Corporate Credit Card Transactions

London Borough of Barnet

Analysis of Credit Card Transactions, 2014-16 - 17?

Prepared by Danilo Diaz , March 2024

C0889539

Installation of Libraries

In [376]:
```
!pip install tabulate
!pip install -U scikit-learn
```

```
Requirement already satisfied: tabulate in /Users/ddiaz/.pyenv/versions/3.12.1/envs/VisualAi/lib/python3.1
2/site-packages (0.9.0)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: scikit-learn in /Users/ddiaz/.pyenv/versions/3.12.1/envs/VisualAi/lib/python
3.12/site-packages (1.4.1.post1)
Requirement already satisfied: numpy<2.0,>=1.19.5 in /Users/ddiaz/.pyenv/versions/3.12.1/envs/VisualAi/lib/
python3.12/site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /Users/ddiaz/.pyenv/versions/3.12.1/envs/VisualAi/lib/python
3.12/site-packages (from scikit-learn) (1.12.0)
Requirement already satisfied: joblib>=1.2.0 in /Users/ddiaz/.pyenv/versions/3.12.1/envs/VisualAi/lib/pytho
n3.12/site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/ddiaz/.pyenv/versions/3.12.1/envs/VisualAi/li
b/python3.12/site-packages (from scikit-learn) (3.3.0)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: pip install --upgrade pip
```

Libraries

In [459]:
```
import datetime
from tabulate import tabulate
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import norm
from sklearn.cluster import KMeans
%matplotlib inline
#hide warnings
import warnings
warnings.filterwarnings('ignore')
```

Table of Contents

Executive Summary

Analysis Requests

1. The Auditor would like to get a summary view of the transactions for each Service Area. The summary view would include at least one visual representation of the transactions in such a way that they could compare them by quarter. Quarters are defined based on the calendar year (Q1 is January to March, Q2 April to June, etc.). The Auditor would also like a summary table with some relevant statistics (The Auditor says something along the lines of “transaction counts and averages”, but welcomes ideas).
2. The Auditor would like to get a view if there are any significant changes in spending behavior by Service Area and by Account. Changes in behavior could be spikes, but could also be permanent increases in the transaction amounts. Please identify instances of both or show that they do not exist in the data.
3. The Auditor would like to get an understanding of how Creditors are classified into accounts. In particular, they are worried about transaction misclassification. Are you able to identify instances in which Creditors are not consistently classified into Accounts (e.g., most of the time Creditor “AirTickets.com” is classified into “Travelling Expenses”, but on some occasions it is also found in “Miscellaneous”)?
4. In terms of spending behavior (defined by the number and the typical size of transactions), are there Service Areas that behave similarly and can be grouped together? How?
5. The auditor has heard that you may know anomaly detection techniques. They would like to ask you for a sample of a few hundred transactions that are anomalous, different or worthwhile inquiring about. The sample should include at least five transactions for each Service Area. Please provide this sample and explain why they are special or different.

2. The Data

In [378]:
```
df_2014 = pd.read_csv(filepath_or_buffer='data/Purchasing Card Data 2014 v1.csv', sep=',')
df_2015 = pd.read_csv(filepath_or_buffer='data/PCard Transactions 15-16.txt', sep=',')
df_2016 = pd.read_csv(filepath_or_buffer='data/PCard 1617.csv', sep=',')
df_2017 = pd.read_csv(filepath_or_buffer='data/1718Pcard.csv', sep=',')
```

First Rows of the Data 2014 - 2015 - 2016 - 2017

In [379]:
```
display(df_2014.head(3))
```

	Service Area	Account Description	Creditor	Transaction Date	JV Reference	JV Date	JV Value
0	Childrens Services	IT Services	123-REG.CO.UK	23/04/2014	93	20/05/2014	143.81
1	Childrens Services	Other Services	ACCESS EXPEDITIONS	03/04/2014	111	20/05/2014	6,000.00
2	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	02/04/2014	6	20/05/2014	309.38

In [380]:
```
display(df_2015.head(3))
```

	Service Area	Account Description	Creditor	Journal Date	Journal Reference	Total
0	Assurance	Miscellaneous Expenses	43033820 COSTA COFFEE	18/08/2015	5,043.00	2
1	Children's Family Services	Miscellaneous Expenses	99 PLUS DISCOUNT MART	08/06/2015	4,184.00	29.97
2	Children's Family Services	E19 - Learning Resources	99P STORES LTD	07/12/2015	6,278.00	34.65

In [381]:
```
display(df_2016.head(3))
```

	Service Area	Account Description	Creditor	Journal Date	Journal Reference	Total
0	Adults and Communities	Books-CDs-Audio-Video	AMAZON EU	05/12/2016	10,510.00	45.00
1	Adults and Communities	Books-CDs-Audio-Video	AMAZON UK MARKETPLACE	05/12/2016	10,509.00	426.57
2	Adults and Communities	Books-CDs-Audio-Video	AMAZON UK RETAIL AMAZO	06/12/2016	10,524.00	121.38

In [382]:
```
display(df_2017.head(3))
```

	FIN.TRANSACTION DATE	FIN.POSTING DATE	FIN.TRANSACTION AMOUNT	MCH.MERCHANT NAME	MCH.CITY NAME	FIN.ORIGINAL CURRENCY AMOUNT	FIN.ORIGINAL ISO CURRENCY CODE SYMBOL	FIN.INE CONVERSIOI
0	06/04/17	07/04/17	36.55	TESCO STORE 2296	COLNEY HATCH	36.55	GBP	1.0
1	06/04/17	07/04/17	58.75	AMFBOWLING.CO.UK	01442 840200	58.75	GBP	1.0
2	10/04/17	11/04/17	40.50	WWW.GOJUMPIN.COM	INTERNET	40.50	GBP	1.0

In [383]:
```
#### Get Column Names of each dataframe

print(df_2014.columns)
print(df_2015.columns)
print(df_2016.columns)
print(df_2017.columns)
```

```
Index(['Service Area', 'Account Description', 'Creditor', 'Transaction Date',
       'JV Reference', 'JV Date', 'JV Value'],
      dtype='object')
Index(['Service Area', 'Account Description', 'Creditor', 'Journal Date',
       'Journal Reference', 'Total'],
      dtype='object')
Index(['Service Area', 'Account Description', 'Creditor', 'Journal Date',
       'Journal Reference', 'Total'],
      dtype='object')
Index(['FIN.TRANSACTION DATE', 'FIN.POSTING DATE', 'FIN.TRANSACTION AMOUNT',
       'MCH.MERCHANT NAME', 'MCH.CITY NAME', 'FIN.ORIGINAL CURRENCY AMOUNT',
       'FIN.ORIGINAL ISO CURRENCY CODE SYMBOL', 'FIN.INET CONVERSION'],
      dtype='object')
```

Base on the columns name we can inffer tha the data from 2017 is miising Service Area We can also see that the data from 2014,2015 and 2016 have the same columns names, so we can merge them.

In [384]:
```
# correct the column names
columns_names = ['Service Area',
                 'Account Description',
                 'Creditor',
                 'Journal Date',
                 'JV Reference',
                 'Total']
```

Columns Names 2014 - 2015 - 2016

In [385]:
```
#### Rename Columns 2014
df_2014.drop('JV Date', axis = 1, inplace = True)
df_2014.columns = columns_names

#### Rename Columns 2015
df_2015.columns = columns_names

#### Rename Columns 2016
df_2016.columns = columns_names
```

In [386]:
```
df_2014.head(3)
```

Out[386]:

	Service Area	Account Description	Creditor	Journal Date	JV Reference	Total
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

In [387]:
```
df_2015.head(3)
```

Out[387]:

	Service Area	Account Description	Creditor	Journal Date	JV Reference	Total
0	Assurance	Miscellaneous Expenses	43033820 COSTA COFFEE	18/08/2015	5,043.00	2
1	Children's Family Services	Miscellaneous Expenses	99 PLUS DISCOUNT MART	08/06/2015	4,184.00	29.97
2	Children's Family Services	E19 - Learning Resources	99P STORES LTD	07/12/2015	6,278.00	34.65

In [388]:
```
df_2016.head(3)
```

Out[388]:

	Service Area	Account Description	Creditor	Journal Date	JV Reference	Total
0	Adults and Communities	Books-CDs-Audio-Video	AMAZON EU	05/12/2016	10,510.00	45.00
1	Adults and Communities	Books-CDs-Audio-Video	AMAZON UK MARKETPLACE	05/12/2016	10,509.00	426.57
2	Adults and Communities	Books-CDs-Audio-Video	AMAZON UK RETAIL AMAZO	06/12/2016	10,524.00	121.38

In [389]:
```
# shape of the dataframes
print('2014:', df_2014.shape)
print('2015:', df_2015.shape)
print('2016:', df_2016.shape)
```

```
2014: (4142, 6)
2015: (3865, 6)
2016: (4582, 6)
```

Concatanating the data

In [390]:
```
data = pd.concat((df_2014, df_2015, df_2016), axis = 0).reset_index(drop = True)
```

In [391]:
```
# shape of the data
print('Data Shape:', data.shape)
```

```
Data Shape: (12589, 6)
```

