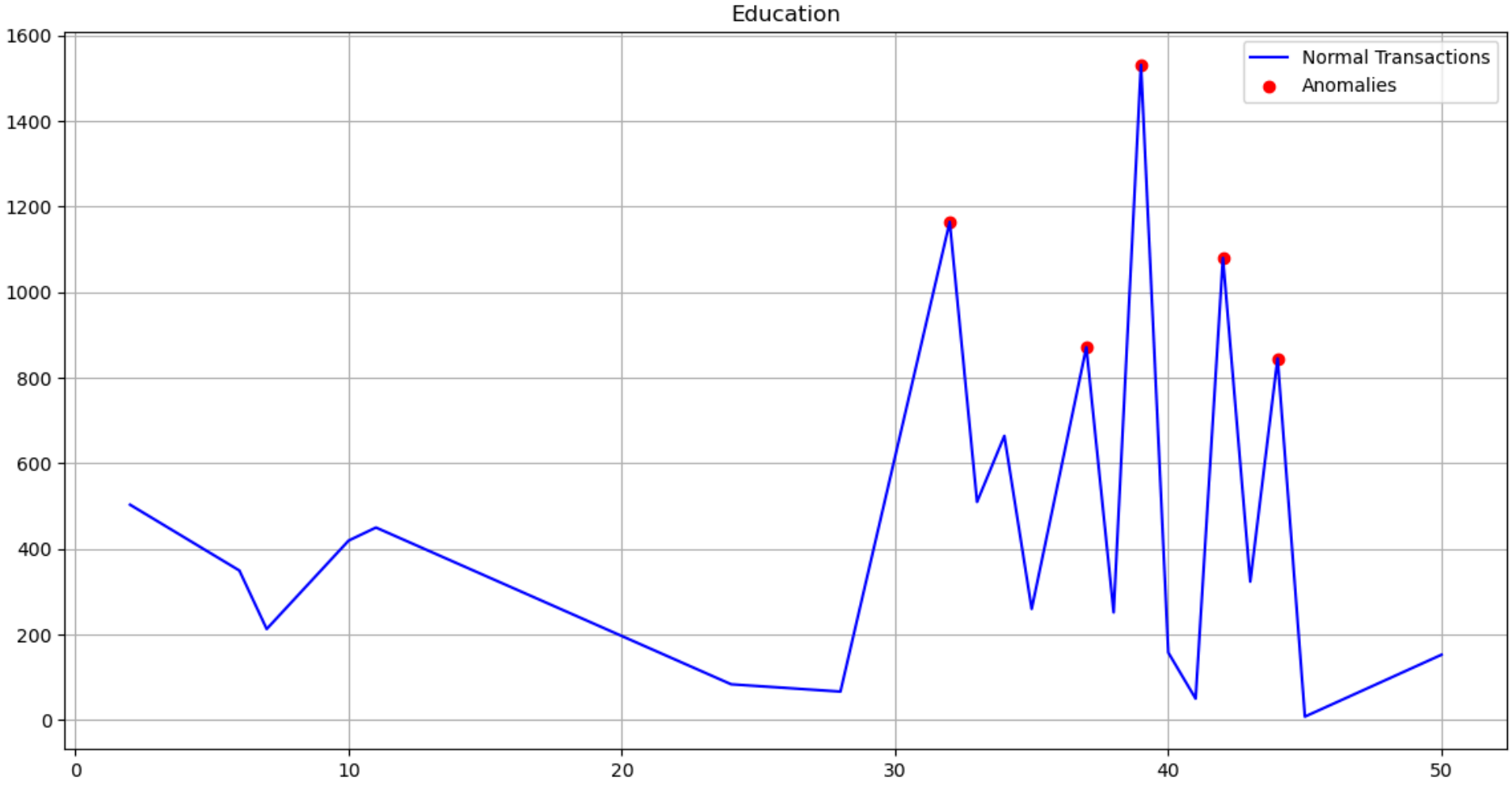

Commercial
Normal Transactions
Anomalies
350
300
250
200
150
100
50
0
20.0
22.5
25.0
27.5
30.0
32.5
35.0
37.5
40.0

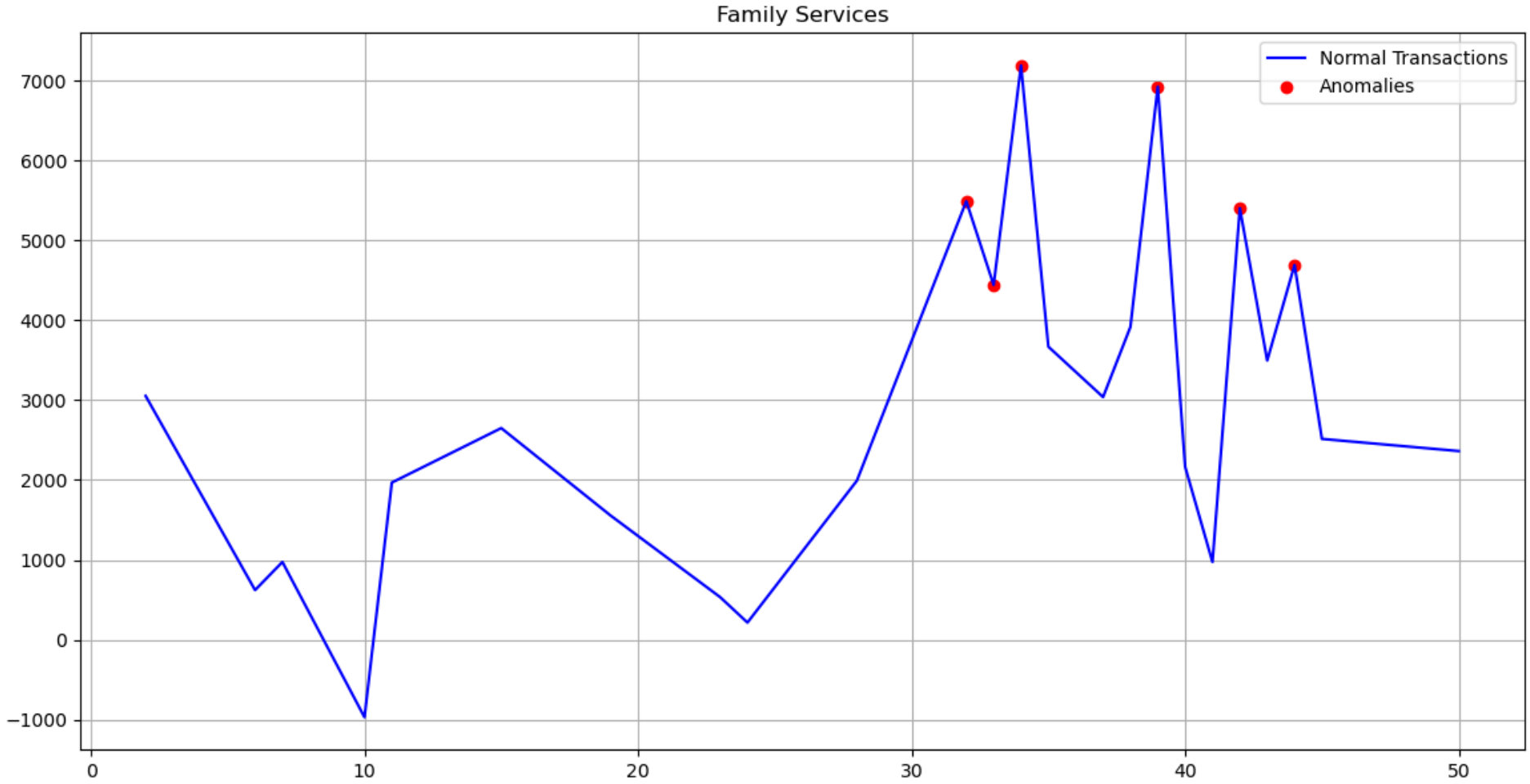

Commissioning
Normal Transactions
Anomalies
800
600
400
200
0
0
10
20
30
40
50

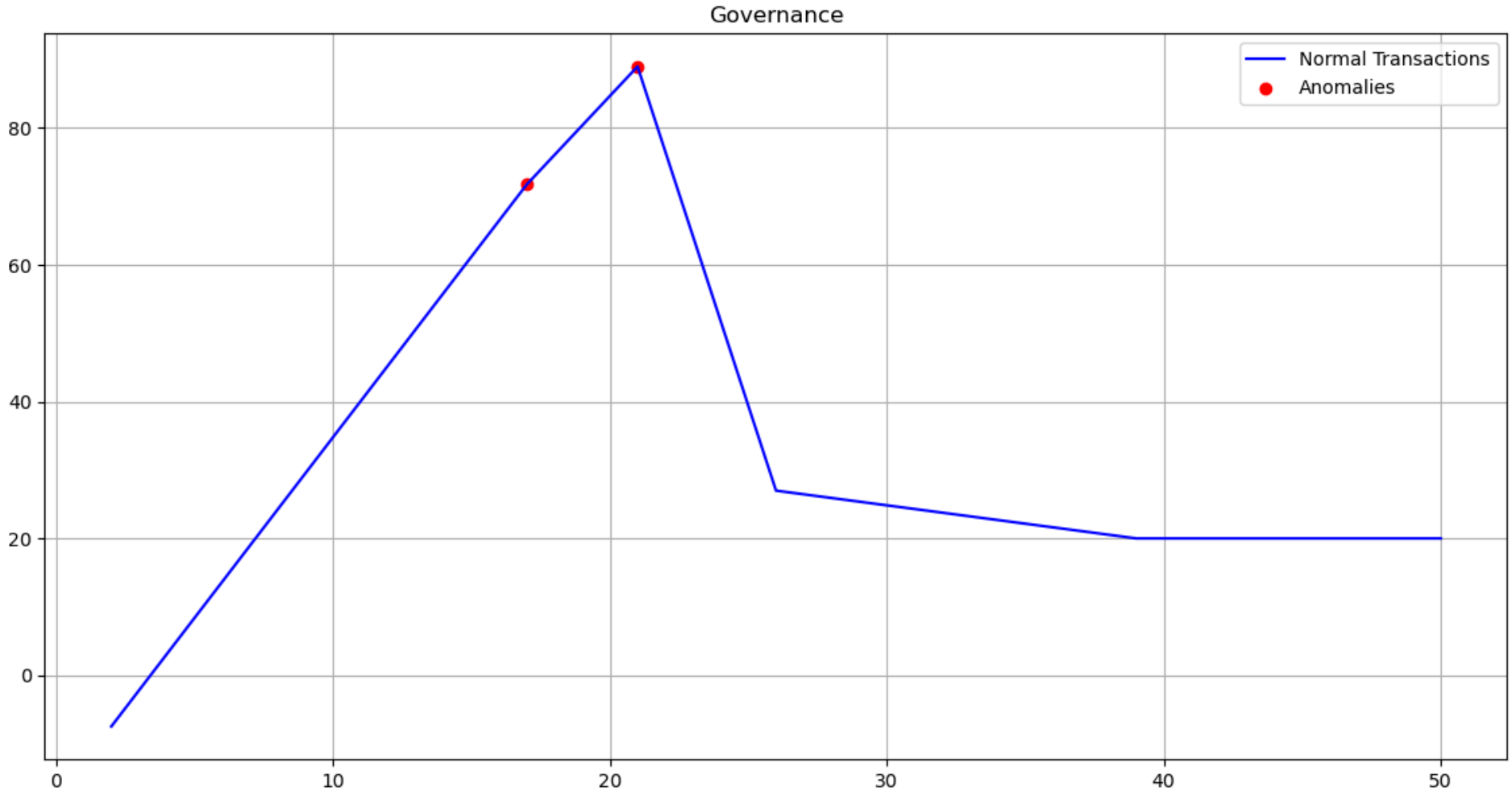

Control Accounts
Normal Transactions
Anomalies
800
600
400
200
0
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5

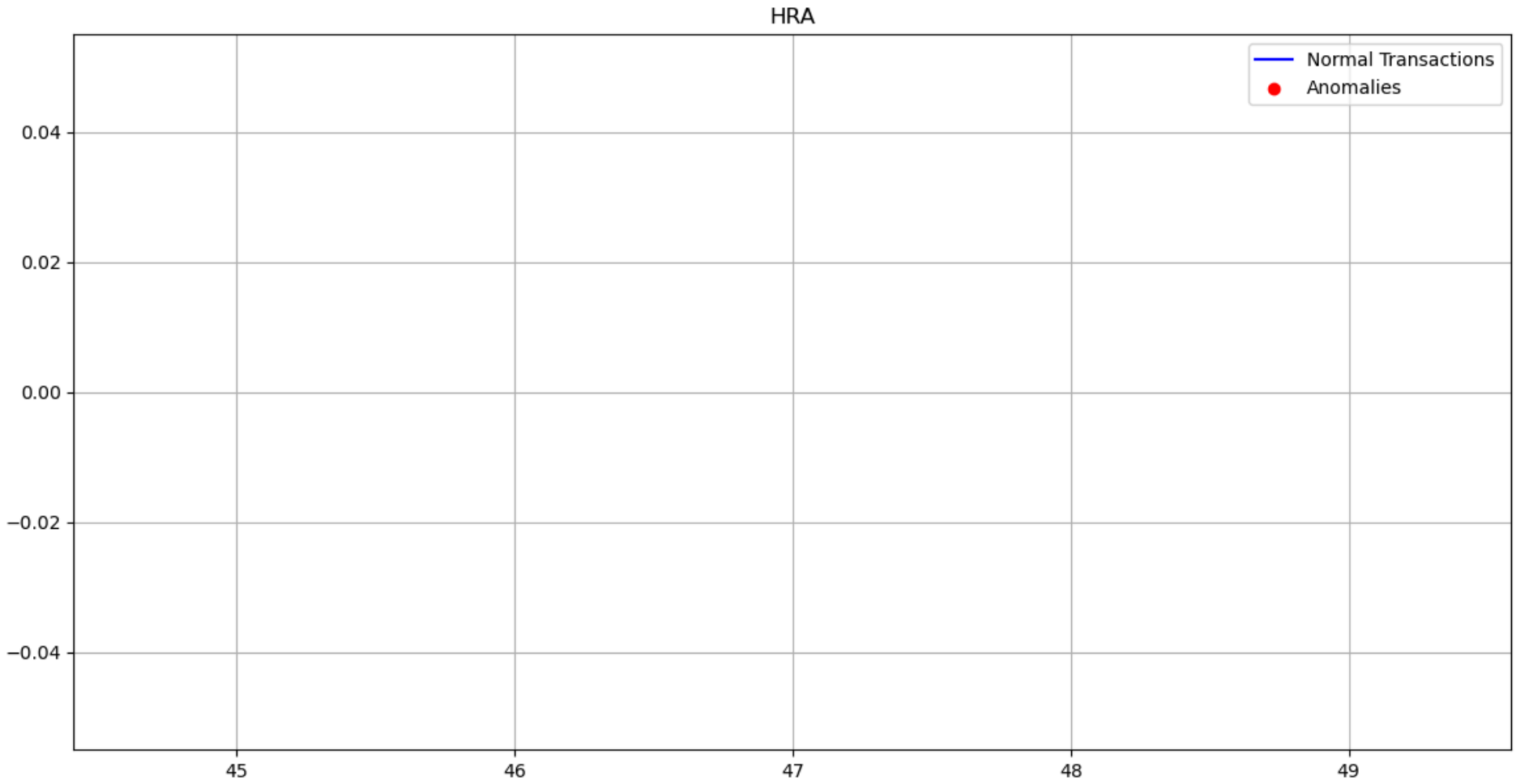

Customer Support Group
Normal Transactions
Anomalies
1200
1000
800
600
400
200
0
0
10
20
30
40
50

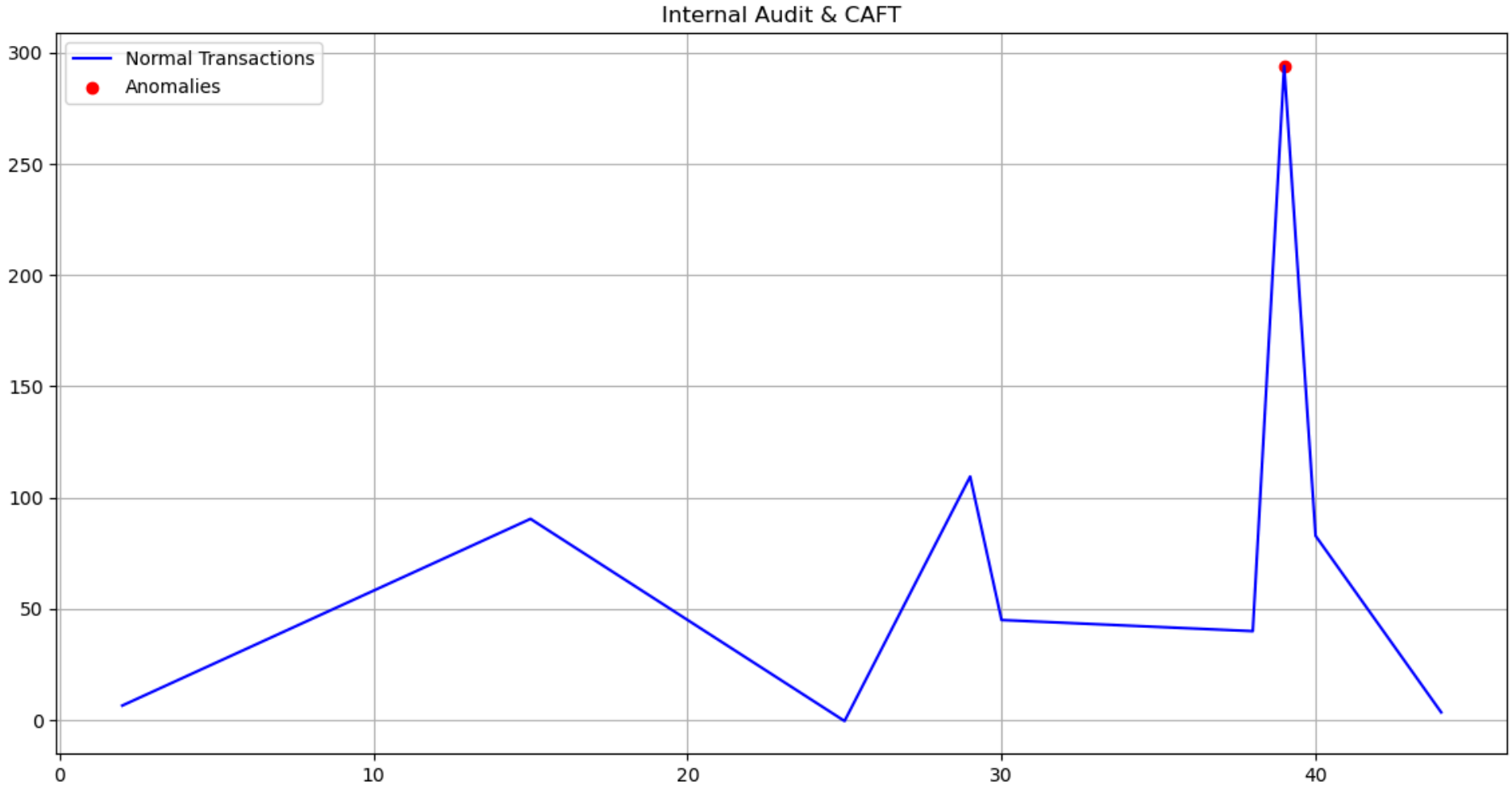

Deputy Chief Operating Officer
Normal Transactions
Anomalies
2000
1500
1000
500
0
0
10
20
30
40
50

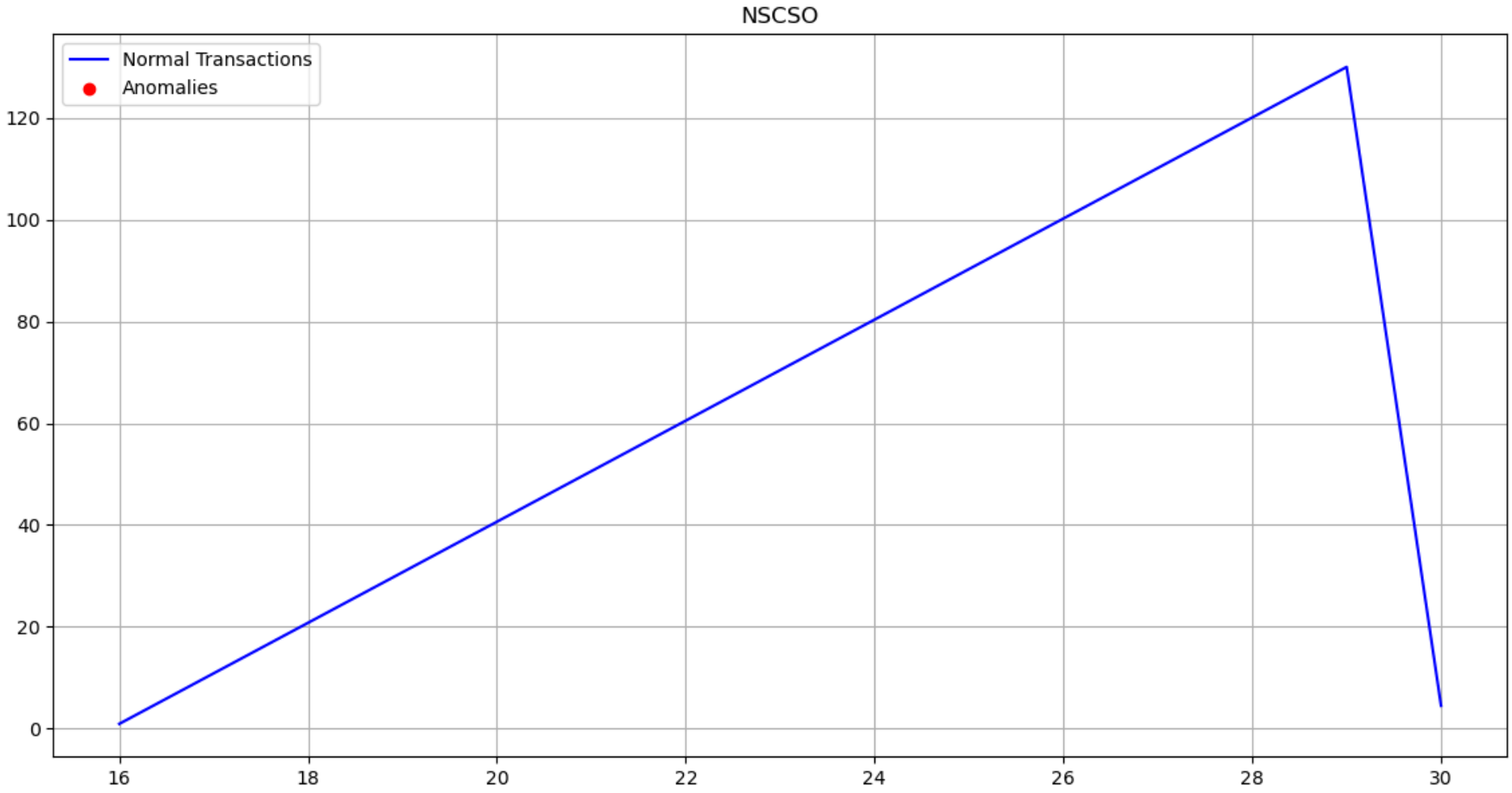

Education
Normal Transactions
Anomalies
1600
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50

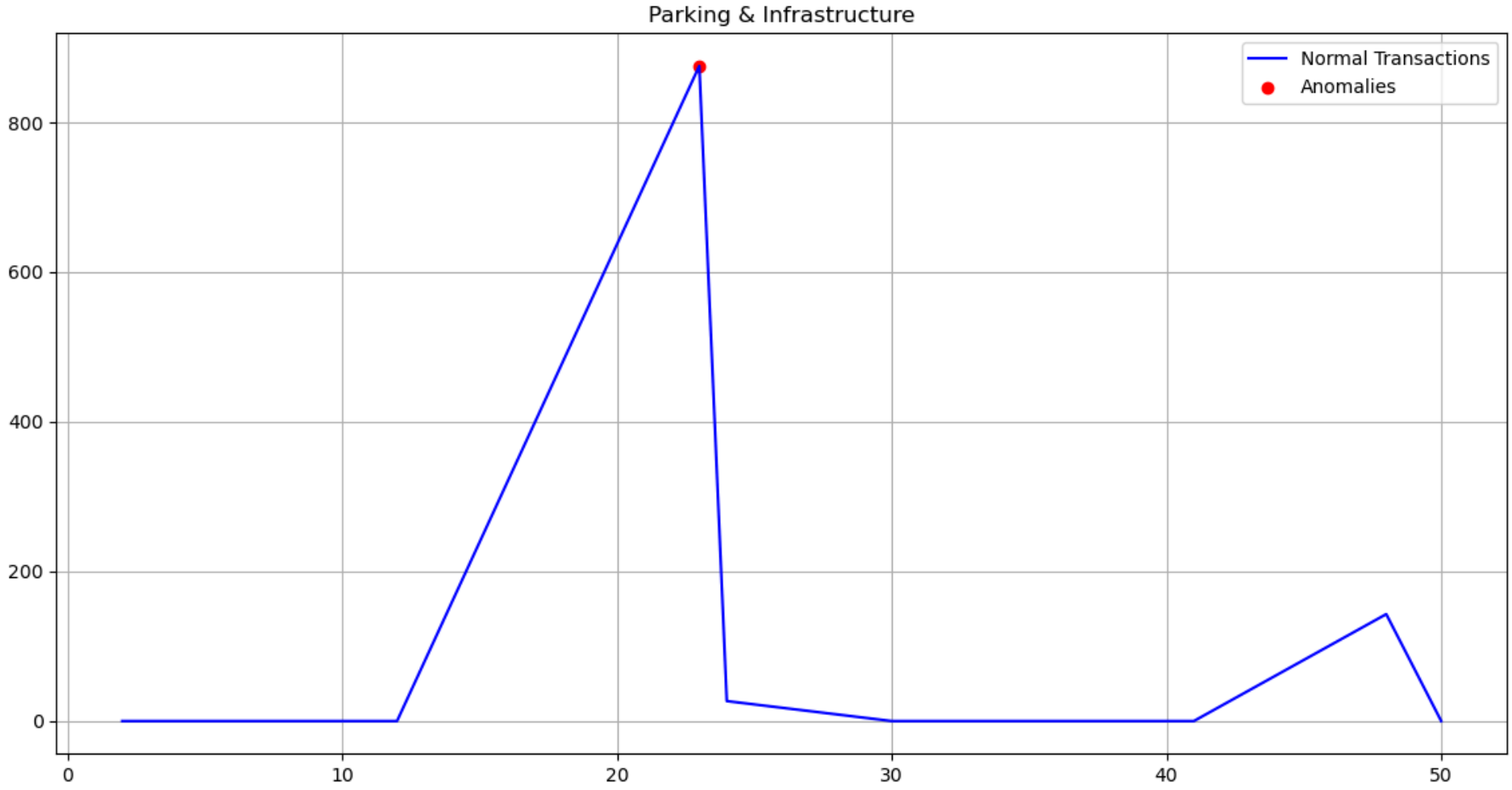

Family Services
Normal Transactions
Anomalies
7000
6000
5000
4000
3000
2000
1000
0
−1000
0
10
20
30
40
50

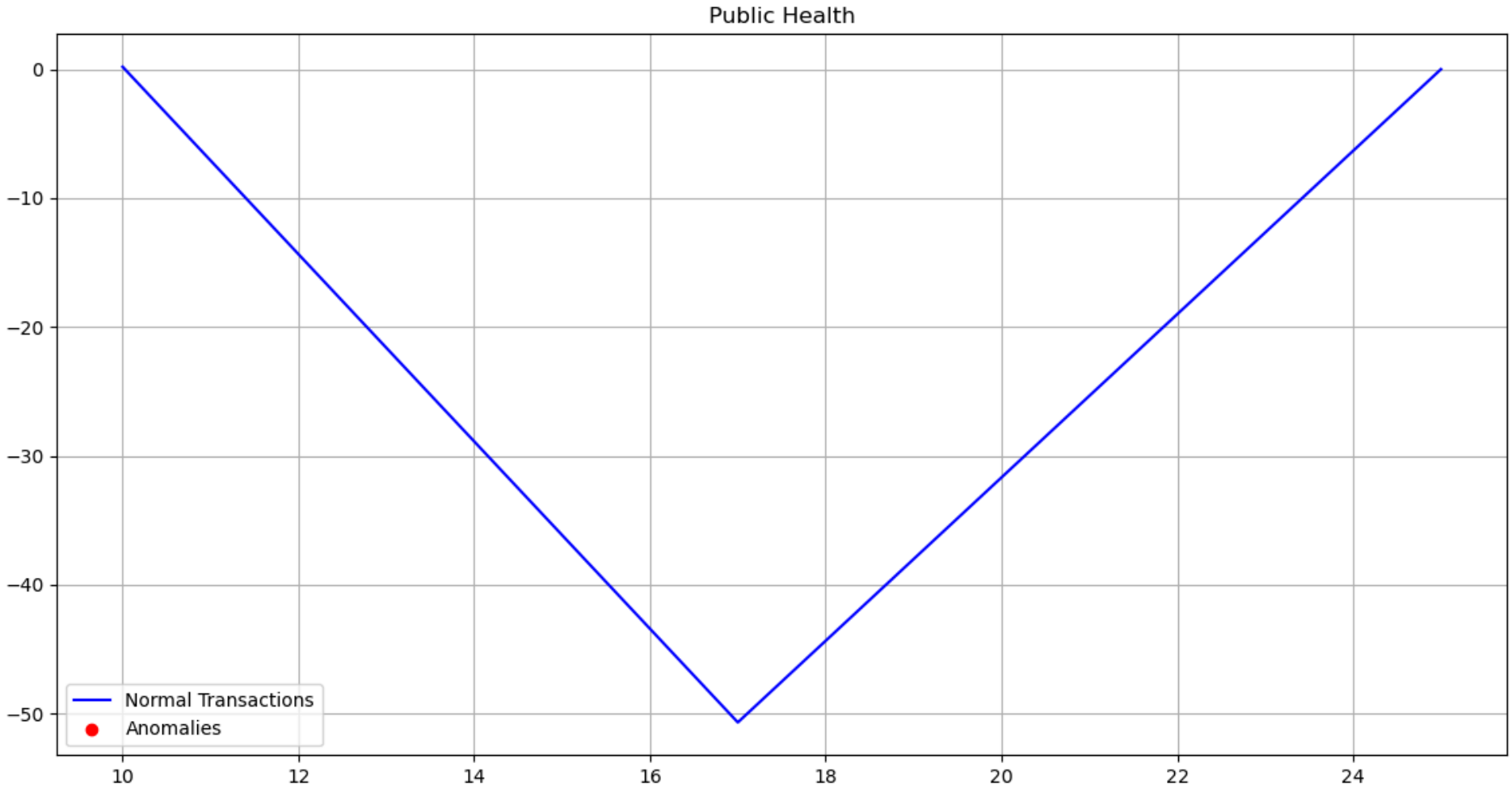

Governance
Normal Transactions
Anomalies
80
60
40
20
0
0
10
20
30
40
50

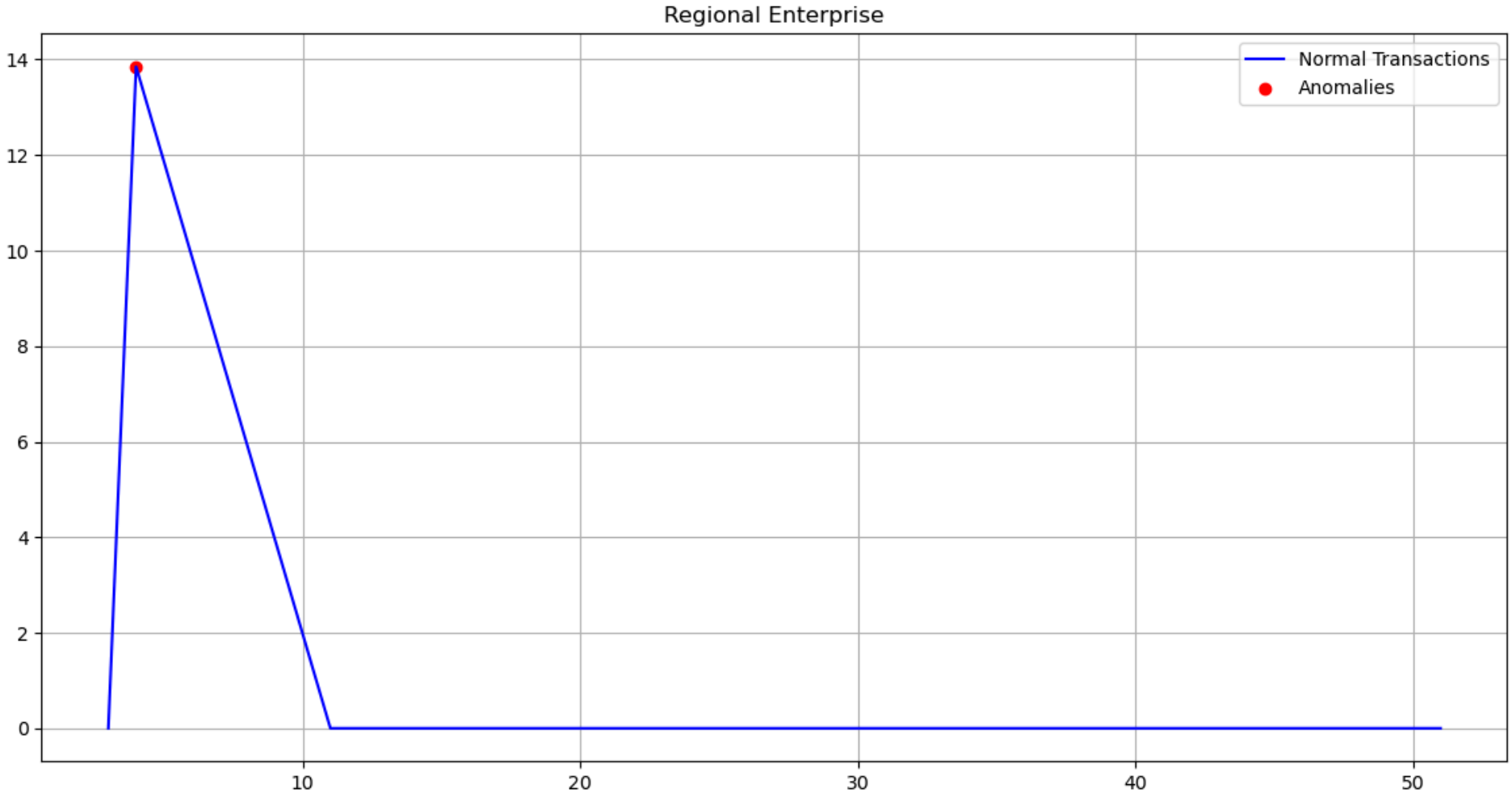

HRA
Normal Transactions
Anomalies
0.04
0.02
0.00
−0.02
−0.04
45
46
47
48
49


Internal Audit & CAFT
Normal Transactions
Anomalies
300
250
200
150
100
50
0
0
10
20
30
40