Get some information about the data

In [392]:
```
### check shape of the data
data.shape
```

Out[392]:
```
(12589, 6)
```

In [393]:
```
## Check duplicates

print('Duplicates:', data.duplicated().sum())
```

```
Duplicates: 0
```

In [394]:
```
## Check missing values
print('Missing Values:', data.isnull().sum())

# Percentage of missing values

print('Percentage of Missing Values:', data.isnull().sum()/len(data)*100)
```

```
Missing Values: Service Area           1
Account Description    2
Creditor               2
Journal Date           2
JV Reference           2
Total                  0
dtype: int64
Percentage of Missing Values: Service Area          0.01
Account Description   0.02
Creditor              0.02
Journal Date          0.02
JV Reference          0.02
Total                 0.00
dtype: float64
```

In [395]:
```
### check the data types
print('Data Types:', data.dtypes)
```

```
Data Types: Service Area           object
Account Description     object
Creditor                object
Journal Date            object
JV Reference           float64
Total                   object
dtype: object
```

In [396]:
```
### Check the data
data.tail(3)
```

Out[396]:

	Service Area	Account Description	Creditor	Journal Date	JV Reference	Total
12586	Streetscene	Vehicle Running Costs	WWW.DVLA.GOV.UK	25/08/2016	9,248.00	232.50
12587	Streetscene	Vehicle Running Costs	WWW.MOT-TESTING-CP	25/11/2016	10,384.00	68.00
12588	Grand Total	NaN	NaN	NaN	NaN	471,044.01

In [397]:
```
### Get unique values from categorical columns and count them

for col in data.columns:
    if data[col].dtype == 'object':
        print(col, data[col].nunique())
```

```
Service Area 25
Account Description 67
Creditor 1936
Journal Date 739
Total 6292
```

Comments on Data

Based on the information, it is evident that the data comprises 6 columns and 12,589 rows. Additionally:

- There are some missing values; however, the percentage is low, so we can proceed to drop them.
- No duplicated values are present in the dataset.
- The data types of certain columns are incorrect, necessitating a change in their data types.

Data Wrangling

In [398]:
```
# drop missing values
data.dropna(inplace = True)
# check the shape of the data
print('Data Shape:', data.shape)
#  journal date to datetime
data['Journal Date'] = pd.to_datetime(data['Journal Date'])

#total to float
data['Total'] = data['Total'].str.replace('$', '').str.replace(',', '').astype(float)

# check the data types
data.dtypes
```

```
Data Shape: (12587, 6)
```

Out[398]:
```
Service Area                    object
Account Description             object
Creditor                        object
Journal Date           datetime64[ns]
JV Reference                  float64
Total                         float64
dtype: object
```

Columns Types

In [399]:
```
# numerical columns
numerical_cols = data.select_dtypes(include = [np.number]).columns
# categorical columns
categorical_cols = data.select_dtypes(exclude = [np.number, np.datetime64]).columns
# time columns
time_cols = data.select_dtypes(include = [np.datetime64]).columns


print('Numerical Columns:', numerical_cols)

print('Categorical Columns:', categorical_cols)
```

```
Numerical Columns: Index(['JV Reference', 'Total'], dtype='object')
Categorical Columns: Index(['Service Area', 'Account Description', 'Creditor'], dtype='object')
```

Numerical Columns Analysis

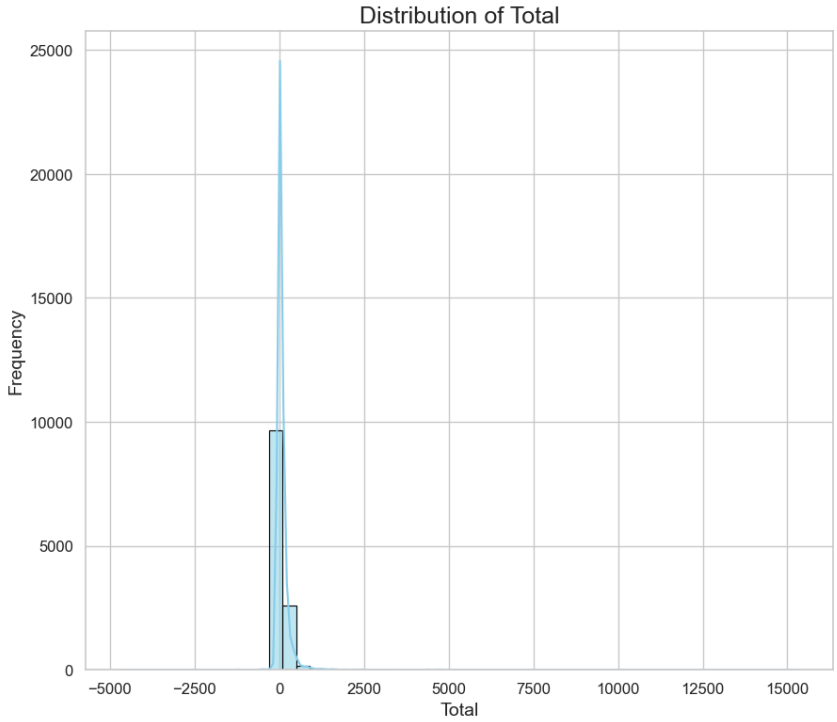
In [400]:
```
### Get the summary statistics of the numerical columns
data[numerical_cols].describe()
```

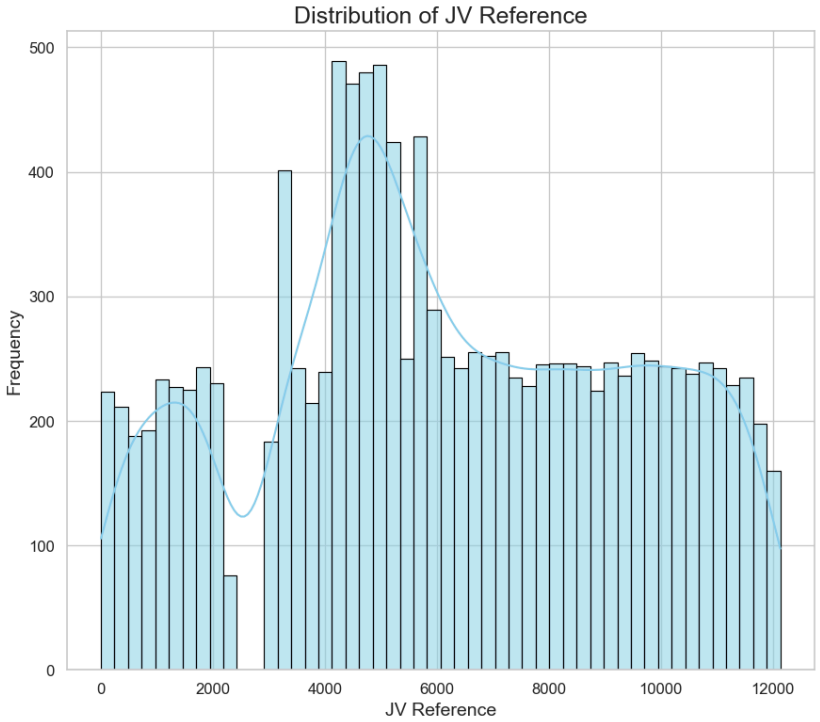
Out[400]:

	JV Reference	Total
count	12,587.00	12,587.00
mean	6,126.85	99.82
std	3,248.97	391.39
min	1.00	-4,707.00
25%	3,943.50	10.00
50%	5,795.00	26.83
75%	8,847.50	92.05
max	12,136.00	15,340.80

In [401]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set style for better aesthetics

# Create subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(18, 8))

# Plot distribution of 'Total' column
sns.histplot(data['Total'], bins=50, kde=True, color='skyblue', edgecolor='black', ax=axes[0])
axes[0].set_title('Distribution of Total', fontsize=18)
axes[0].set_xlabel('Total', fontsize=14)
axes[0].set_ylabel('Frequency', fontsize=14)
axes[0].tick_params(axis='both', labelsize=12)

# Plot distribution of 'JV Reference' column
sns.histplot(data['JV Reference'], bins=50, kde=True, color='skyblue', edgecolor='black', ax=axes[1])
axes[1].set_title('Distribution of JV Reference', fontsize=18)
axes[1].set_xlabel('JV Reference', fontsize=14)
axes[1].set_ylabel('Frequency', fontsize=14)
axes[1].tick_params(axis='both', labelsize=12)

# Adjust layout
plt.tight_layout()
plt.show()
```


Distribution of Total
Frequency
Total
25000
20000
15000
10000
5000
0
−5000
−2500
0
2500
5000
7500
10000
12500
15000


Distribution of JV Reference
Frequency
JV Reference
500
400
300
200
100
0
0
2000
4000
6000
8000
10000
12000

In [402]:
```
### Get maximum value of total and its index

max_total = data['Total'].max()
max_total_index = data['Total'].idxmax()

print('Max Total:', max_total)

print('Max Total Index:', max_total_index)