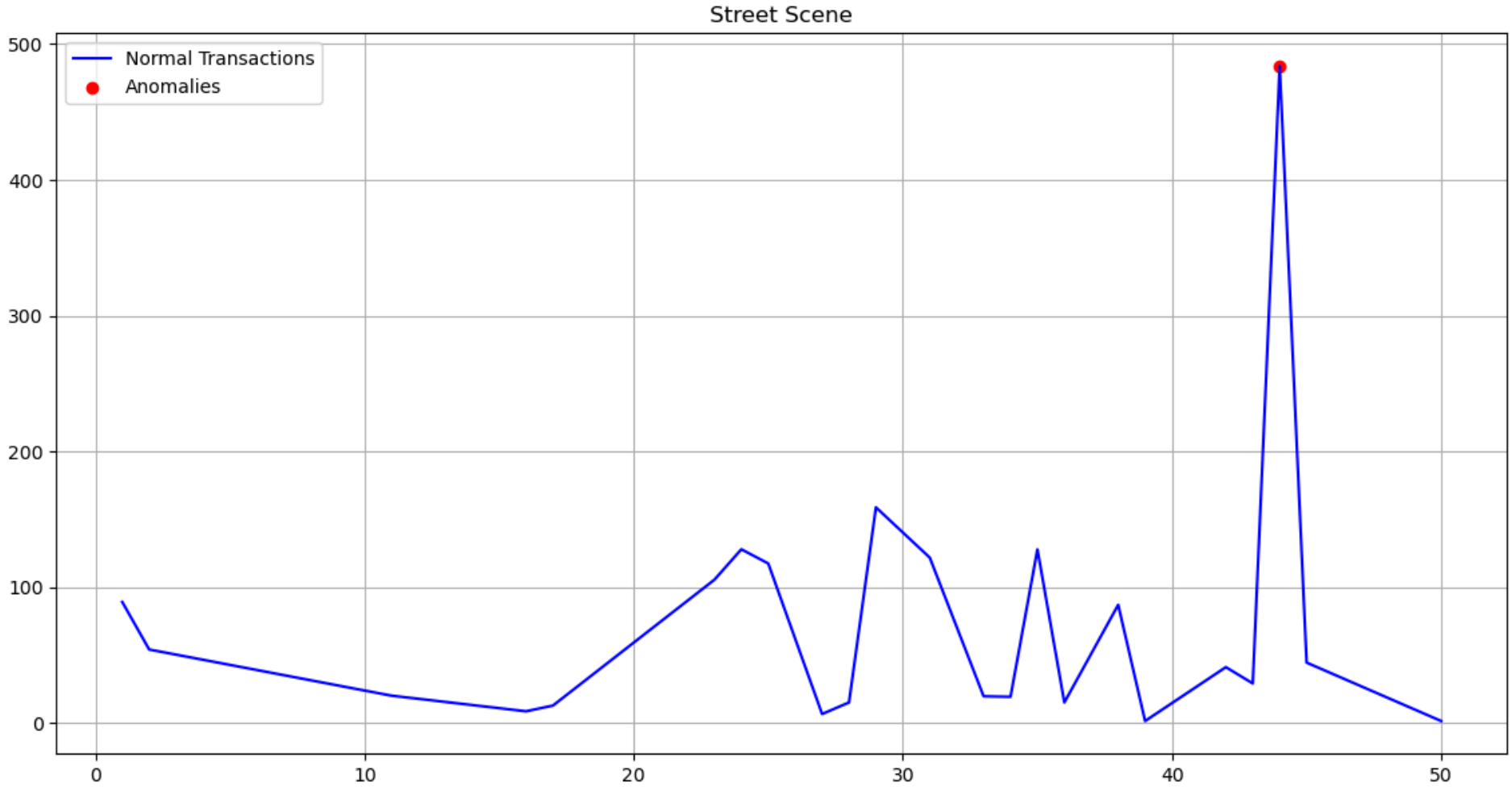

NSCSO
Normal Transactions
Anomalies
120
100
80
60
40
20
0
16
18
20
22
24
26
28
30

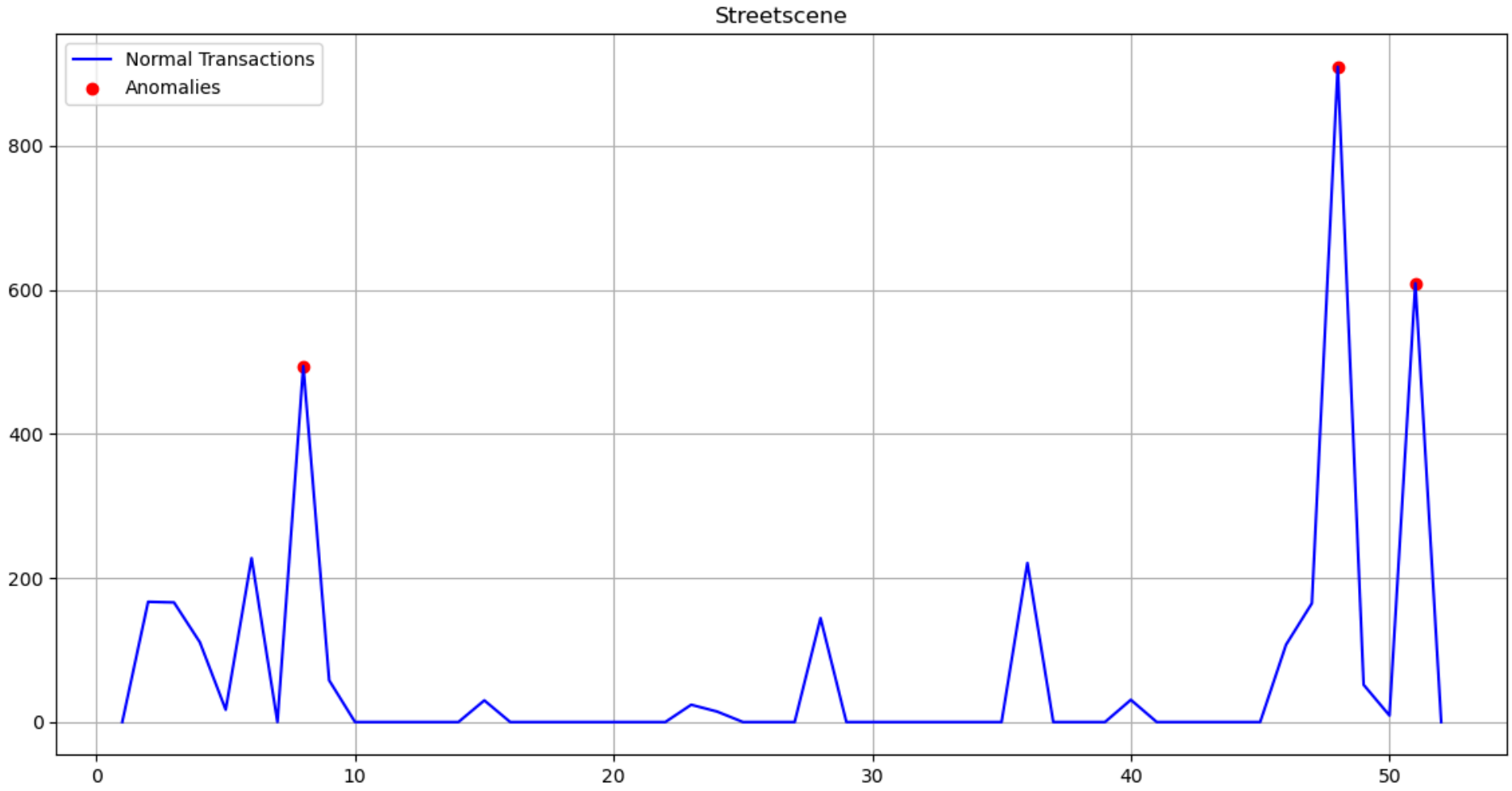

Parking & Infrastructure
Normal Transactions
Anomalies
800
600
400
200
0
0
10
20
30
40
50


Public Health
0
−10
−20
−30
−40
−50
10
12
14
16
18
20
22
24
Normal Transactions
Anomalies


Regional Enterprise
Normal Transactions
Anomalies
14
12
10
8
6
4
2
0
10
20
30
40
50


Strategic Commissioning Board
Normal Transactions
Anomalies
3.75
3.70
3.65
3.60
3.55
3.50
3.45
25.0
25.5
26.0
26.5
27.0


Street Scene
Normal Transactions
Anomalies
500
400
300
200
100
0
0
10
20
30
40
50



Question 5 part 2

service area transaction sum over week (consective)

In [99]:
```
weekly_trans_sum[:10].T
```

Out[99]:

	0	1	2	3	4	5	6	7	8	
Service Area	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adult Commu
Transaction Date	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	
JV Value	0.0	36.85	42.5	0.0	13.85	41.93	0.0	0.0	0.0	

Question 5 part 3

anomaly encoded dataframe

In [95]:
```
anomaly_encoded_df[:10].T
```

Out[95]:

	0	1	2	3	4	5	6	7	8	
Service Area	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adults and Communities	Adult Commu
Transaction Date	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	
JV Value	0.0	36.85	42.5	0.0	13.85	41.93	0.0	0.0	0.0	
Anomaly	False	False	False	False	False	False	False	False	False	

Question 5 part 4

anomaly data present in the df

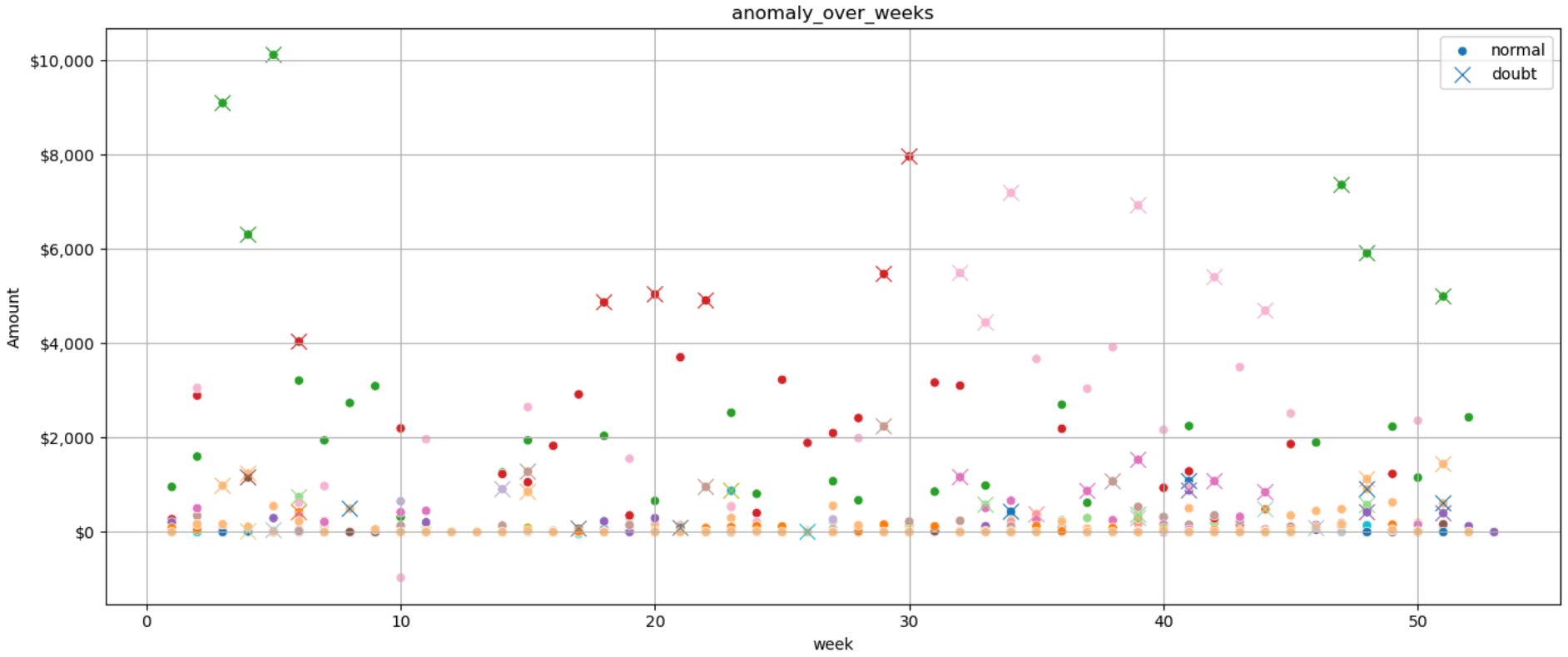
In [96]:
```
only_anomaly[:10].T
```

Out[96]:

	33	40	57	98	107	123	124	133	165	168
Service Area	Adults and Communities	Adults and Communities	Assurance	Assurance	CSG Managed Budget	Children's Education & Skills	Children's Education & Skills	Children's Education & Skills	Children's Education & Skills	Children's Education & Skills
Transaction Date	34.0	41.0	5.0	46.0	6.0	3.0	4.0	15.0	48.0	51.0
JV Value	431.05	1079.5	39.85	83.25	418.46	981.49	1238.16	852.28	1121.65	1441.11