### Get the row with the maximum total
data.loc[max_total_index]
```

```
Max Total: 15340.8
Max Total Index: 5730
```

Out[402]:
```
Service Area           Customer Support Group
Account Description          Fees and Charges
Creditor                   HMCOURTS-SERVICE.G
Journal Date              2015-05-20 00:00:00
JV Reference                         3,161.00
Total                               15,340.80
Name: 5730, dtype: object
```

In [403]:
```
# get the top 10 highest total

top_10_total = data.nlargest(10, 'Total')

print('Top 10 Total:', top_10_total)
```

```
Top 10 Total:                  Service Area   Account Description           Creditor  \
5730   Customer Support Group      Fees and Charges  HMCOURTS-SERVICE.G
12350  Customer Support Group  Legal and Court Fees  HMCOURTS-SERVICE.G
12353  Customer Support Group  Legal and Court Fees  HMCOURTS-SERVICE.G
1856       CSG Managed Budget  Legal and Court Fees  HMCOURTS-SERVICE.G
12359  Customer Support Group  Legal and Court Fees  HMCOURTS-SERVICE.G
531        CSG Managed Budget  Legal and Court Fees  HMCOURTS-SERVICE.G
5731   Customer Support Group      Fees and Charges  HMCOURTS-SERVICE.G
532        CSG Managed Budget  Legal and Court Fees  HMCOURTS-SERVICE.G
12351  Customer Support Group  Legal and Court Fees  HMCOURTS-SERVICE.G
77                 Governance        Other Services  BETTER LIFE HEALTH

      Journal Date  JV Reference     Total
5730    2015-05-20      3,161.00 15,340.80
12350   2016-06-23      8,504.00 11,487.00
12353   2016-05-26      8,190.00 11,088.00
1856    2014-09-23      2,235.00  8,058.00
12359   2017-01-30     10,995.00  7,968.00
531     2014-05-22        498.00  7,800.00
5731    2015-07-21      4,722.00  6,955.20
532     2014-05-22        497.00  6,777.00
12351   2016-11-23     10,357.00  6,762.00
77      2014-04-24        286.00  6,388.20
```

Type *Markdown* and LaTeX: α^2

- Both columns have a similar count, indicating that there are no missing values.
- 'JV Reference' has a higher mean and standard deviation compared to 'Total.'
- 'Total' has a wider range, especially with a negative minimum value, suggesting potential outliers or errors in the data.

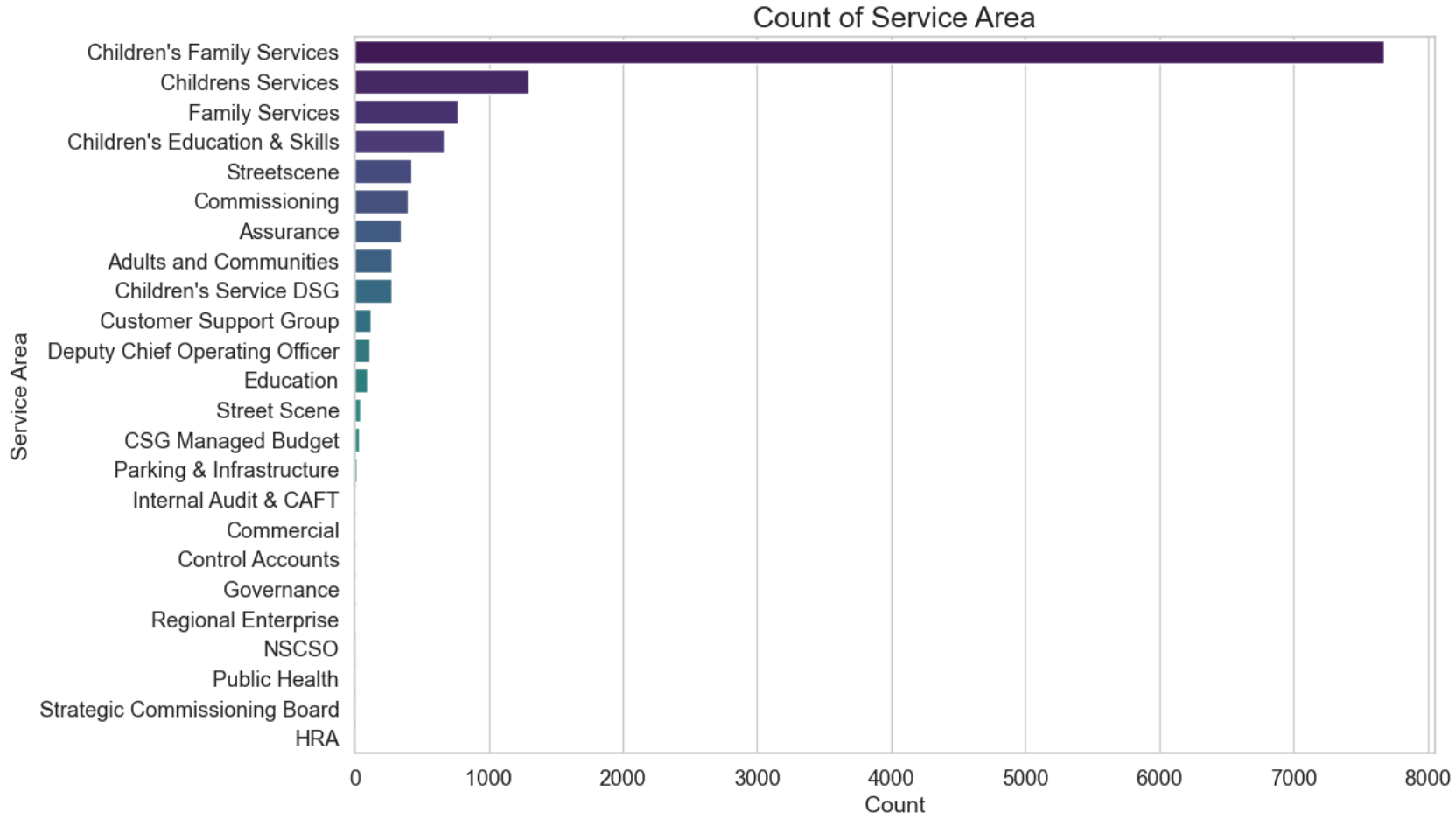
Categorical Columns Analysis

In [404]:
```
data.head(3)
```

Out[404]:

	Service Area	Account Description	Creditor	Journal Date	JV Reference	Total
0	Childrens Services	IT Services	123-REG.CO.UK	2014-04-23	93.00	143.81
1	Childrens Services	Other Services	ACCESS EXPEDITIONS	2014-04-03	111.00	6,000.00
2	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	2014-04-02	6.00	309.38

In [405]:
```
# Plot All service areas
plt.figure(figsize=(12, 8))
sns.countplot(y='Service Area', data=data, palette='viridis', order=data['Service Area'].value_counts().inde
plt.title('Count of Service Area', fontsize=18)
plt.xlabel('Count', fontsize=14)
plt.ylabel('Service Area', fontsize=14)
plt.show()

## Nunbers of service areas

print('Number of Service Areas:', data['Service Area'].nunique())
```



```
Number of Service Areas: 24
```

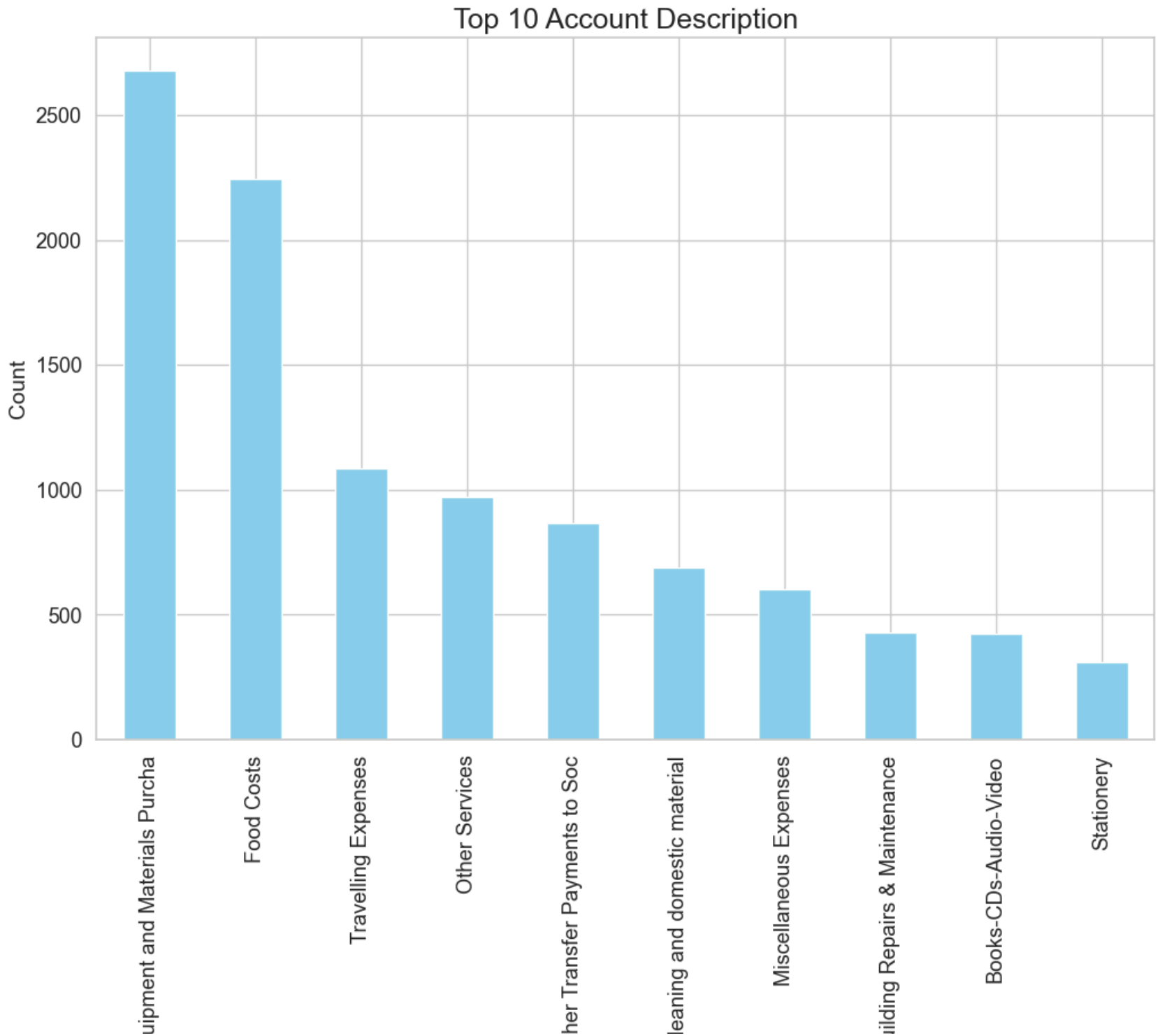
In [406]:

```python
### Get the top 10 service areas

top_10_service_area = data['Service Area'].value_counts().nlargest(10)

print('Top 10 Service Area:', top_10_service_area)
```

```
Top 10 Service Area: Service Area
Children's Family Services       7672
Childrens Services               1297
Family Services                   770
Children's Education & Skills     667
Streetscene                       420
Commissioning                     400
Assurance                         344
Adults and Communities            278
Children's Service DSG            277
Customer Support Group            117
Name: count, dtype: int64
```

In [407]:
```
# Plot all accounts and count

print('Number of Account Description:', data['Account Description'].nunique())
top_10_account_description = data['Account Description'].value_counts().nlargest(10)
print('Top 10 Account Description:', top_10_account_description)


# Plot just top 10 accounts

top_10_account_description.plot(kind='bar', figsize=(12, 8), color='skyblue')
plt.title('Top 10 Account Description', fontsize=18)
plt.xlabel('Account Description', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.show()
```

```
Number of Account Description: 67
Top 10 Account Description: Account Description
Equipment and Materials Purcha    2676
Food Costs                        2245
Travelling Expenses               1084
Other Services                     971
Other Transfer Payments to Soc     865
Cleaning and domestic material     688
Miscellaneous Expenses             603
Building Repairs & Maintenance     431
Books-CDs-Audio-Video              425
Stationery                         312
Name: count, dtype: int64
```


Top 10 Account Description
Count
2500
2000
1500
1000
500
0
uipment and Materials Purcha
Food Costs
Travelling Expenses
Other Services
her Transfer Payments to Soc
leaning and domestic material
Miscellaneous Expenses
ilding Repairs & Maintenance
Books-CDs-Audio-Video
Stationery

Eq Ot Cl Bu

Account Description

In [408]:
```
creditors = np.unique(data['Creditor'])
num_creditors = len(creditors)
print('Number of Creditors:', num_creditors)
```

```
Number of Creditors: 1936
```

In [409]:
```
earliest_date = data['Journal Date'].min()
latest_date = data['Journal Date'].max()

# Create a list of tuples for tabulating
table_data = [
    ('Earliest Date', earliest_date),
    ('Latest Date', latest_date)
]

table = tabulate(table_data, headers=['Date Type', 'Date'], tablefmt='fancy_grid')
print(table)
```

Date Type	Date
Earliest Date	2014-04-02 00:00:00
Latest Date	2017-04-03 00:00:00

Quarter Column

In [410]:
```
# convert date strings to datetime and create quarters column
data['Quarter'] = data['Journal Date'].dt.to_period('Q')


num_transactions_per_quarter = data['Quarter'].value_counts().sort_index(ascending=True)

#Print unique quarters

print('Number Quarters:', len(data['Quarter'].unique()))
```

```
Number Quarters: 13
```

2. Summary Graphs and Tables

Create a class to make the analysis easier, based on the requests of the Auditor. taking account quarter, service area, account and creditor.

In [411]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

class QuarterAggPlotter:
    def __init__(self, data):
        self.data = data
        self.quarters = data['Quarter'].unique()
        self.num_quarters = len(data['Quarter'].unique())
        self.service_areas = data['Service Area'].unique()
        self.num_service_areas = len(data['Service Area'].unique())

    def plot_single_subplot(self, ax, heights, color, agg_name, title):
        """
        Plots a single subplot with bar chart.

        Parameters:
        - ax: matplotlib axes object
        - heights: list or array of bar heights
        - color: color of the bars
        - agg_name: label for the y-axis
        - title: title of the subplot
        """
        ind = np.arange(self.num_quarters)
        width = 0.35

        bars = ax.bar(ind, heights, width, color=color)

        ax.set_xticks(ind + (width / 2))
        ax.set_xticklabels(self.quarters)

        ax.set_xlim(-0.5, self.num_quarters)

        ax.set_xlabel("Quarter")
        ax.set_ylabel(agg_name)
        ax.set_title(title, y=1.05, fontsize=14)


        for bar in bars:
            height = bar.get_height()
            ax.annotate(f'{height:.2f}',
                        xy=(bar.get_x() + bar.get_width() / 2, height),
```

```
                    xytext=(0, 3),
                    textcoords="offset points",
                    ha='center', va='bottom', fontsize=10, color='black')

def plot_transactions_overall(self, table, agg_name, color = 'green'):

    # allocate subplots
    fig = plt.figure(figsize = (12, 2))
    ax = fig.add_subplot(1, 1, 1)

    self.plot_single_subplot(ax, table[0], color, agg_name, "All Service Areas".upper())

    # show plot
    plt.show()

def create_qtr_agg_table_overall(self, agg_func):
    """
    Creates a pivot table for a given aggregate function.

    Parameters:
    - agg_func: function to aggregate data

    Returns:
    - pandas DataFrame with aggregated values for each quarter
    """

    aggs = np.zeros(self.num_quarters)
    for qtr in range(self.num_quarters):
        aggs[qtr] = agg_func(data[data['Quarter'] == self.quarters[qtr]]['Total'])
    aggs[np.isnan(aggs) == True] = 0.0
    aggs_df = pd.DataFrame(aggs.reshape(-1, 1))
    aggs_df.index = self.quarters

    return aggs_df

def create_qtr_agg_table(self, agg_func):
    aggs = np.zeros((self.num_service_areas, self.num_quarters))
    for area in range(self.num_service_areas):
        for qtr in range(self.num_quarters):
            aggs[area, qtr] = agg_func(data[(data['Service Area'] == self.service_areas[area]) & \
                                            (data['Quarter'] == self.quarters[qtr])]['Total'])
    aggs[np.isnan(aggs) == True] = 0.0
    aggs_df = pd.DataFrame(aggs)
```

```
        aggs_df.columns = self.quarters
        aggs_df.index = self.service_areas

        return aggs_df

    def plot_aggregated_data(self, agg_func, color, agg_name, title):


        # Create a figure and axis
        fig, ax = plt.subplots(1, 1, figsize=(12, 6))

        # Create aggregated table
        aggs_df = self.create_qtr_agg_table_overall(agg_func)

        # Plot single subplot
        self.plot_single_subplot(ax, aggs_df[0], color, agg_name, title)
        # Show the plot
        plt.show()

    def plot_tables(self, table, agg_name, color = 'darkblue'):

        # allocate subplots
        fig, ax_ar = plt.subplots(self.num_service_areas, 1, figsize = (12, 55))

        # create all subplots
        for index in range(self.num_service_areas):
            self.plot_single_subplot(ax_ar[index], table.iloc[index, :], color, agg_name, self.service_areas

        # show plot
        plt.tight_layout()
        plt.show()

    def summary_table(self):
        """
        Creates a summary table for the data.

        Returns:
        - pandas DataFrame with aggregated values for each quarter
        """

        # Create aggregated table
        pd.options.display.float_format = '{:,.0f}'.format
        pd.set_option('display.max_rows', self.num_service_areas * self.num_quarters)
```

```
        data_qtr = self.data.groupby(['Service Area', 'Quarter'])
        summary = data_qtr[['Total']].agg((np.count_nonzero, np.sum, np.mean))

        summary.columns = summary.columns.set_levels(['Transactions'], level = 0)
        summary.columns = summary.columns.set_levels(['Count', 'Sum ($)', 'Average ($)'], level = 1)

        return summary
```