In [83]:
```
def anomaly_sct(data,date_col,grp_col,val_col,tres):
    """
    arg:same to the old function
    function : scatter plot on whole data anomaly or not
    reutrn: None
    """
    anom_temp=data[val_col]>tres.fillna(0)
#     anom_temp=tres
    plt.figure(figsize=(14,6))

    g=sns.scatterplot(data=data,x=date_col,y=val_col,hue=grp_col, legend=False, label='normal',palette='tab20')
    sns.scatterplot(data=data[anom_temp],x=date_col,y=val_col,hue=grp_col,palette='tab20',s=100,marker='x',legend=False,label='dou
    g.yaxis.set_major_formatter('${:,.0f}'.format)

    plt.title('anomaly_over_weeks')
    plt.xlabel('week')
    plt.ylabel('Amount')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

In [84]:
```
anomaly_sct(anomaly_encoded_df,'Transaction Date','Service Area','JV Value',treshold)
```

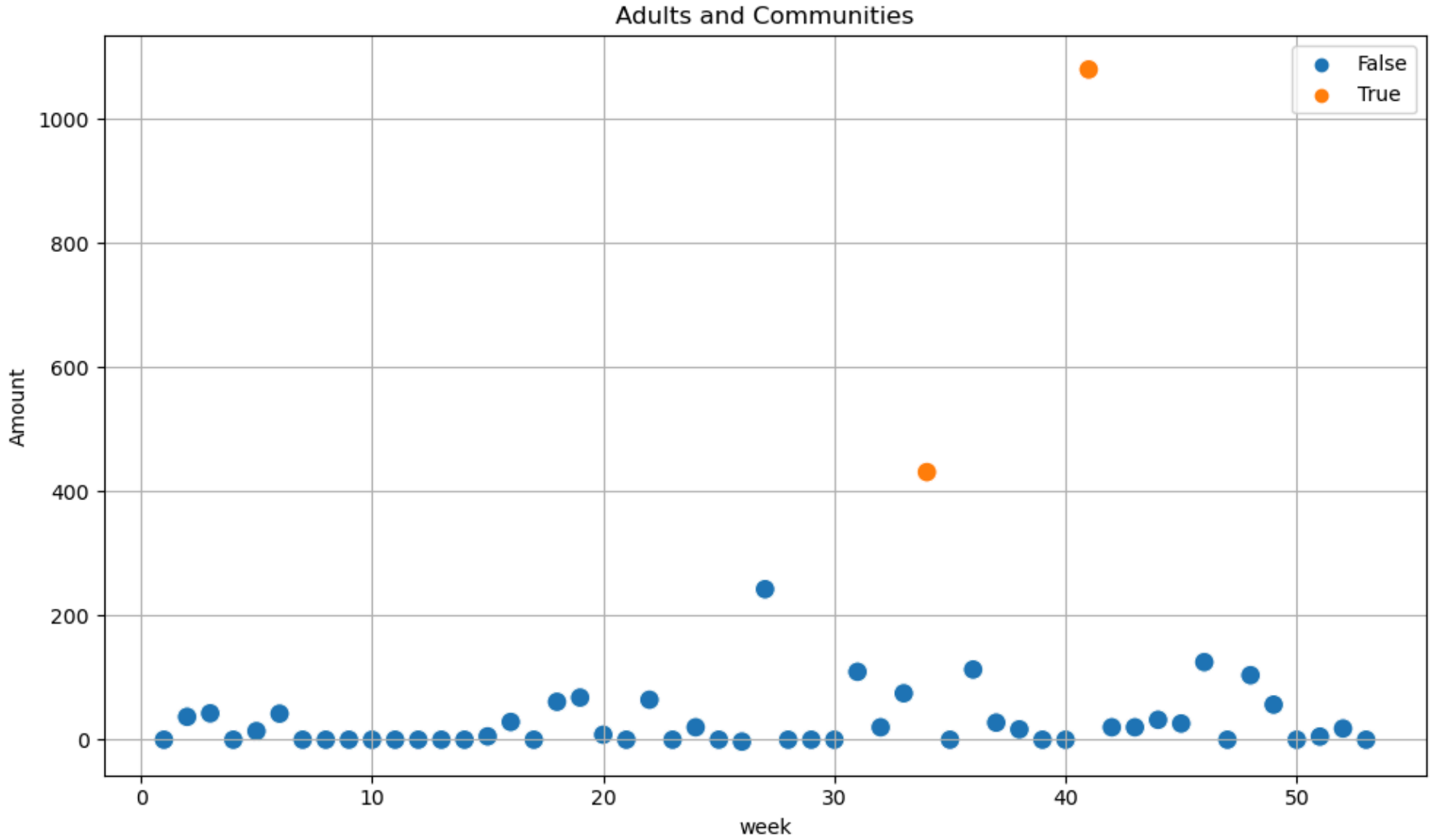


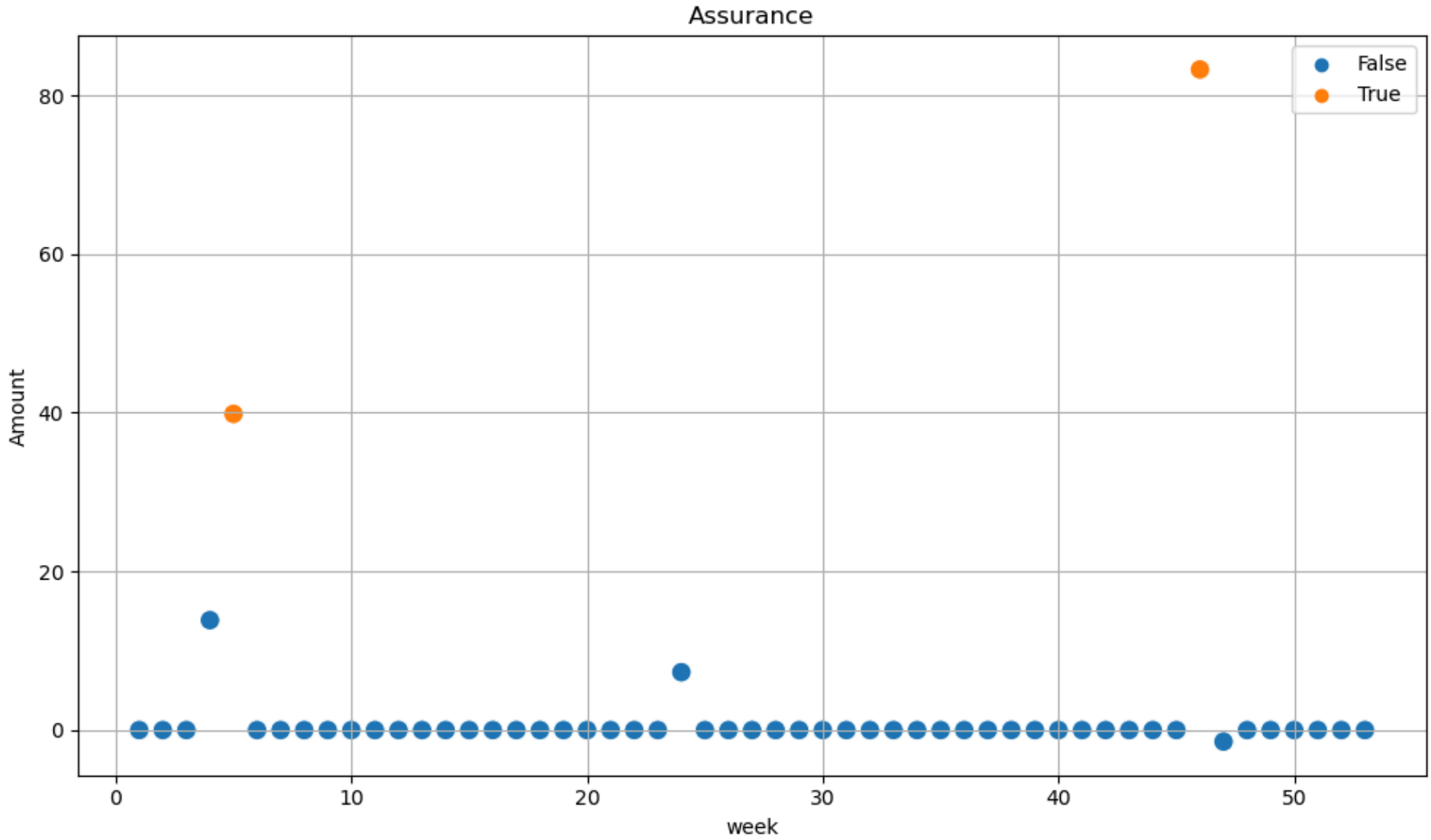
appendix

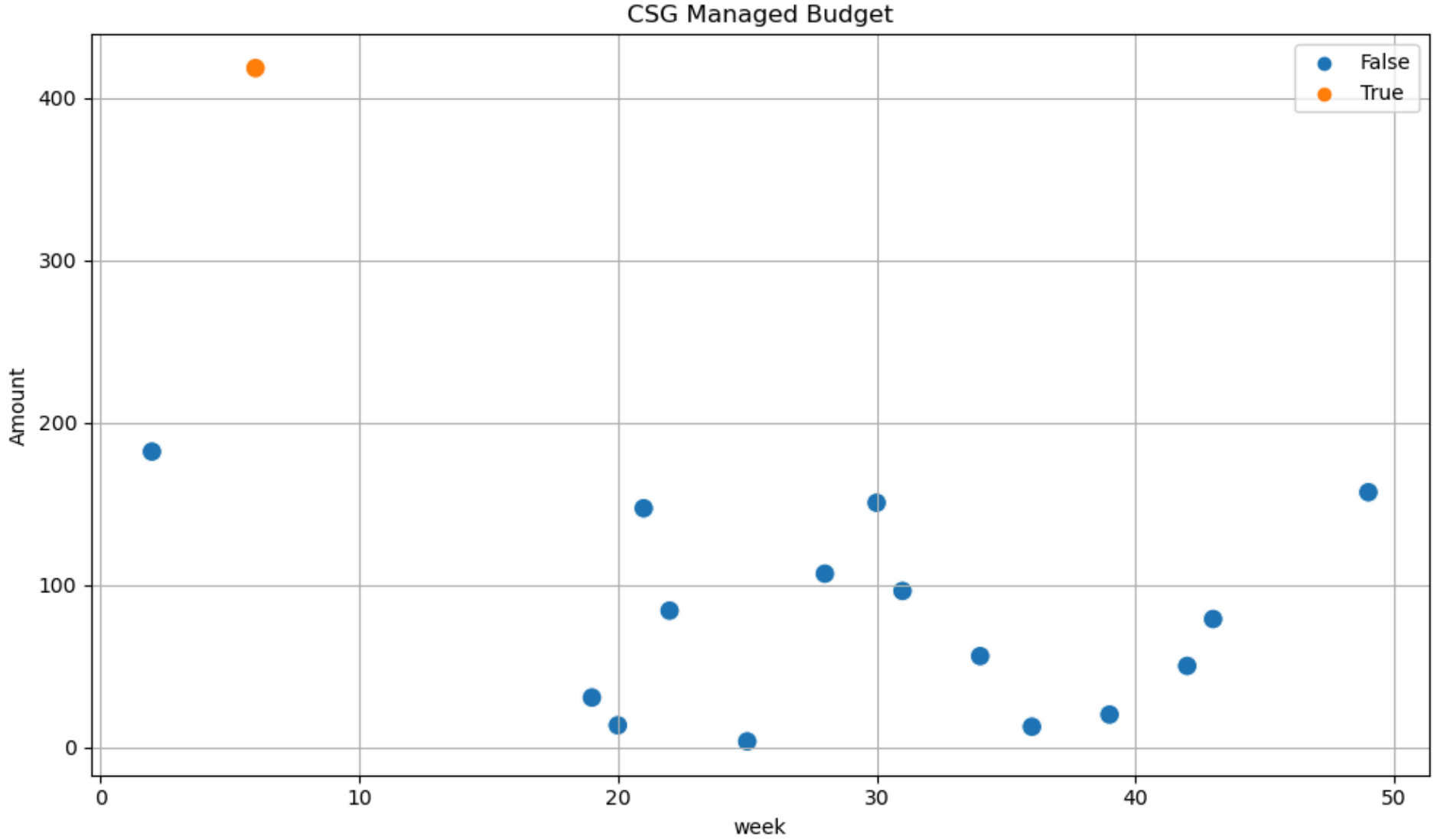
Question 5 part 5

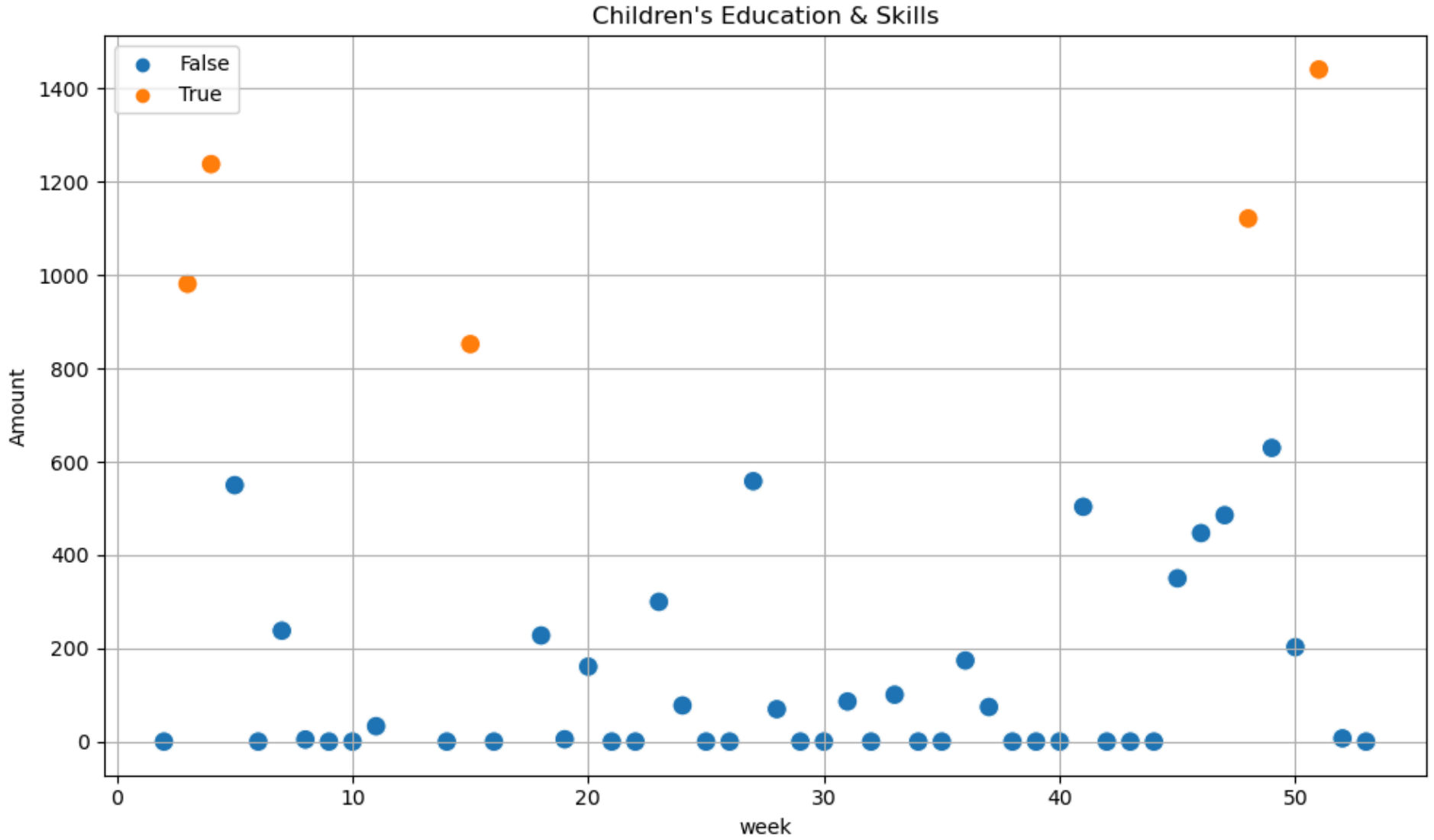
different combinations

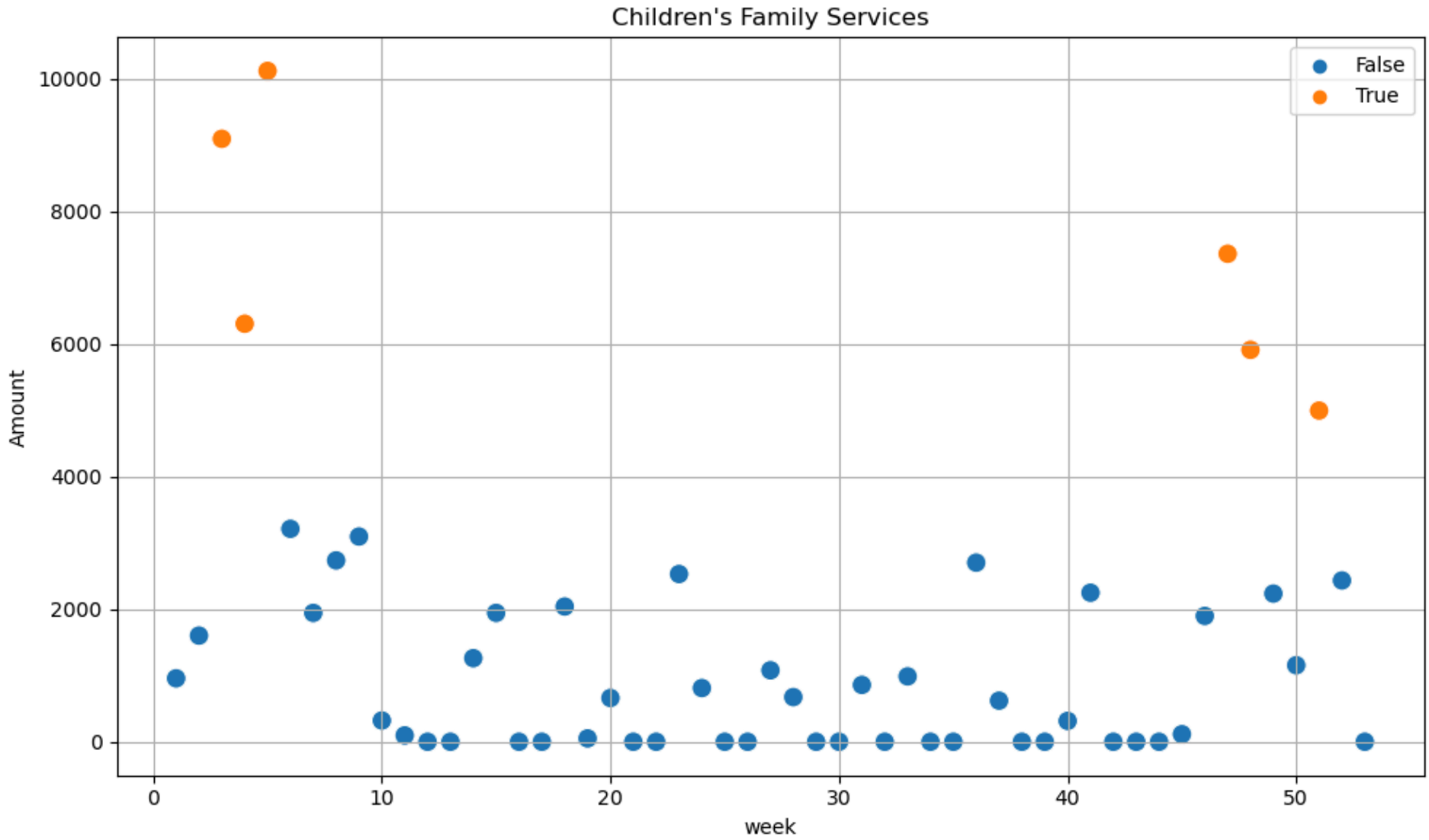
In [85]:
```
# Plot anomalies for each service area separately
for service_area, data in anomaly_encoded_df.groupby('Service Area'):
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=data, x='Transaction Date', y='JV Value', hue='Anomaly', palette='tab10', markers=['o', 'x'], s=100)
    plt.title(service_area)
    plt.xlabel('week')
    plt.ylabel('Amount')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

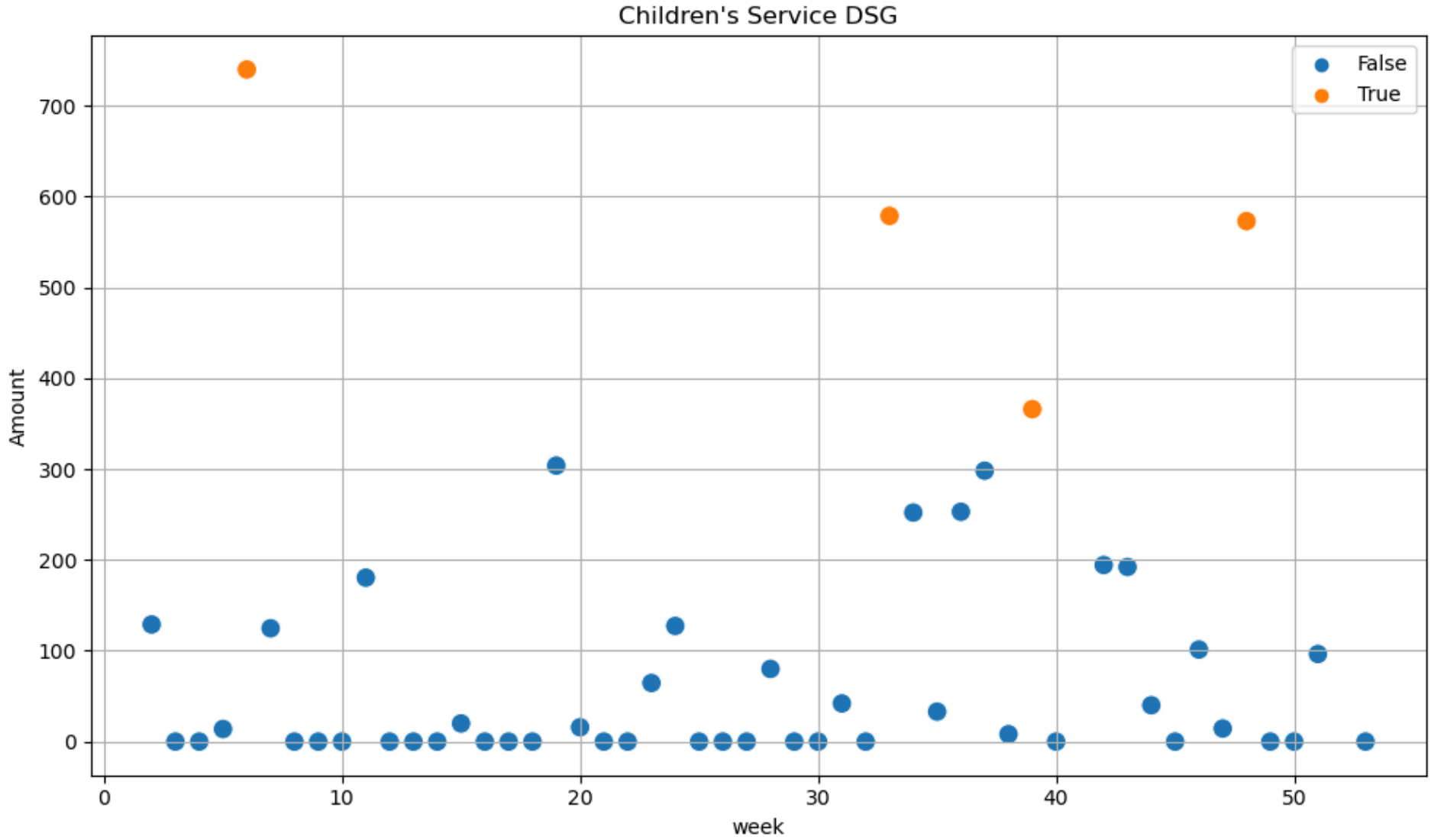

Adults and Communities
False
True
1000
800
600
400
200
0
Amount
0
10
20
30
40
50
week