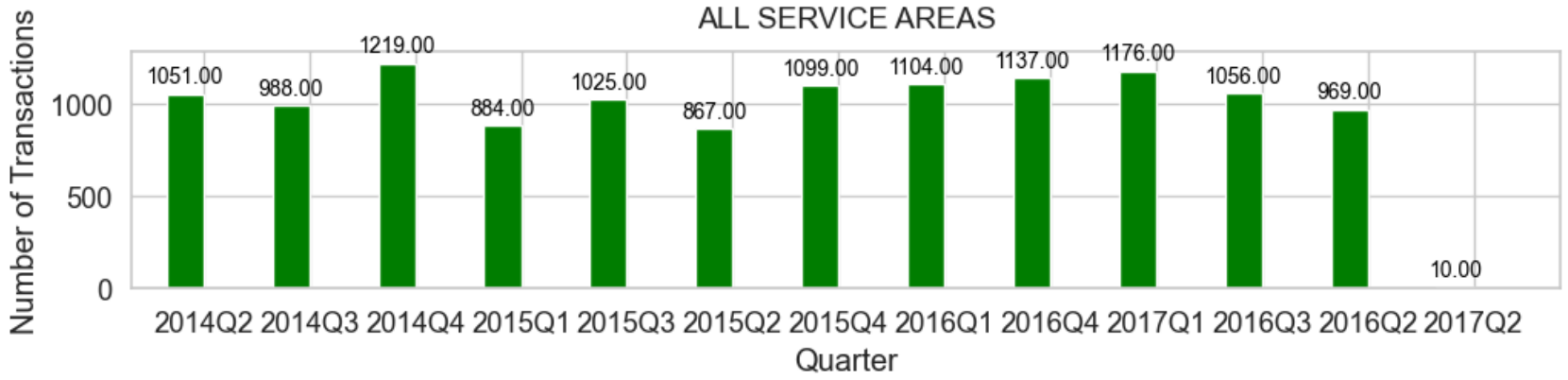
Intance of the class

In [412]:
```
# Create an instance of the QuarterAggPlotter class
quarter_agg_plotter = QuarterAggPlotter(data)
```

We offer comprehensive support for our graphical data through the inclusion of four summary tables. The primary table employs a hierarchical structure to present a detailed breakdown of quarterly spending across service areas, exclusively featuring active quarters. This table encompasses counts, sums, and averages for a holistic perspective. Additionally, the subsequent trio of tables provides a granular breakdown of the three aggregates, organized by both service area and quarter. While the hierarchical table facilitates a focused examination of specific service areas, the grid tables prove invaluable for scrutinizing trends over time.

Transactions Counts by Quarter

In [413]:
```
### Transaction Counts by Quarter
services_areas_counts_qtr_overall = quarter_agg_plotter.create_qtr_agg_table_overall(np.count_nonzero)
quarter_agg_plotter.plot_transactions_overall(services_areas_counts_qtr_overall, 'Number of Transactions', '
```



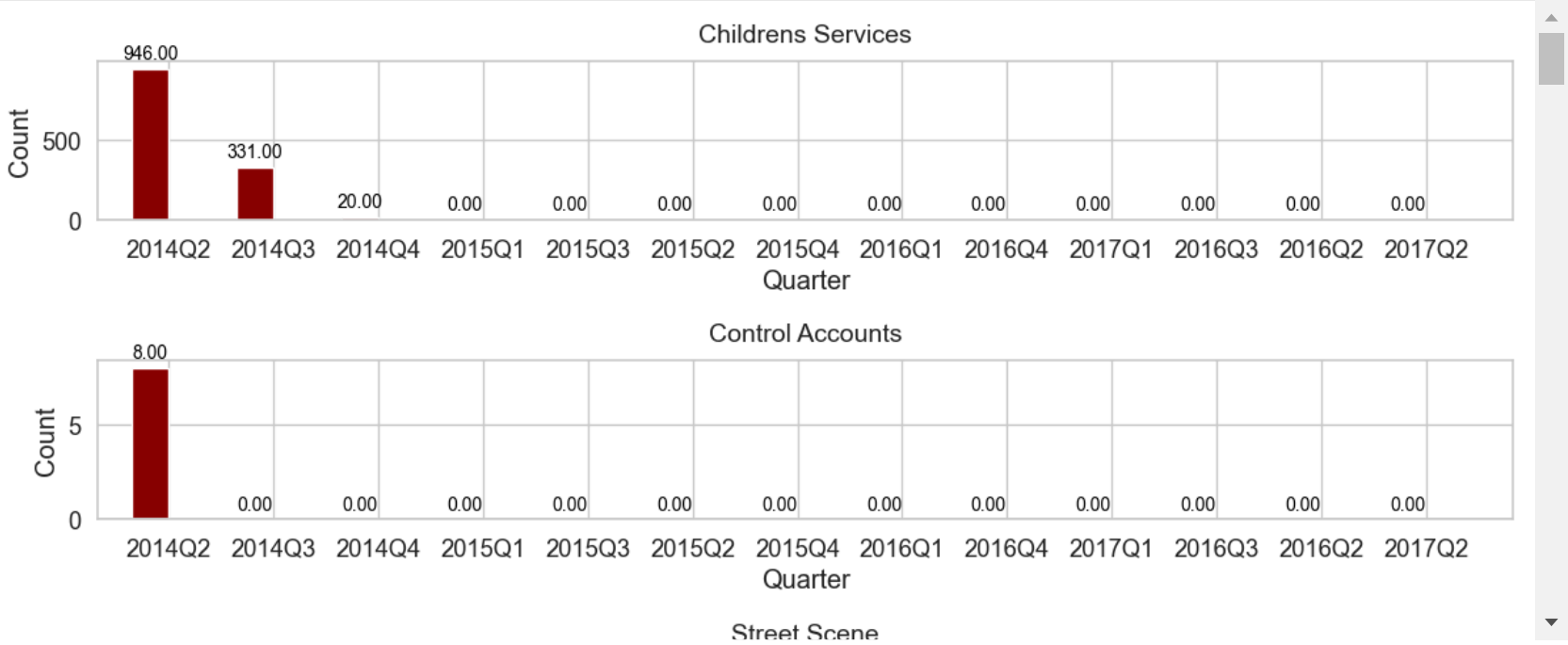
In [414]:
```
services_areas_counts_qtr = quarter_agg_plotter.create_qtr_agg_table(np.count_nonzero)
```

In [415]:
```
display(services_areas_counts_qtr)
```

	2014Q2	2014Q3	2014Q4	2015Q1	2015Q3	2015Q2	2015Q4	2016Q1	2016Q4	2017Q1	2016Q3	2016Q2	2017Q2
Childrens Services	946.00	331.00	20.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Control Accounts	8.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Street Scene	11.00	16.00	12.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Governance	4.00	3.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Deputy Chief Operating Officer	40.00	49.00	24.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Public Health	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
Adults and Communities	16.00	8.00	18.00	11.00	15.00	8.00	19.00	23.00	45.00	52.00	37.00	25.00	1.00
Internal Audit & CAFT	2.00	7.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NSCSO	1.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CSG Managed Budget	20.00	12.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Strategic Commissioning Board	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Family Services	0.00	460.00	310.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Children's Service DSG	0.00	31.00	32.00	15.00	20.00	12.00	59.00	31.00	24.00	10.00	19.00	24.00	0.00
Education	0.00	60.00	35.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Commercial	0.00	9.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Children's Family Services	0.00	0.00	597.00	607.00	750.00	590.00	814.00	867.00	895.00	980.00	804.00	758.00	8.00
Commissioning	0.00	0.00	44.00	29.00	18.00	11.00	40.00	62.00	52.00	38.00	63.00	43.00	0.00
Streetscene	0.00	0.00	25.00	37.00	72.00	23.00	36.00	27.00	48.00	41.00	73.00	37.00	1.00
Parking & Infrastructure	0.00	0.00	2.00	1.00	0.00	0.00	1.00	0.00	2.00	4.00	1.00	1.00	0.00
Children's Education & Skills	0.00	0.00	72.00	165.00	96.00	160.00	86.00	55.00	6.00	2.00	15.00	10.00	0.00

	2014Q2	2014Q3	2014Q4	2015Q1	2015Q3	2015Q2	2015Q4	2016Q1	2016Q4	2017Q1	2016Q3	2016Q2	2017Q2
Customer Support Group	0.00	0.00	18.00	14.00	15.00	11.00	13.00	9.00	10.00	10.00	7.00	10.00	0.00
Assurance	0.00	0.00	3.00	4.00	39.00	52.00	30.00	29.00	53.00	38.00	36.00	60.00	0.00
Regional Enterprise	0.00	0.00	0.00	1.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00
HRA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00

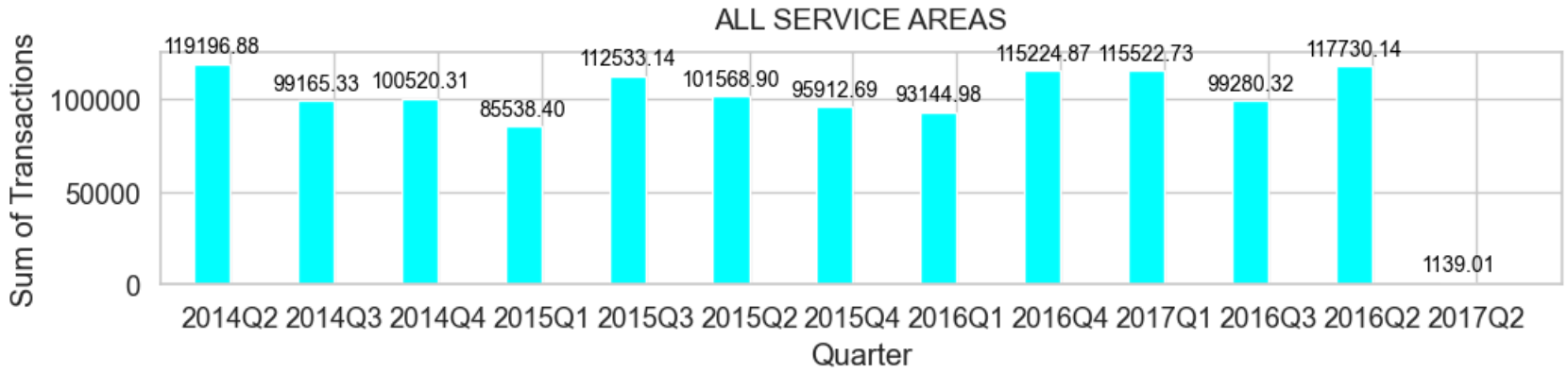
In [416]:
```
quarter_agg_plotter.plot_tables(services_areas_counts_qtr, 'Count', 'darkred')
```