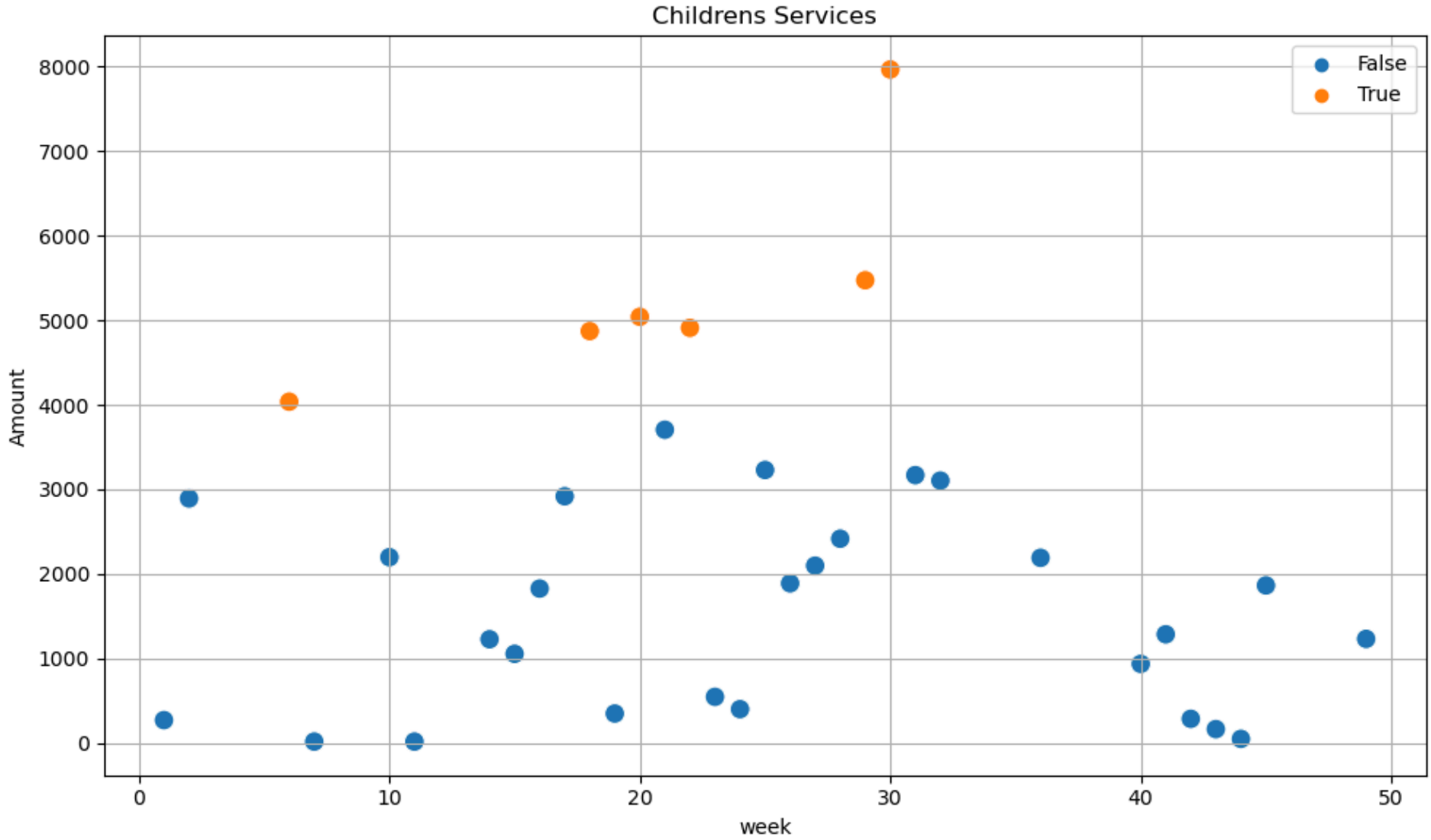

Assurance
False
True
Amount
week
0
20
40
60
80
0
10
20
30
40
50

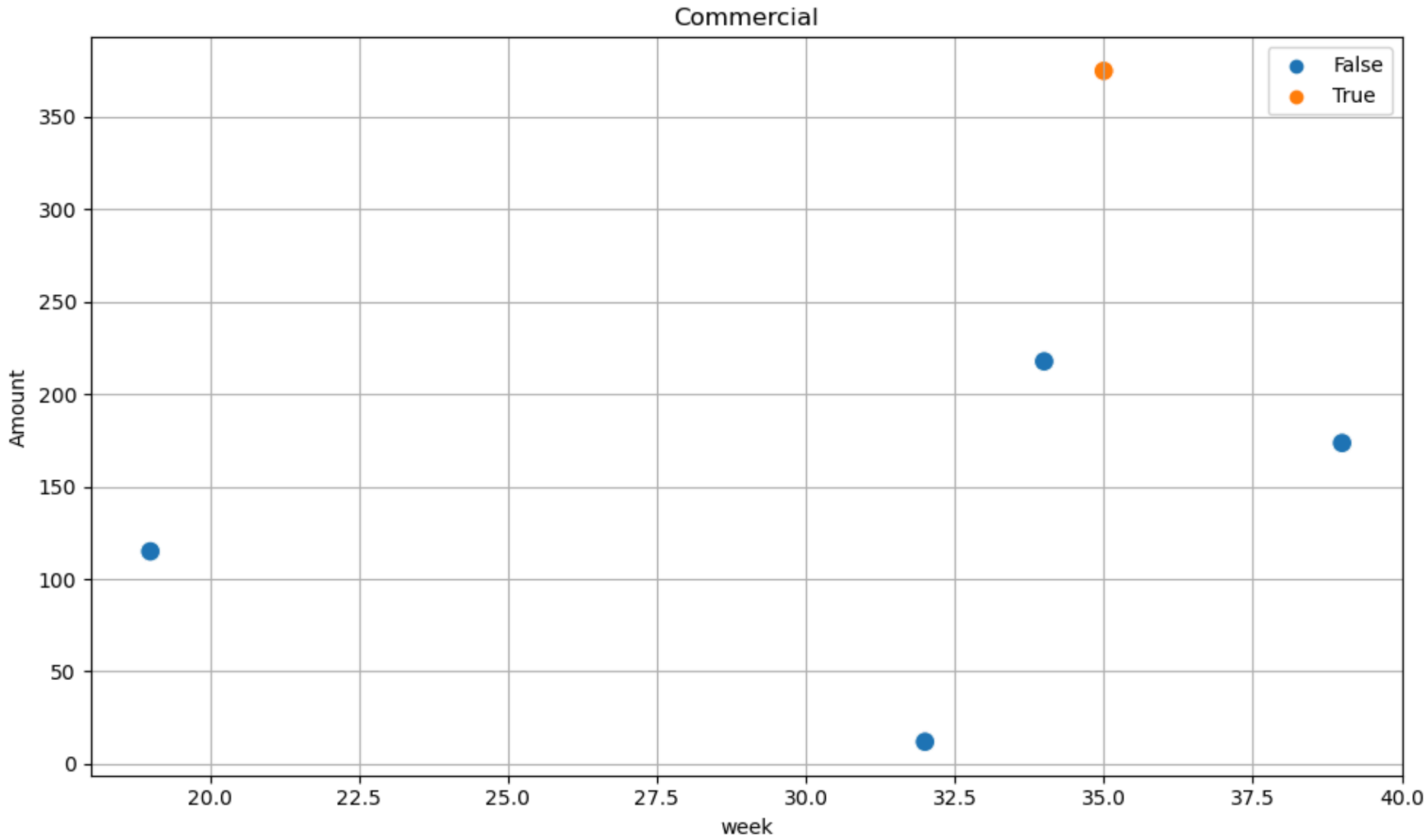

CSG Managed Budget
False
True
Amount
week
0
100
200
300
400
0
10
20
30
40
50

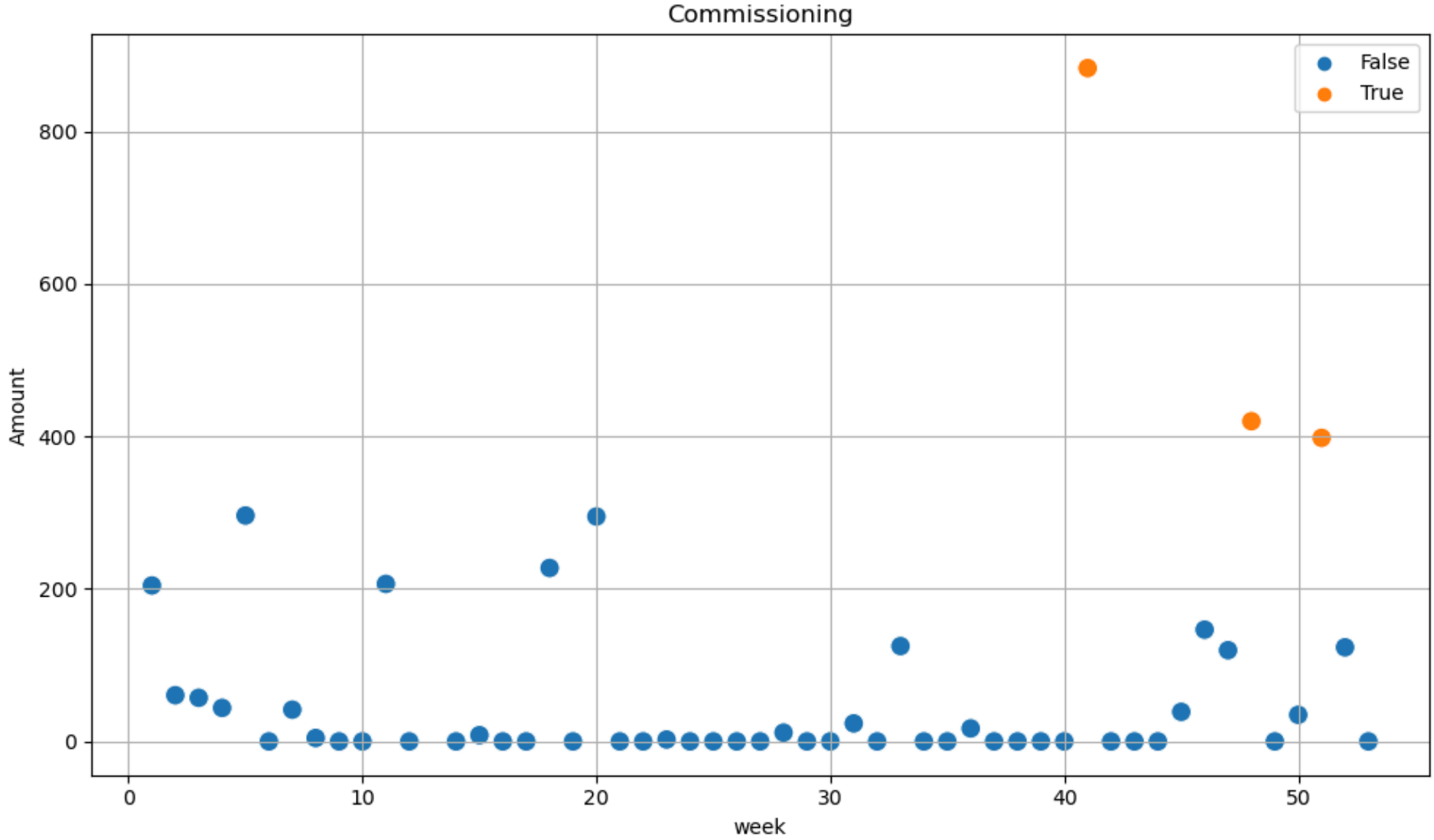

Children's Education & Skills
False
True
Amount
week
0
200
400
600
800
1000
1200
1400
0
10
20
30
40
50

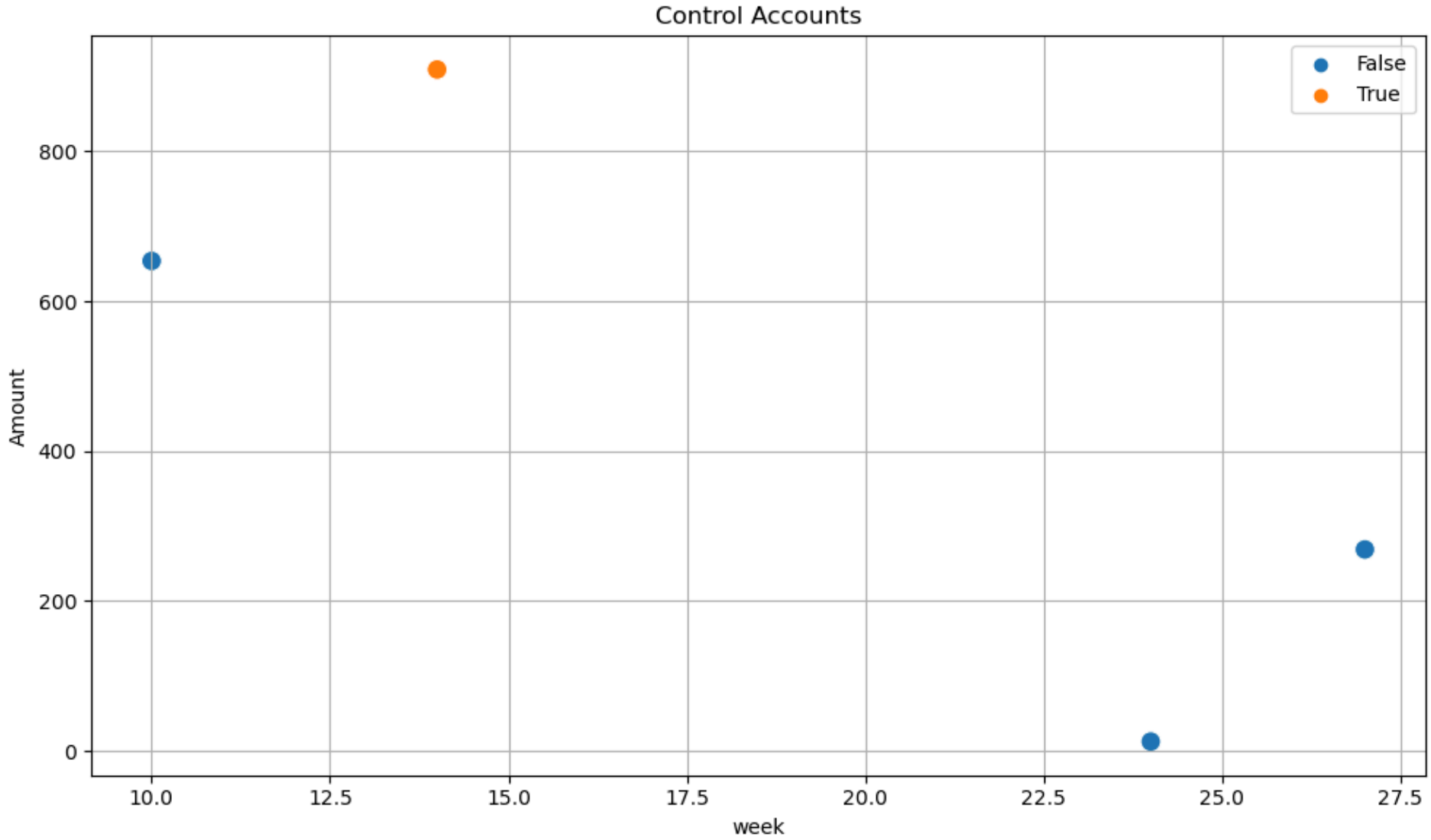

Children's Family Services
False
True
Amount
10000
8000
6000
4000
2000
0
0
10
20
30
40
50
week

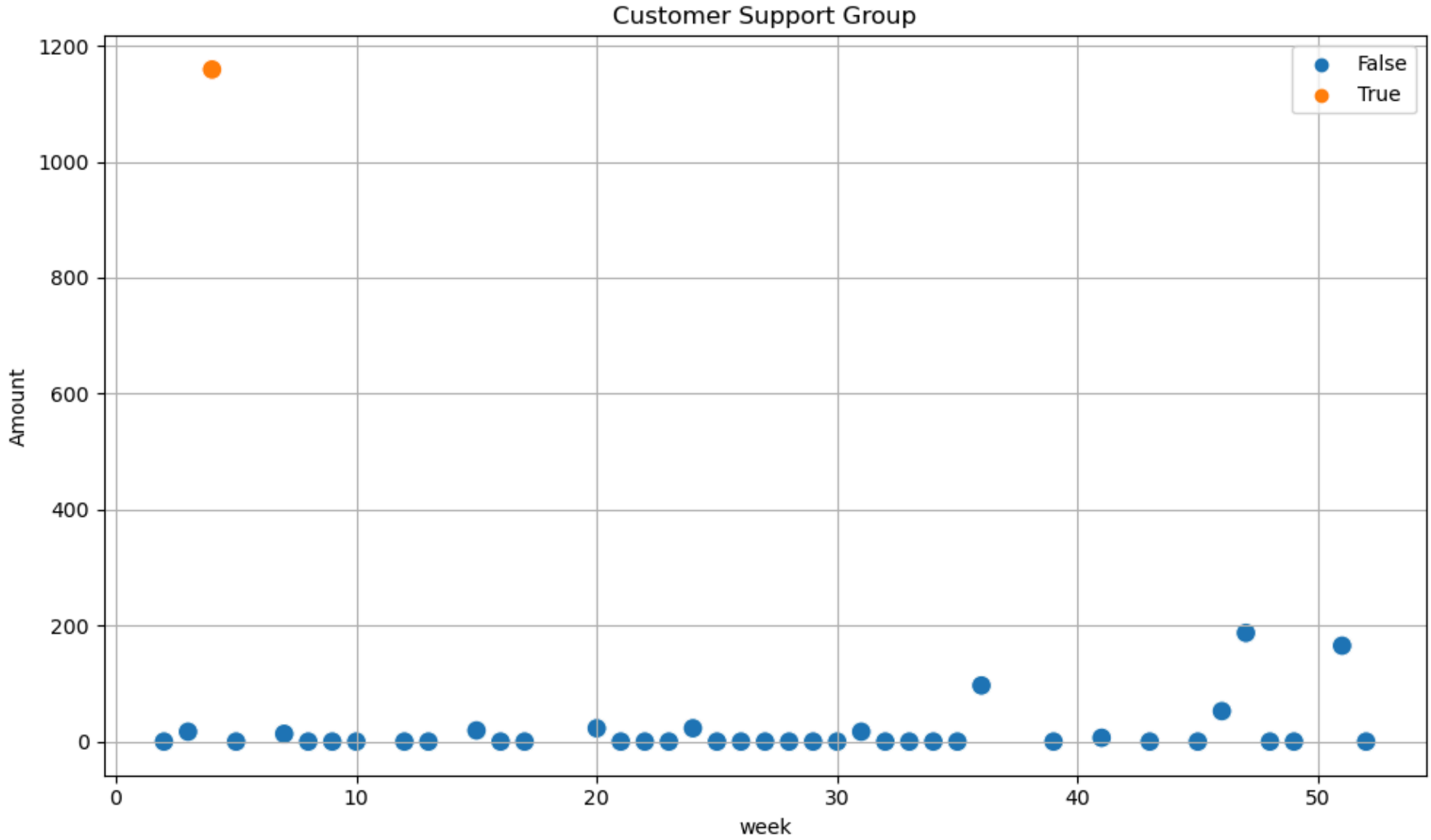

Children's Service DSG
False
True
Amount
week
0
100
200
300
400
500
600
700
0
10
20
30
40
50

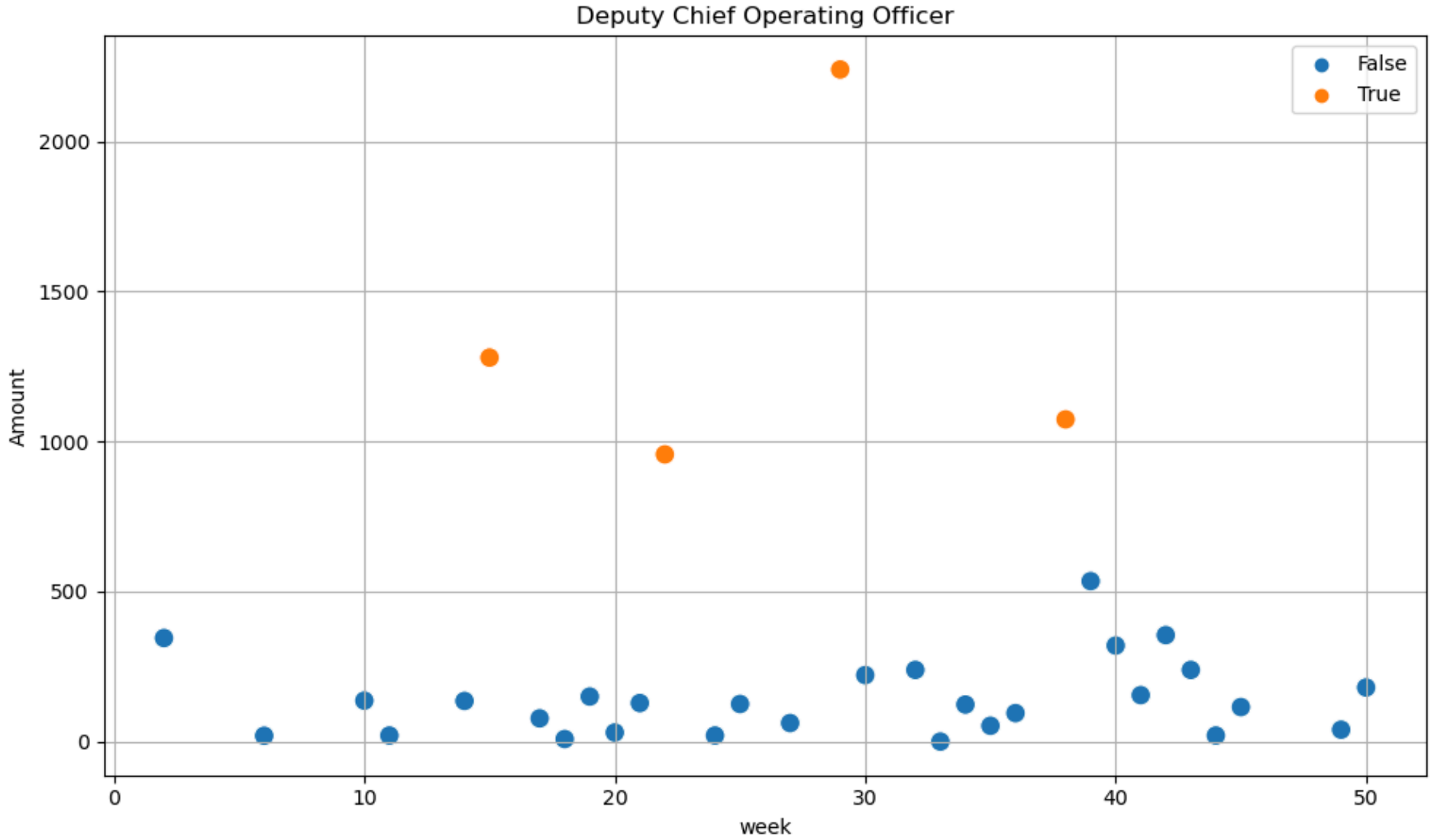

Childrens Services
False
True
Amount
week
0
1000
2000
3000
4000
5000
6000
7000
8000
0
10
20
30
40
50

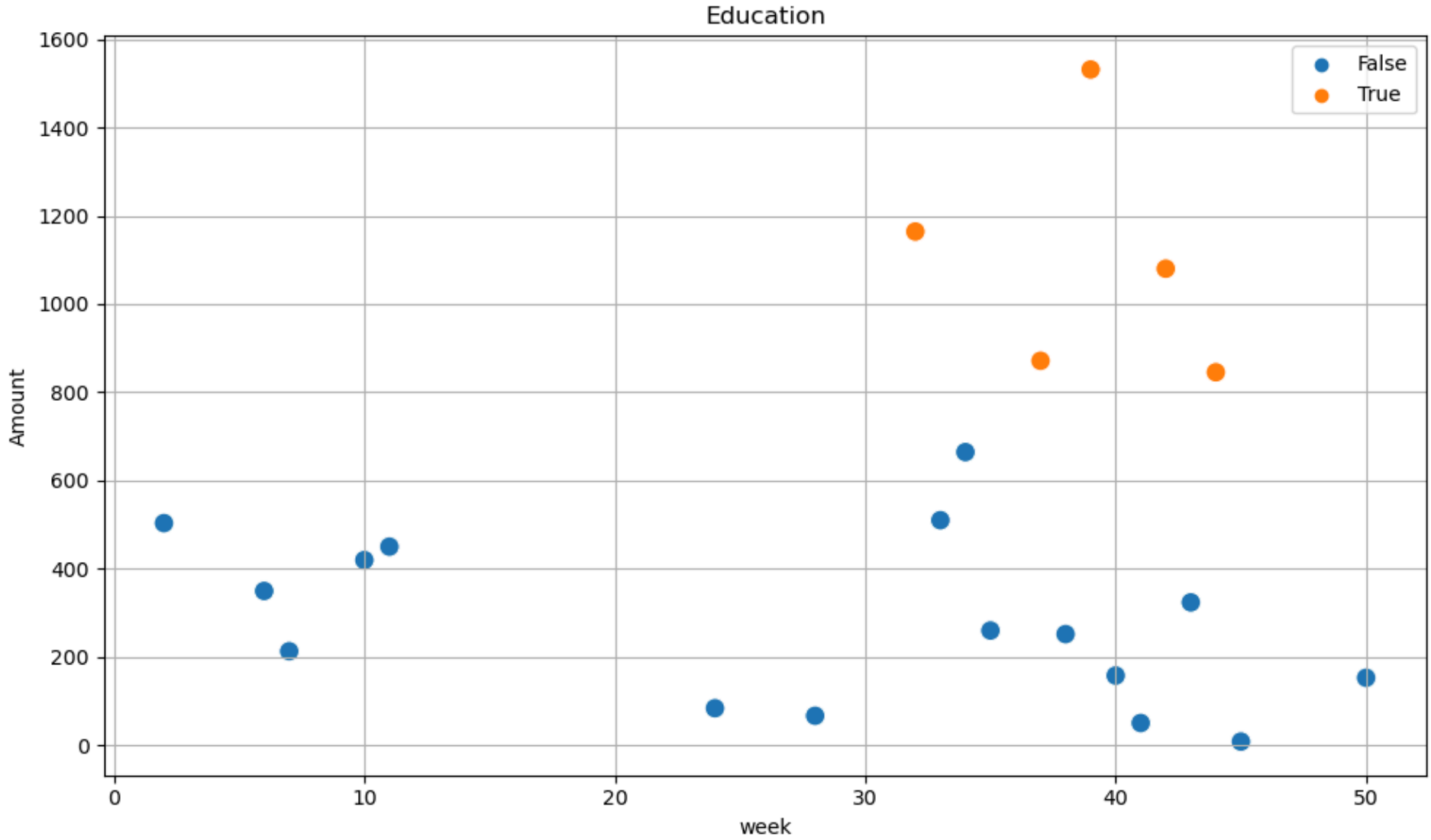

Commercial
False
True
Amount
week
0
50
100
150
200
250
300
350
20.0
22.5
25.0
27.5
30.0
32.5
35.0
37.5
40.0

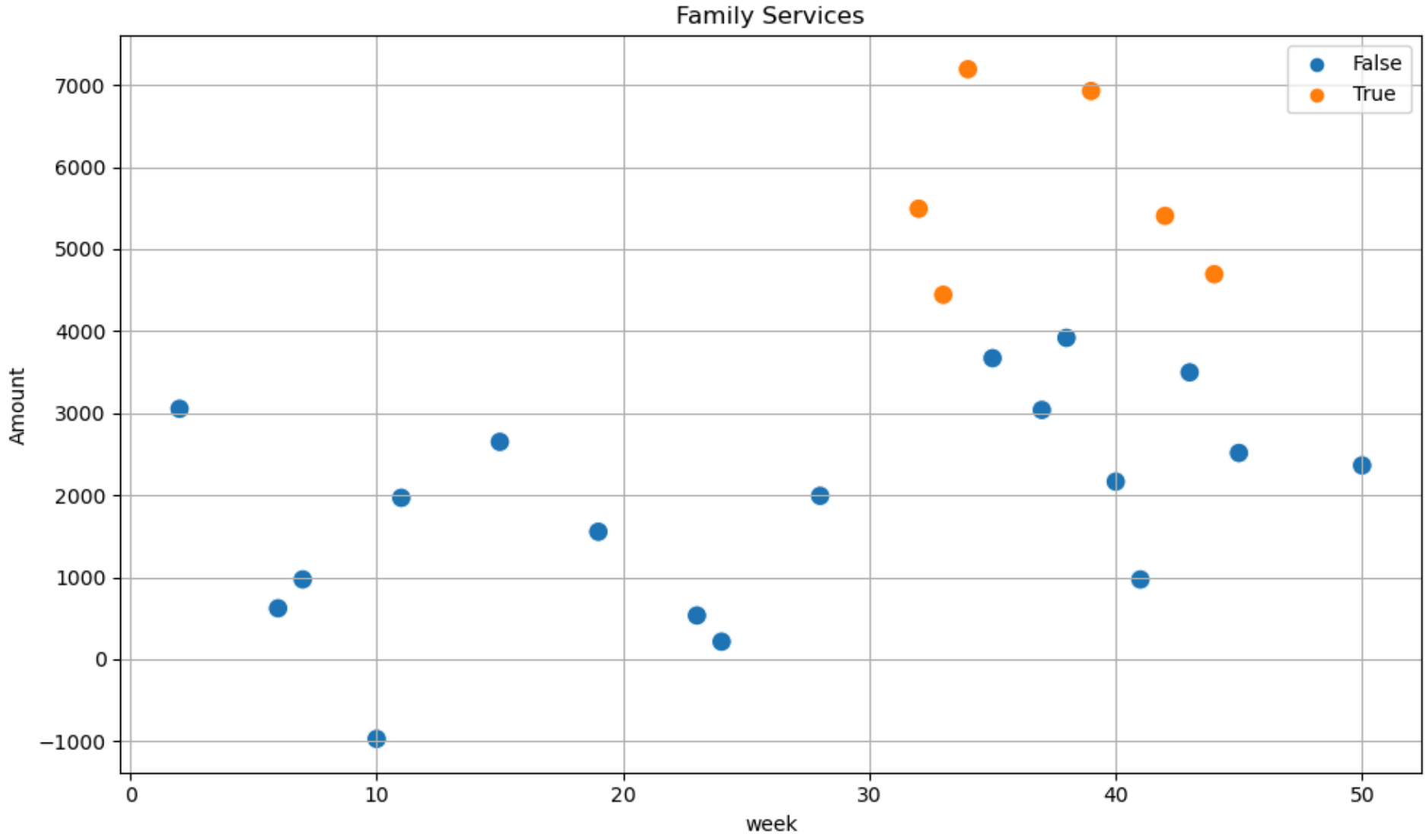

Commissioning
False
True
800
600
400
200
0
Amount
0
10
20
30
40
50
week

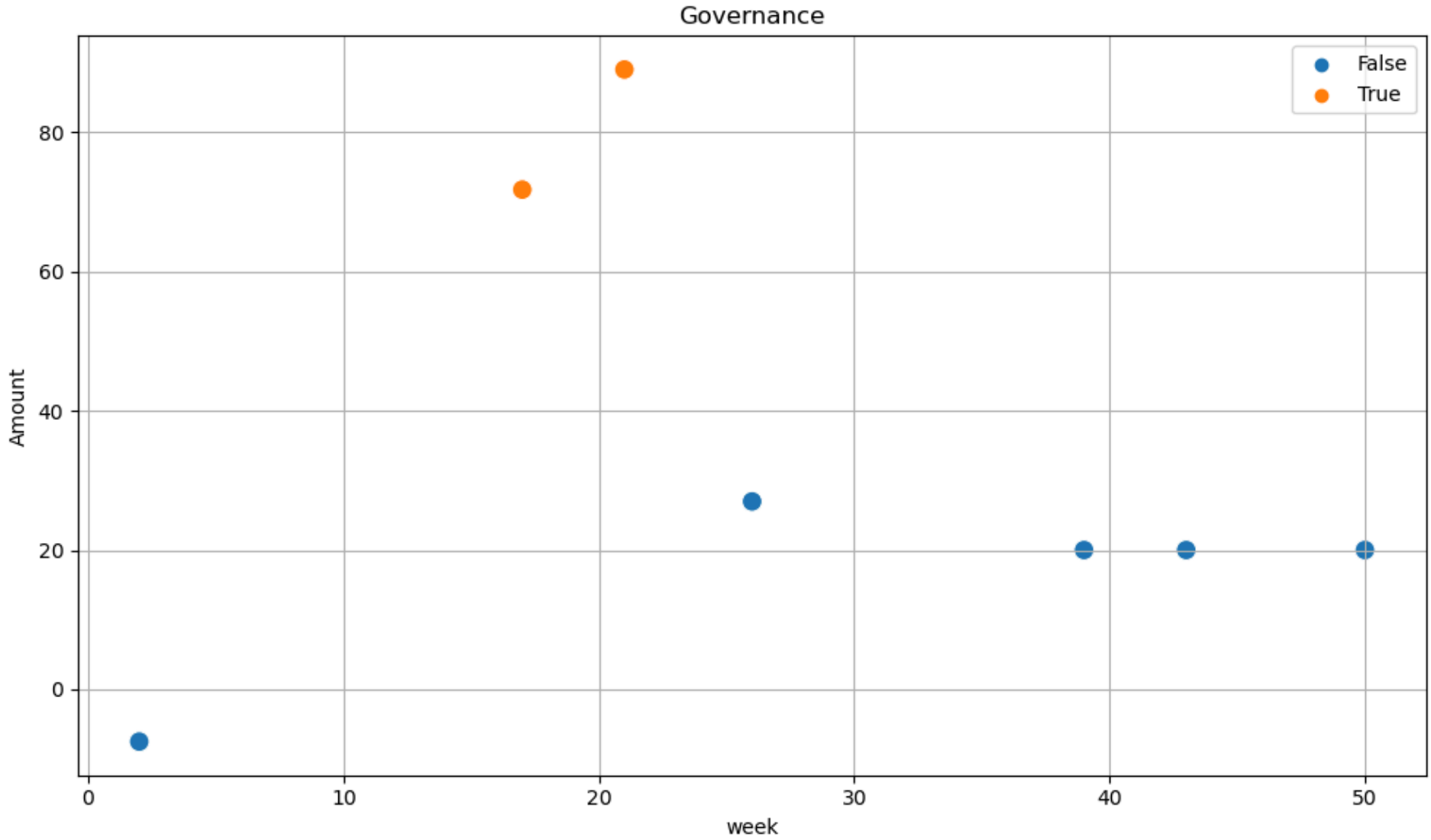

Control Accounts
False
True
800
600
400
200
0
Amount
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
week

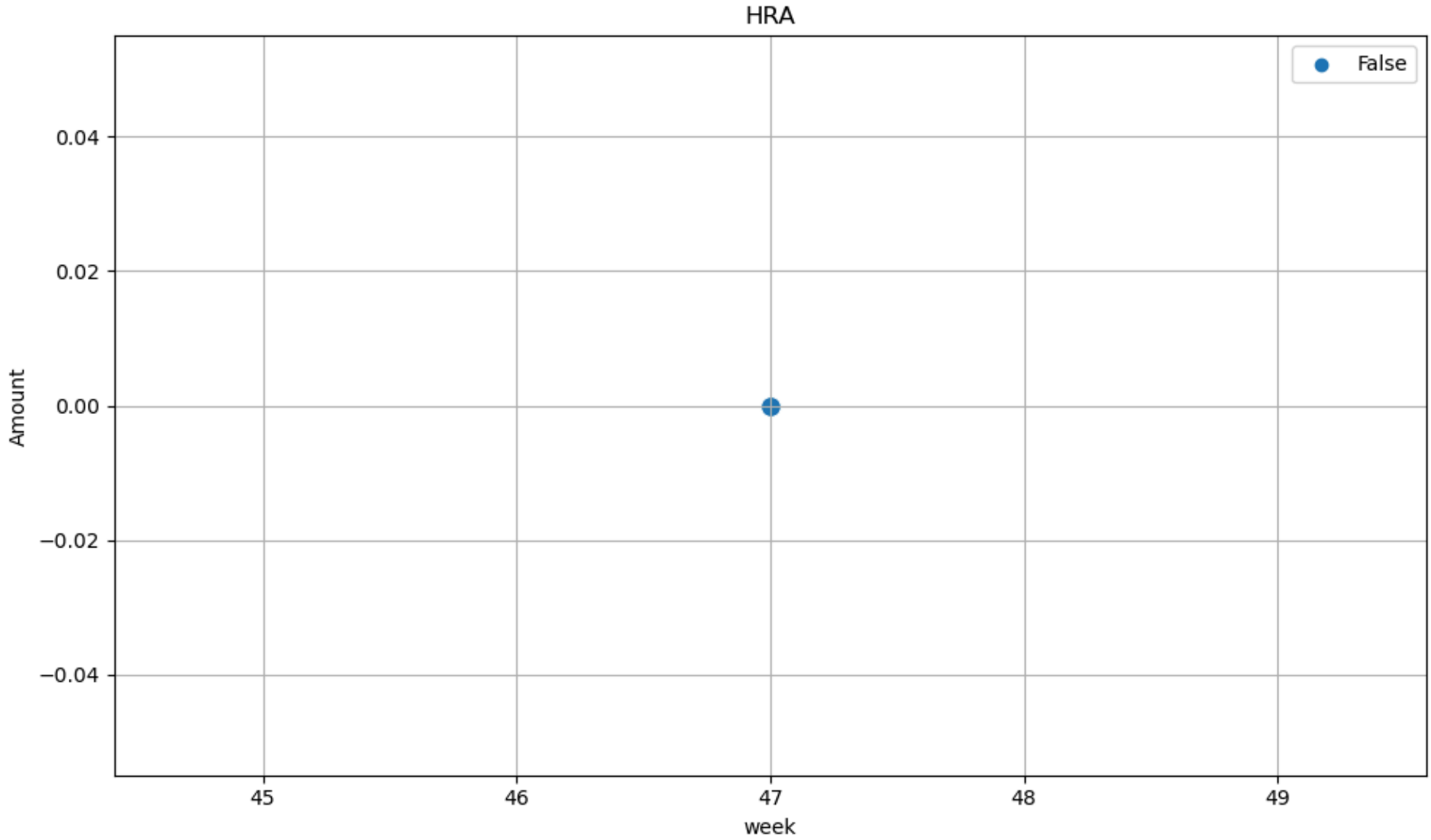

Customer Support Group
False
True
Amount
week
0
200
400
600
800
1000
1200
0
10
20
30
40
50