In [ ]:
```
```

Transaction Sums by Quarter

In [417]:
```
### Transaction Sum by Quarter
services_areas_sum_qtr_overall = quarter_agg_plotter.create_qtr_agg_table_overall(np.sum)
quarter_agg_plotter.plot_transactions_overall(services_areas_sum_qtr_overall, 'Sum of Transactions', 'cyan')
```



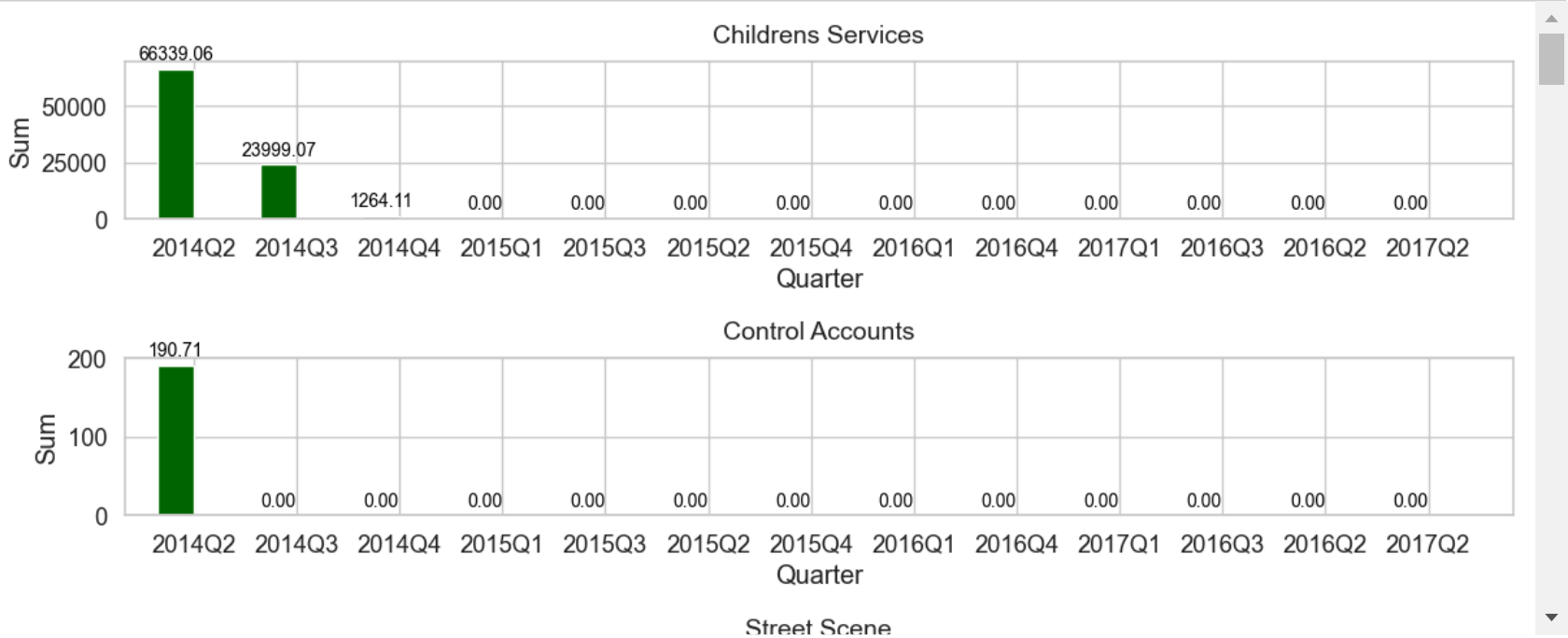
In [418]:
```
services_areas_sum_qtr = quarter_agg_plotter.create_qtr_agg_table(np.sum)
```

In [419]:
```
display(services_areas_sum_qtr)
```

	2014Q2	2014Q3	2014Q4	2015Q1	2015Q3	2015Q2	2015Q4	2016Q1	2016Q4	2017Q1	2016Q3	
Childrens Services	66,339.06	23,999.07	1,264.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Control Accounts	190.71	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Street Scene	630.08	570.63	1,331.55	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Governance	13,011.60	1,176.96	53.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Deputy Chief Operating Officer	1,591.24	1,728.23	848.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Public Health	-2.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Adults and Communities	4,608.00	2,570.45	2,149.33	2,017.86	1,635.85	1,187.40	3,039.03	2,468.52	9,117.74	9,165.53	6,742.40	
Internal Audit & CAFT	407.20	192.95	116.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
NSCSO	10.00	445.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
CSG Managed Budget	32,167.34	24,549.00	14,270.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Strategic Commissioning Board	244.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Family Services	0.00	30,283.29	17,694.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Children's Service DSG	0.00	3,047.75	5,213.69	3,030.71	2,776.49	1,543.27	4,530.62	4,457.30	1,599.30	168.33	2,618.12	
Education	0.00	7,858.45	4,956.69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Commercial	0.00	2,743.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Children's Family Services	0.00	0.00	33,508.71	39,858.59	56,767.84	43,137.14	47,277.81	47,883.39	69,385.05	74,350.63	55,320.69	5
Commissioning	0.00	0.00	3,425.41	5,189.44	3,401.85	2,776.29	8,602.64	8,434.73	7,314.04	2,331.35	8,138.99	
Streetscene	0.00	0.00	2,078.24	5,548.83	14,608.48	2,801.73	5,390.81	5,037.15	4,773.59	6,205.94	11,097.15	
Parking & Infrastructure	0.00	0.00	92.82	28.43	0.00	0.00	159.67	0.00	3,568.25	234.68	76.25	

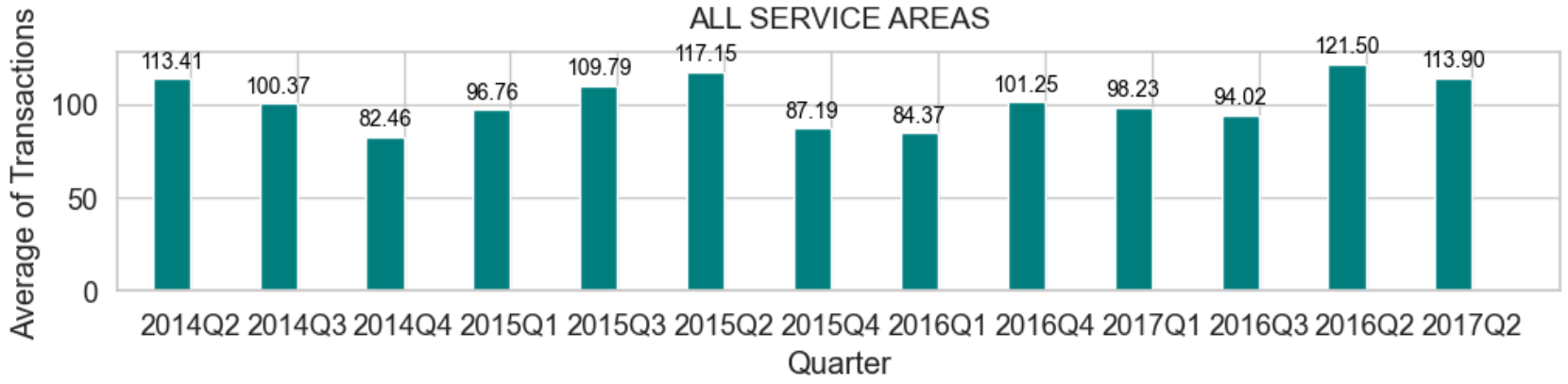
	2014Q2	2014Q3	2014Q4	2015Q1	2015Q3	2015Q2	2015Q4	2016Q1	2016Q4	2017Q1	2016Q3	
Children's Education & Skills	0.00	0.00	8,035.56	10,836.12	10,777.72	12,524.97	11,115.55	7,481.79	1,253.91	116.10	2,229.14	
Customer Support Group	0.00	0.00	5,375.31	18,900.97	20,302.63	34,636.00	12,041.55	16,472.82	15,911.30	21,554.00	13,197.00	2
Assurance	0.00	0.00	105.34	67.45	2,262.28	2,962.10	2,110.01	849.28	1,785.75	1,170.17	-151.42	
Regional Enterprise	0.00	0.00	0.00	60.00	0.00	0.00	1,645.00	60.00	226.00	226.00	12.00	
HRA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	289.94	0.00	0.00	