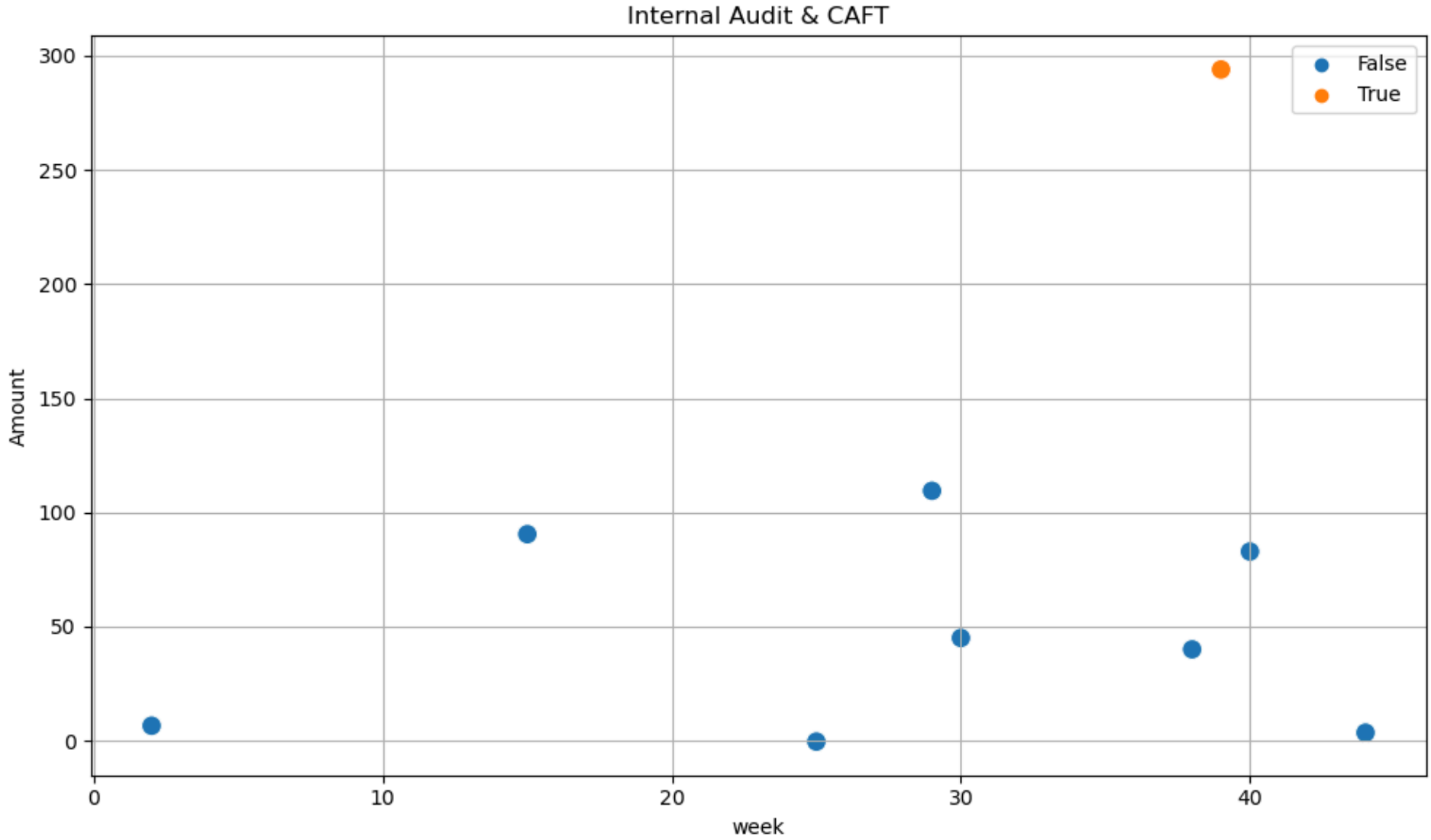

Deputy Chief Operating Officer
False
True
2000
1500
1000
500
0
Amount
0
10
20
30
40
50
week

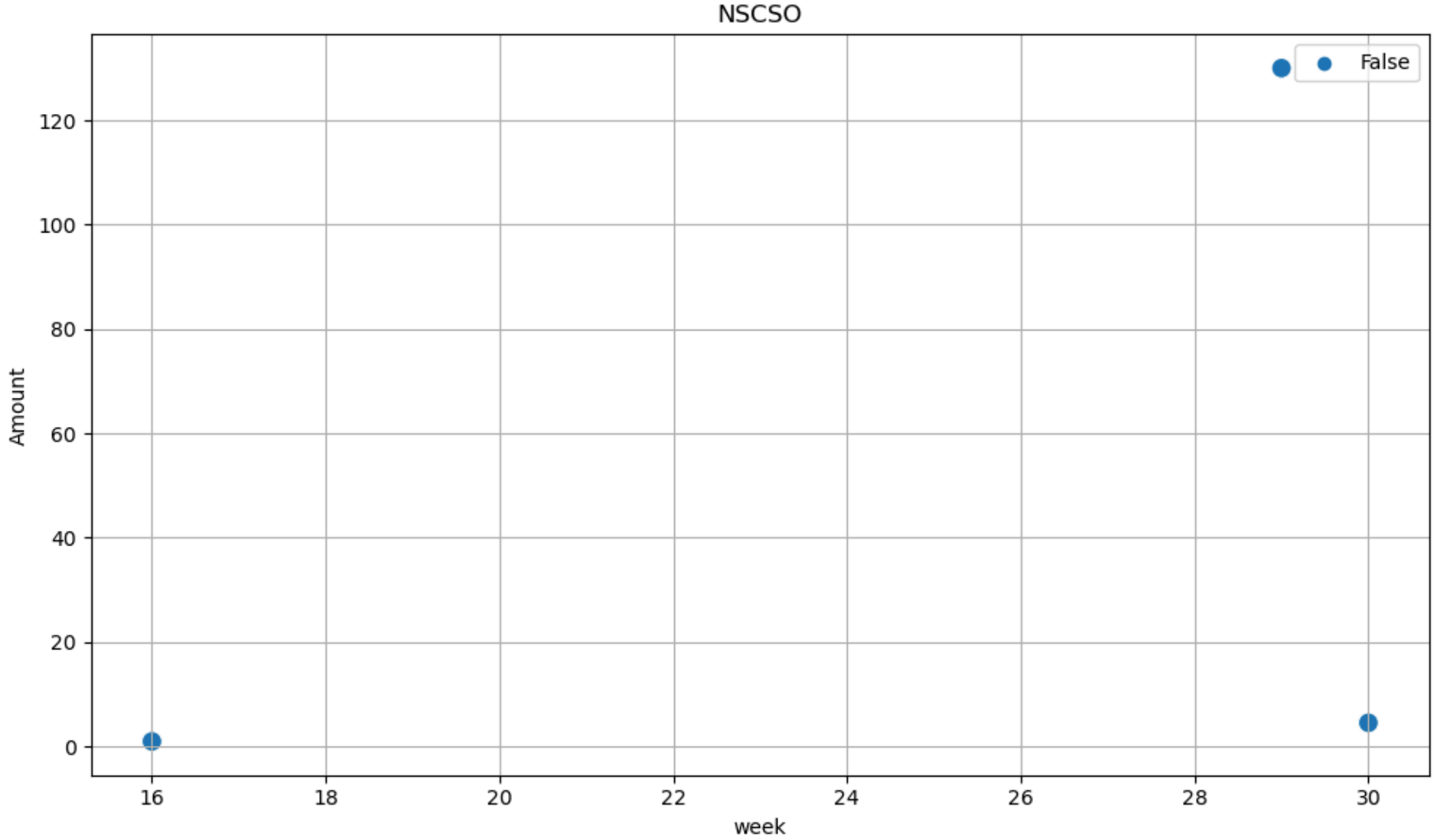

Education
False
True
Amount
week
0
200
400
600
800
1000
1200
1400
1600
0
10
20
30
40
50

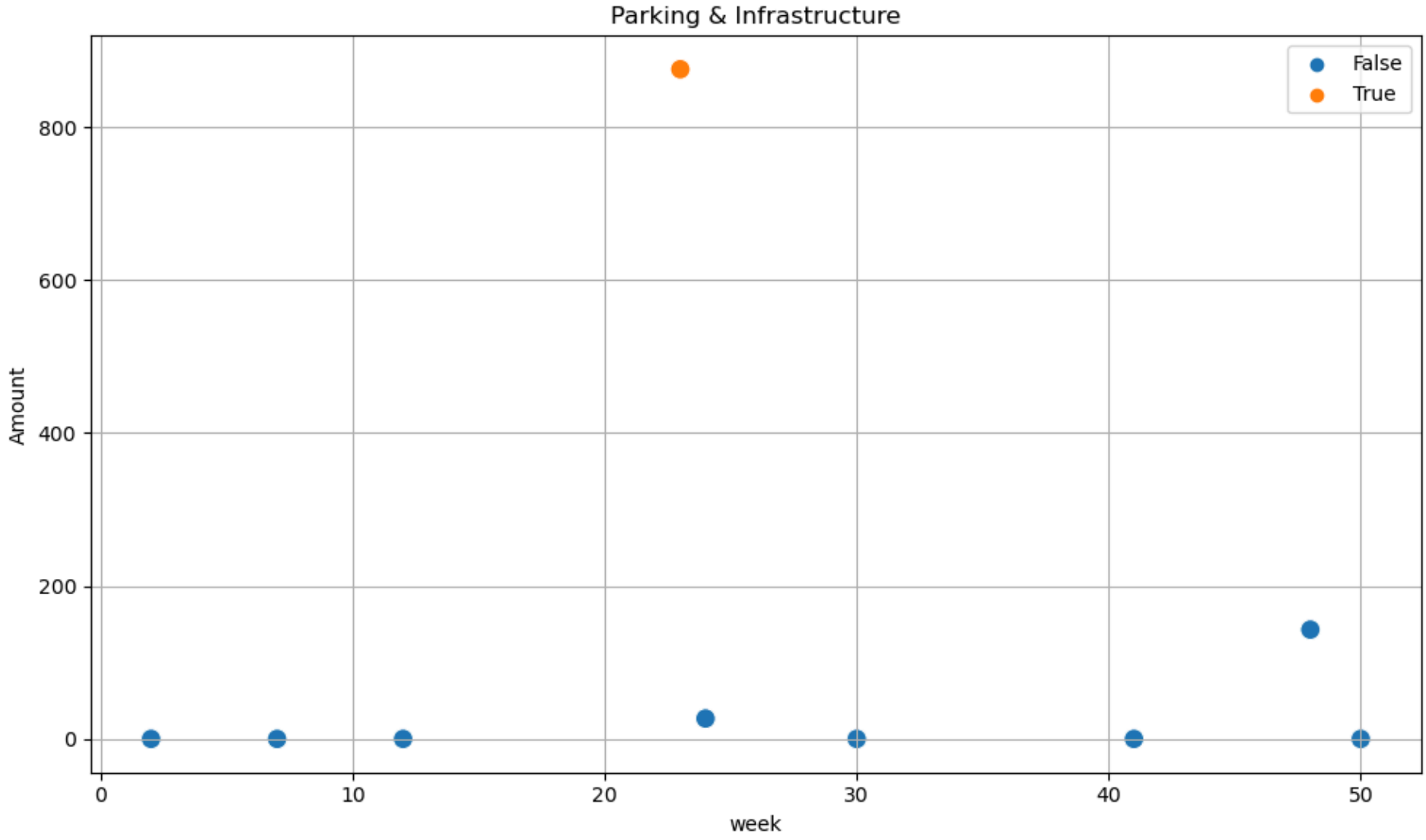

Family Services
False
True
7000
6000
5000
4000
3000
2000
1000
0
−1000
Amount
0
10
20
30
40
50
week