In [420]:
```
quarter_agg_plotter.plot_tables(services_areas_sum_qtr, 'Sum', 'darkgreen')
```



Transaction Average by Quarter

In [421]:
```
### Transaction Sum by Quarter
services_areas_average_qtr_overall = quarter_agg_plotter.create_qtr_agg_table_overall(np.mean)
quarter_agg_plotter.plot_transactions_overall(services_areas_average_qtr_overall, 'Average of Transactions',
```



In [422]:
```
services_areas_average_qtr = quarter_agg_plotter.create_qtr_agg_table(np.mean)
display(services_areas_average_qtr)
```

	2014Q2	2014Q3	2014Q4	2015Q1	2015Q3	2015Q2	2015Q4	2016Q1	2016Q4	2017Q1	2016Q3	2016Q2	201
Childrens Services	70.13	72.50	63.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Control Accounts	23.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Street Scene	57.28	35.66	110.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Governance	3,252.90	392.32	53.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Deputy Chief Operating Officer	39.78	35.27	35.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Public Health	-1.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.55	(
Adults and Communities	288.00	321.31	119.41	183.44	109.06	148.43	159.95	107.33	202.62	176.26	182.23	146.37	7
Internal Audit & CAFT	203.60	27.56	58.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
NSCSO	10.00	222.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
CSG Managed Budget	1,608.37	2,045.75	3,567.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Strategic Commissioning Board	244.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Family Services	0.00	65.83	57.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Children's Service DSG	0.00	98.31	162.93	202.05	138.82	128.61	76.79	143.78	66.64	16.83	137.80	139.28	(
Education	0.00	130.97	141.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Commercial	0.00	304.78	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(
Children's Family Services	0.00	0.00	56.13	65.66	75.69	73.11	58.01	55.23	77.44	75.87	68.81	72.48	12
Commissioning	0.00	0.00	77.85	178.95	188.99	252.39	215.07	136.04	140.65	61.35	129.19	171.40	(
Streetscene	0.00	0.00	83.13	149.97	202.90	121.81	149.74	186.56	99.45	151.36	152.02	241.34	8
Parking & Infrastructure	0.00	0.00	46.41	28.43	0.00	0.00	159.67	0.00	1,784.12	58.67	76.25	500.00	(

	2014Q2	2014Q3	2014Q4	2015Q1	2015Q3	2015Q2	2015Q4	2016Q1	2016Q4	2017Q1	2016Q3	2016Q2	201
Children's Education & Skills	0.00	0.00	111.60	65.67	112.27	78.28	129.25	136.03	208.98	58.05	148.61	48.17	
Customer Support Group	0.00	0.00	298.63	1,350.07	1,353.51	3,148.73	926.27	1,830.31	1,591.13	2,155.40	1,885.29	2,898.88	
Assurance	0.00	0.00	35.11	16.86	58.01	56.96	70.33	29.29	33.69	30.79	-4.21	158.50	
Regional Enterprise	0.00	0.00	0.00	60.00	0.00	0.00	1,645.00	60.00	226.00	226.00	12.00	0.00	
HRA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	289.94	0.00	0.00	0.00	

In [423]:
```
quarter_agg_plotter.plot_tables(services_areas_average_qtr, 'Average', 'darkorange')
```

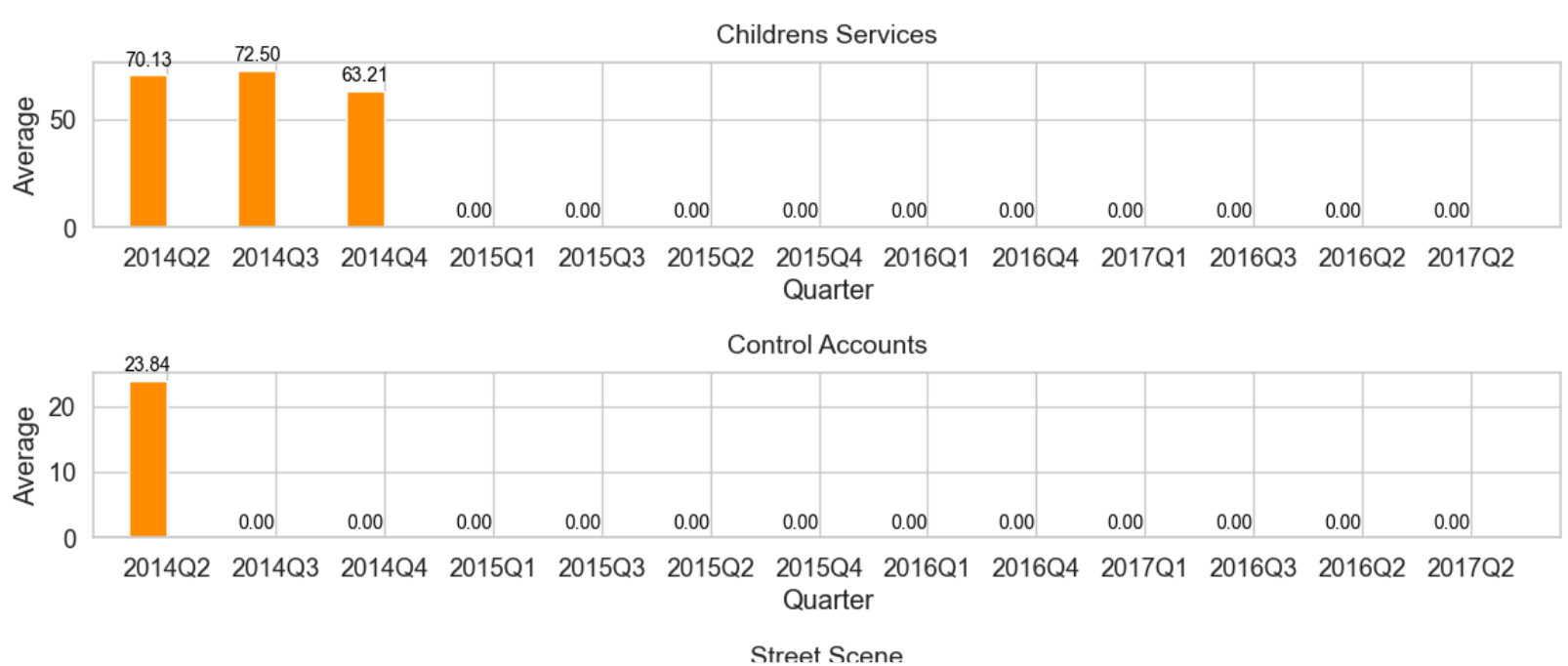


Table by Quarter All Service Areas

In [424]:
```
display(quarter_agg_plotter.summary_table())
```

		Transactions		
		Count	Sum ($)	Average ($)
Service Area	Quarter			
Adults and Communities	2014Q2	16	4,608	288
	2014Q3	8	2,570	321
	2014Q4	18	2,149	119
	2015Q1	11	2,018	183
	2015Q2	8	1,187	148
	2015Q3	15	1,636	109
	2015Q4	19	3,039	160
	2016Q1	23	2,469	107
	2016Q2	25	3,659	146
	2016Q3	37	6.742	182

Significant Changes in Spending Behavior

In [425]:
```
accounts = np.unique(data['Account Description'])
quarters = np.unique(data['Quarter'])
num_accounts = np.unique(data['Account Description']).shape[0]
num_quarters = np.unique(data['Quarter']).shape[0]
num_services_areas = np.unique(data['Service Area']).shape[0]
services_ares = np.unique(data['Service Area'])

print('Number of Accounts:', num_accounts)
print('Number of Quarters:', num_quarters)
```

```
Number of Accounts: 67
Number of Quarters: 13
```

Spikes

Service Areas

In [426]:

```
# calculate account sums by quarter
account_sums_qtr = np.zeros((num_accounts, num_quarters))
for account in range(num_accounts):
    for qtr in range(num_quarters):
        account_sums_qtr[account, qtr] = np.sum(data[(data['Account Description'] == accounts[account]) & \
                                (data['Quarter'] == quarters[qtr])]['Total'])

#dataframe for account sums by quarter

account_sums_qtr_df = pd.DataFrame(account_sums_qtr, columns = quarters, index = accounts)


area_sum_qtr_means = np.array(services_areas_sum_qtr.mean(axis = 1)).reshape(-1, 1)
area_sum_qtr_vars = np.array(services_areas_sum_qtr.var(axis = 1)).reshape(-1, 1)

account_sum_qtr_means = account_sums_qtr.mean(axis = 1).reshape(-1, 1)
account_sum_qtr_vars = account_sums_qtr.var(axis = 1).reshape(-1, 1)

### calculate quarterly spikes in service areas
pd.options.display.float_format = '{:,.2f}'.format
spikes = []

for index in range(num_services_areas):
    for qtr in range(num_quarters):
        z_score = float((services_areas_sum_qtr.iloc[index, qtr] - area_sum_qtr_means[index]) \
                    / np.power(area_sum_qtr_vars[index], 0.5))
        if z_score > 2:
            spikes.append((services_ares[index], quarters[qtr], services_areas_sum_qtr.iloc[index, qtr], z_s

spikes_df = pd.DataFrame(spikes, columns = ['Service Area', 'Quarter', 'Transaction Sum', 'SDs from Average'
display(spikes_df.sort_values(by = 'SDs from Average', ascending = False).reset_index(drop = True))
```