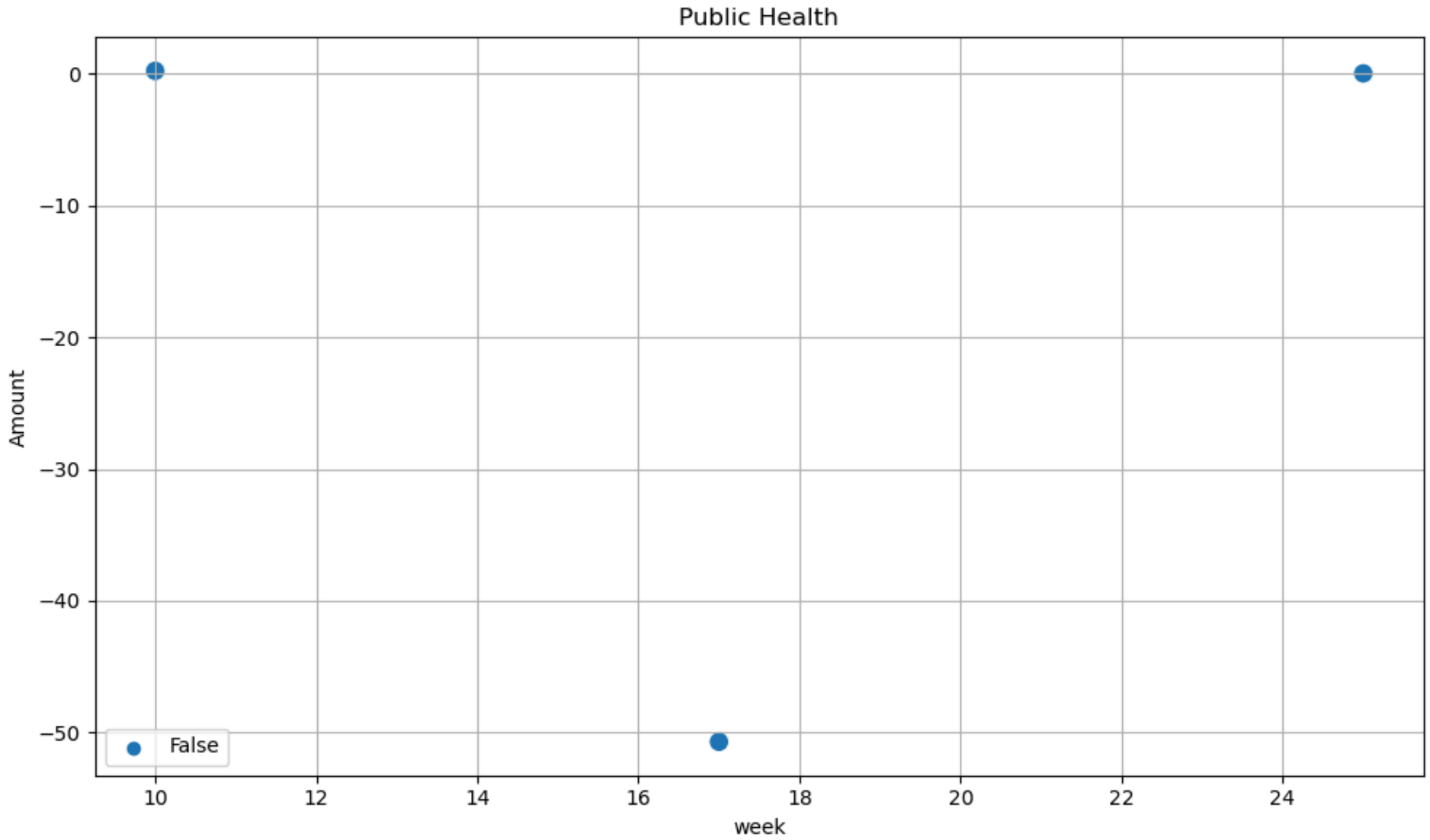

Governance
False
True
Amount
week
0
20
40
60
80
0
10
20
30
40
50

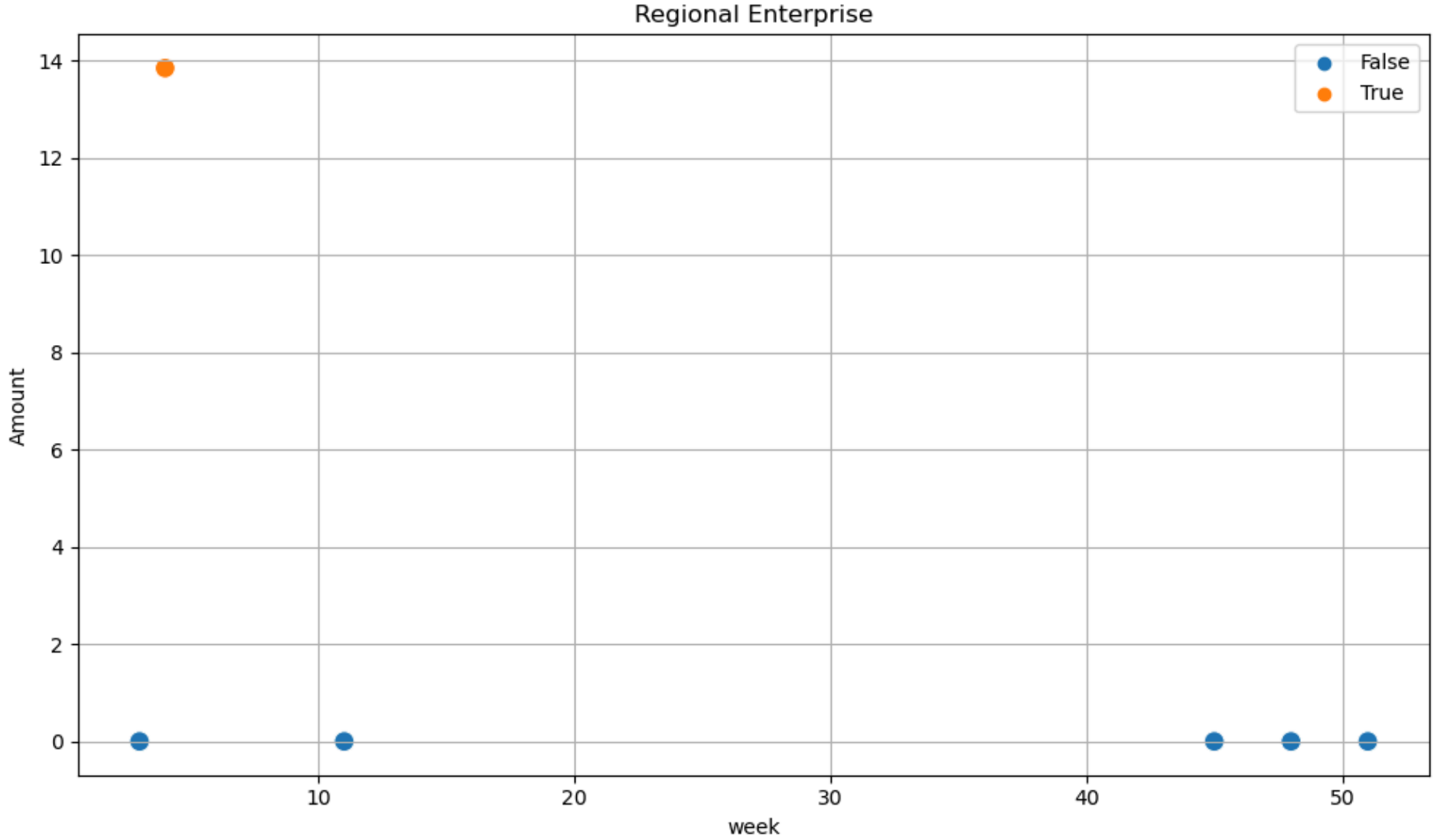

HRA
False
0.04
0.02
0.00
−0.02
−0.04
Amount
45
46
47
48
49
week

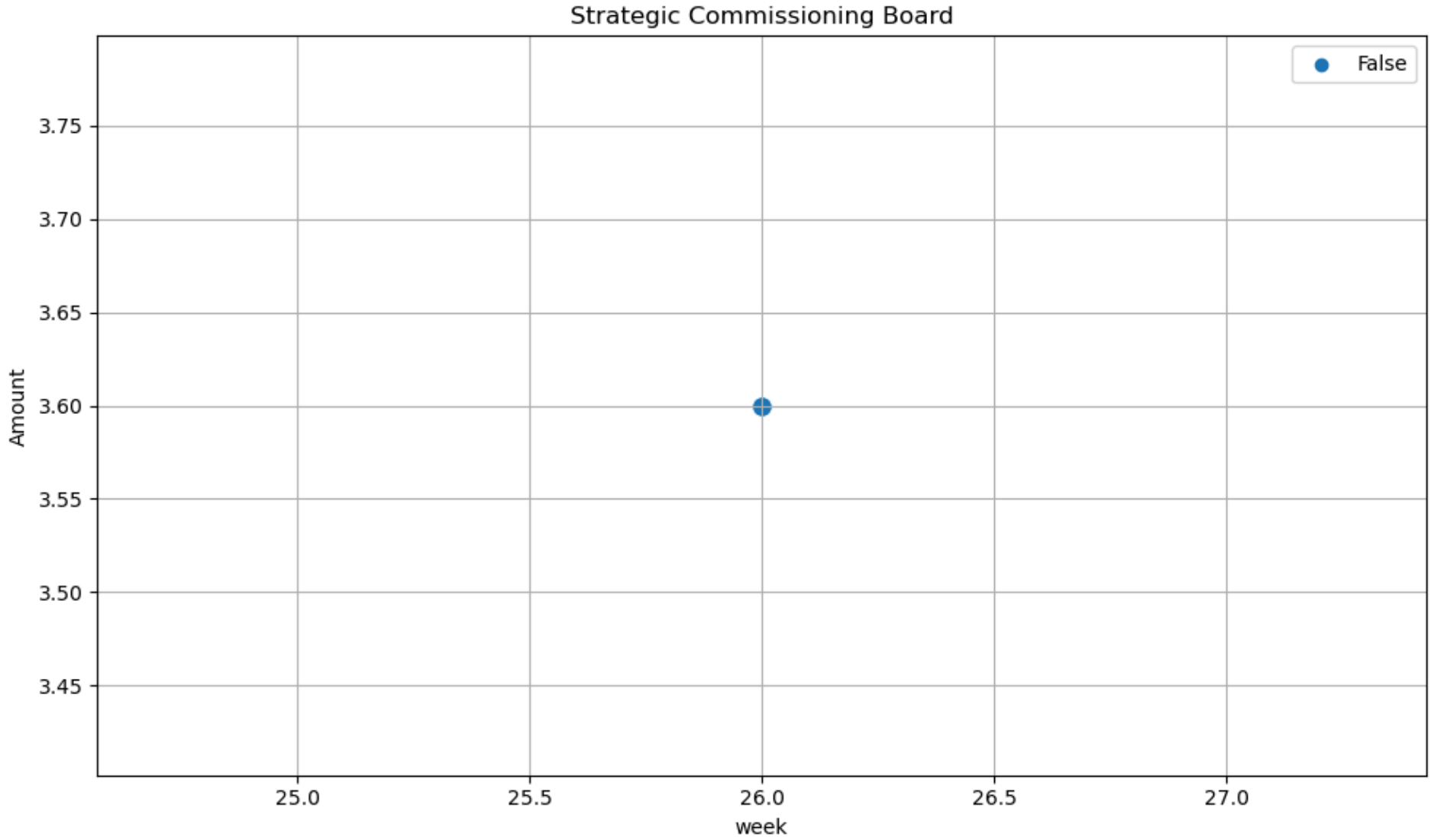

Internal Audit & CAFT
False
True
Amount
week
300
250
200
150
100
50
0
0
10
20
30
40

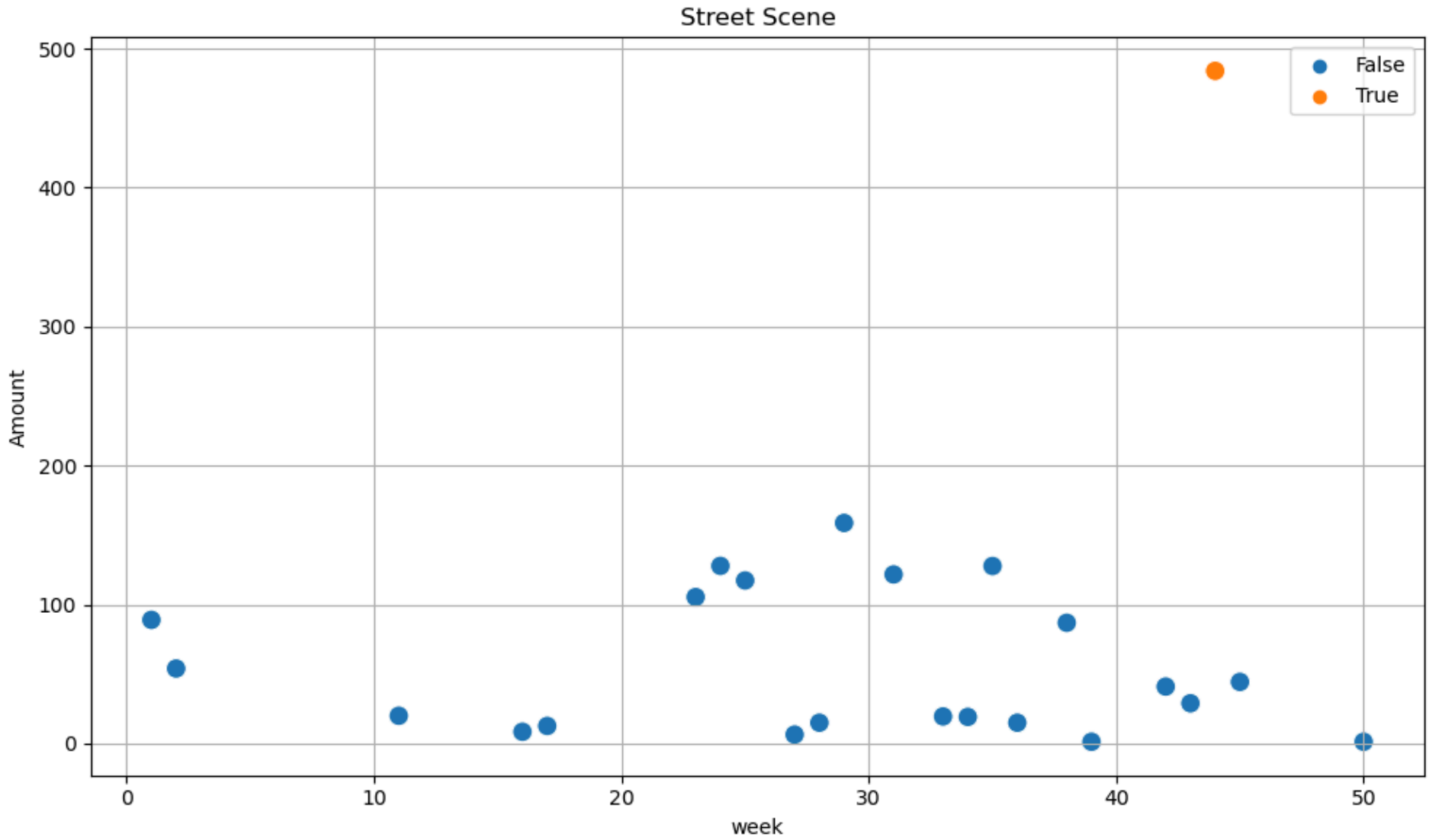

NSCSO
False
120
100
80
60
40
20
0
Amount
16
18
20
22
24
26
28
30
week

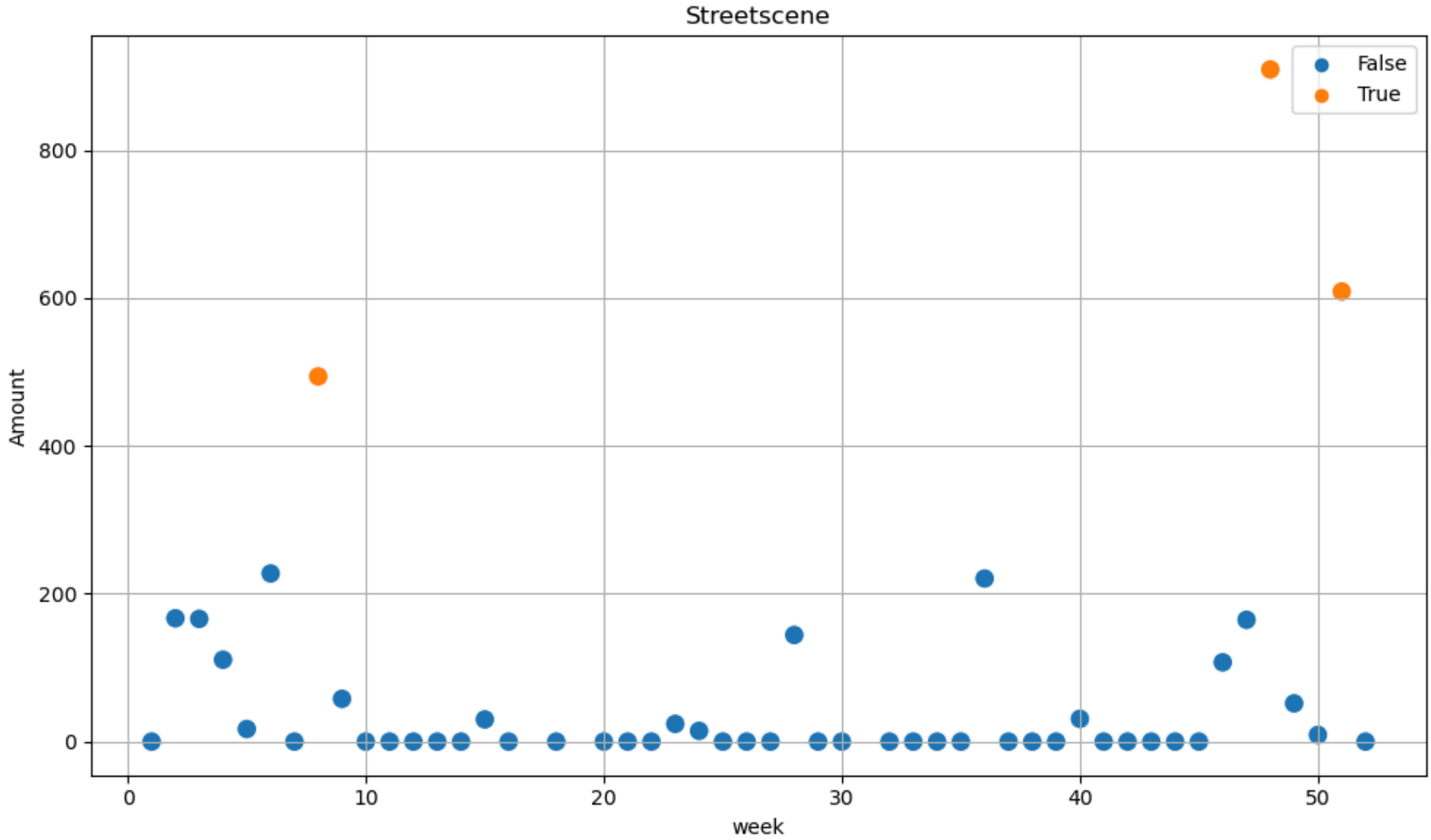

Parking & Infrastructure
False
True
800
600
400
200
0
Amount
0
10
20
30
40
50
week


Public Health
0
−10
−20
−30
−40
−50
Amount
False
10
12
14
16
18
20
22
24
week


Regional Enterprise
False
True
Amount
week
0
2
4
6
8
10
12
14
10
20
30
40
50


Strategic Commissioning Board
False
3.75
3.70
3.65
3.60
3.55
3.50
3.45
Amount
25.0
25.5
26.0
26.5
27.0
week


Street Scene
False
True
Amount
week
0
100
200
300
400
500
0
10
20
30
40
50



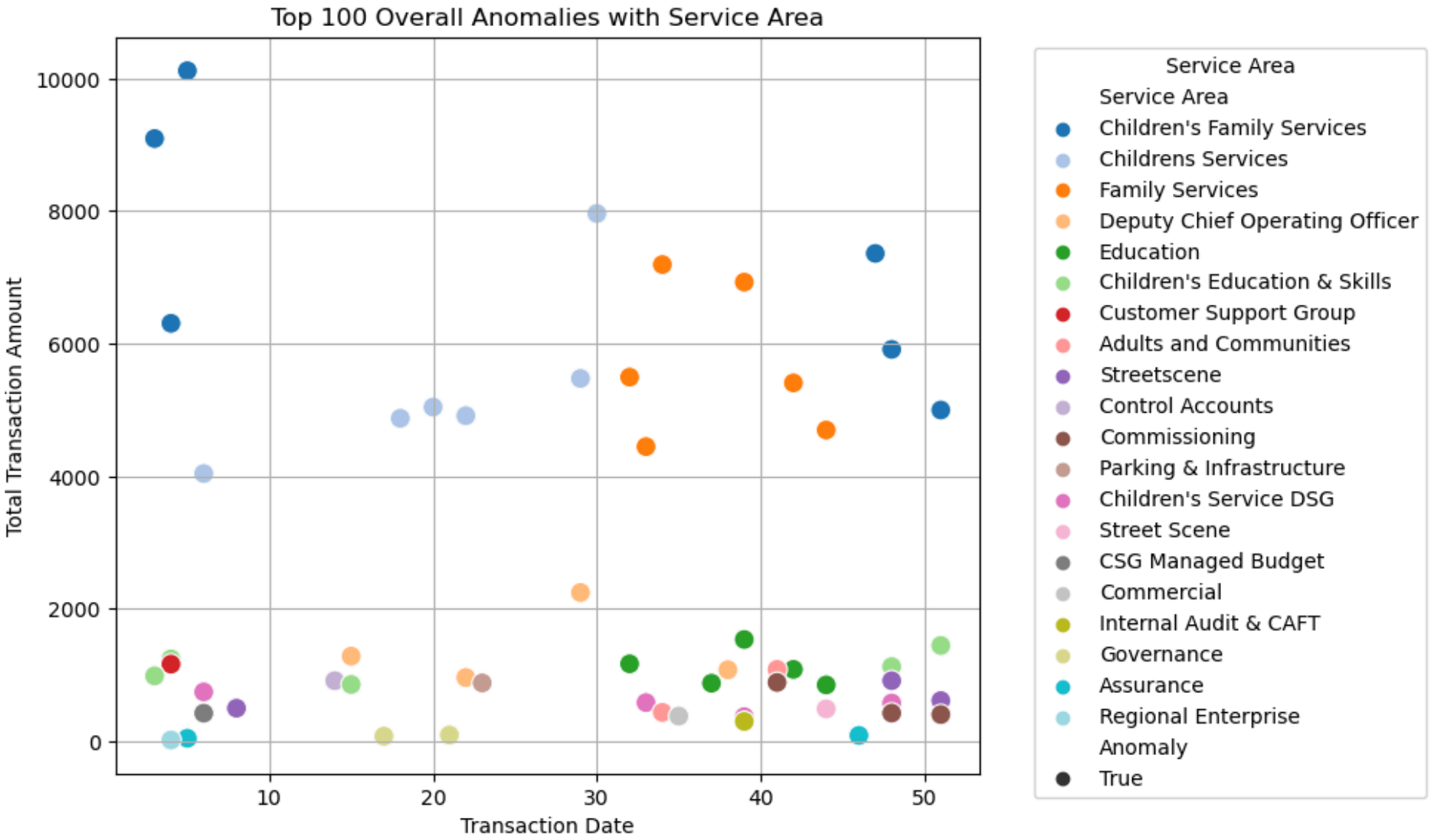
appendix

Question 5 part 6

top 100 anomaly visual representation

In [86]:
```
#  top 100 anomalies
top_anomalies = anomaly_encoded_df[anomaly_encoded_df['Anomaly']].nlargest(100, 'JV Value')

plt.figure(figsize=(10, 6))
```

```
sns.scatterplot(data=top_anomalies, x='Transaction Date', y='JV Value', hue='Service Area', palette='tab20', style='Anomaly', mark
plt.title('Top 100 Overall Anomalies with Service Area')
plt.xlabel('Transaction Date')
plt.ylabel('Total Transaction Amount')
plt.legend(title='Service Area', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```



********** END **********