	Service Area	Quarter	Transaction Sum	SDs from Average
0	Assurance	2014Q2	190.71	3.33
1	Customer Support Group	2014Q2	244.00	3.33
2	Governance	2014Q3	2,743.05	3.33
3	Streetscene	2016Q2	289.94	3.33
4	Commissioning	2014Q3	445.50	3.33
5	Children's Education & Skills	2014Q2	13,011.60	3.31
6	Parking & Infrastructure	2016Q2	3,568.25	3.29
7	Street Scene	2015Q4	1,645.00	3.27
8	Adults and Communities	2014Q2	66,339.06	3.12
9	Strategic Commissioning Board	2017Q1	9,510.24	3.04
10	Children's Service DSG	2017Q1	4.55	2.98
11	Commercial	2014Q2	407.20	2.90
12	Deputy Chief Operating Officer	2014Q3	30,283.29	2.84
13	CSG Managed Budget	2014Q4	1,331.55	2.78
14	Family Services	2014Q3	7,858.45	2.77
15	Control Accounts	2014Q2	32,167.34	2.43
16	Children's Family Services	2014Q3	1,728.23	2.20
17	NSCSO	2015Q2	14,608.48	2.15

Accounts

In [427]:

```
### calculate quarterly spikes in accounts

pd.options.display.float_format = '{:,.2f}'.format
spikes = []

for index in range(num_accounts):
    for qtr in range(num_quarters):
        z_score = float((account_sums_qtr[index, qtr] - account_sum_qtr_means[index]) \
                    / np.power(account_sum_qtr_vars[index], 0.5))
        if z_score > 2.0:
            spikes.append((accounts[index], quarters[qtr], account_sums_qtr[index, qtr], z_score))

spikes_df = pd.DataFrame(spikes, columns = ['Account', 'Quarter', 'Transaction Sum', 'SDs from Average'])
display(spikes_df.sort_values(by = 'SDs from Average', ascending = False).reset_index(drop = True))
```

	Account	Quarter	Transaction Sum	SDs from Average
0	Counsels Fees	2016Q1	1,200.00	3.46
1	NNDR Collected	2016Q2	4.10	3.46
2	Software Purchases	2017Q1	288.41	3.46
3	Advertising for staff	2016Q2	450.00	3.46
4	Grant Payments	2015Q3	660.00	3.46
5	Catering Recharge	2016Q1	100.00	3.46
6	Clothing - Uniforms	2014Q3	107.50	3.46
7	Employer's National Insurance	2015Q4	6.67	3.46
8	Furniture-Purchase-Repair	2015Q4	3,984.00	3.46
9	Consultants Fees	2016Q2	900.00	3.46
10	Other Energy	2014Q4	3,889.48	3.46
11	Operating Leases - Transport	2017Q1	585.00	3.44
12	Equipment Hire	2016Q2	1,489.20	3.44
13	Ttl IT & Comms	2016Q4	234.95	3.42
14	Rents	2016Q3	3,506.20	3.39
15	Printing-Contract	2015Q3	216.35	3.37
16	Venue Hire	2016Q2	4,342.20	3.32
17	Electricity	2017Q1	4,096.06	3.30
18	Other Establishments - Third P	2015Q3	397.86	3.03
19	Gas	2016Q2	1,695.22	3.01
20	Education CFR Other Occupation	2015Q4	45.78	2.93
21	CSG - IT	2017Q1	159.90	2.92
22	Software Licences & Support	2014Q2	189.00	2.86
23	Advertising	2015Q3	2,955.60	2.84
24	Pool Transport Charges	2015Q3	510.20	2.80
25	Other Indirect Employee Expens	2015Q4	737.59	2.78

	Account	Quarter	Transaction Sum	SDs from Average
26	Hardware Purchases	2017Q1	258.75	2.76
27	Other Services	2014Q2	21,559.54	2.76
28	Water Services	2017Q1	586.33	2.76
29	E25 - Catering Supplies	2015Q4	1,595.07	2.65
30	Stationery	2014Q3	3,238.89	2.62
31	Non Education Staff GPay	2016Q2	115.80	2.60
32	Grounds maintenance	2014Q4	682.99	2.57
33	Professional Services	2015Q1	3,639.18	2.54
34	Publications	2015Q4	1,564.23	2.52
35	Parking Permit Fees	2016Q3	25.26	2.51
36	Miscellaneous Expenses	2016Q4	14,506.72	2.43
37	Training	2016Q4	10,498.94	2.43
38	Other Vehicle Costs	2016Q2	575.83	2.39
39	Subsistence	2014Q2	2,577.21	2.34
40	Consumable Catering Supplies	2014Q2	1,663.60	2.33
41	Conference Expenses	2016Q1	2,633.50	2.29
42	E19 - Learning Resources	2015Q4	637.85	2.28
43	E19 - Learning Resources	2015Q3	634.15	2.27
44	Education CFR Administrative S	2015Q4	1,698.39	2.27
45	Equipment and Materials Repair	2015Q2	1,499.24	2.26
46	Vehicle Running Costs	2015Q3	11,527.49	2.22
47	Other Vehicle Costs	2014Q3	534.45	2.17
48	Parking Permit Fees	2016Q2	22.10	2.14
49	Subsistence	2014Q4	2,338.77	2.07
50	Building Repairs & Maintenance	2014Q2	3,061.51	2.06
51	Non Education Staff GPay	2016Q4	95.00	2.06
52	Telephones Calls	2014Q3	1,495.47	2.00

Permanent Changes

Service Areas

In [428]:
```
# find permanent changes in spending by service areas
changes = []
for area in range(num_services_areas):
    for qtr in range(1, num_quarters - 1):
        before_mean = services_areas_sum_qtr.iloc[area, :(qtr + 1)].mean()
        after_mean = services_areas_sum_qtr.iloc[area, (qtr + 1):].mean()
        var = services_areas_sum_qtr.iloc[area, 1:].var()
        means_diff = after_mean - before_mean
        z_score = means_diff / np.power(var, 0.5)

        # flag positive changes where long_term averages increase by more than 1 SD
        if (z_score > 1) and (services_areas_sum_qtr.iloc[area, qtr] < services_areas_sum_qtr.iloc[area, (qt
            changes.append((services_ares[area], quarters[qtr + 1], before_mean, after_mean, z_score))

changes_df = pd.DataFrame(changes, columns = ['Service Area', 'After Quarter', 'Old Average', \
                                              'New Average', 'SDs Change'])
changes_df = changes_df[changes_df['Old Average'] > 0]
display(changes_df.reset_index(drop = True))
```

	Service Area	After Quarter	Old Average	New Average	SDs Change
0	Childrens Services	2016Q2	2,459.55	5,752.78	1.09
1	HRA	2015Q1	11,169.57	48,989.82	1.63
2	HRA	2015Q2	18,341.82	50,004.40	1.37
3	Internal Audit & CAFT	2015Q1	1,141.80	5,355.96	1.32
4	Internal Audit & CAFT	2015Q4	2,465.50	6,027.44	1.12
5	NSCSO	2015Q1	692.75	6,447.93	1.33
6	NSCSO	2015Q2	1,906.77	6,547.83	1.07
7	Regional Enterprise	2015Q1	1,791.77	18,200.50	1.56
8	Regional Enterprise	2015Q2	6,069.07	18,122.68	1.14
9	Strategic Commissioning Board	2017Q1	1,014.63	4,755.12	1.40

Accounts

In [429]:
```
# find permanent changes in spending by accounts
changes = []
for account in range(num_accounts):
    for qtr in range(1, num_quarters - 1):
        before_mean = account_sums_qtr[account, :(qtr + 1)].mean()
        after_mean = account_sums_qtr[account, (qtr + 1):].mean()
        var = account_sums_qtr[account, 1:].var()
        means_diff = after_mean - before_mean
        z_score = means_diff / np.power(var, 0.5)

        # flag positive changes where long_term averages increase by more than 1 SD
        if (z_score > 1) and (services_areas_sum_qtr.iloc[area, qtr] < services_areas_sum_qtr.iloc[area, (qt
            changes.append((accounts[account], quarters[qtr + 1], before_mean, after_mean, z_score))

changes_df = pd.DataFrame(changes, columns = ['Account', 'Starting Quarter', 'Old Average', \
                                              'New Average', 'SDs Change'])
changes_df = changes_df[changes_df['New Average'] > 0]
display(changes_df.reset_index(drop = True))
```

	Account	Starting Quarter	Old Average	New Average	SDs Change
0	CSG - IT	2016Q2	0.00	51.48	1.04
1	Gas	2016Q2	0.00	589.42	1.17
2	Laundry and Dry Cleaning	2016Q2	0.00	23.69	1.55
3	Miscellaneous Expenses	2016Q2	3,559.18	7,560.38	1.02
4	Non Education Staff GPay	2016Q2	0.00	42.16	1.07
5	Other Transfer Payments to Soc	2016Q2	4,067.88	8,614.05	1.20
6	Parking Permit Fees	2016Q2	0.00	10.07	1.15
7	Telephone Rentals	2016Q2	58.26	270.65	1.23

Classifying Creditors to Accounts

In [430]:
```
data.head(3)
```

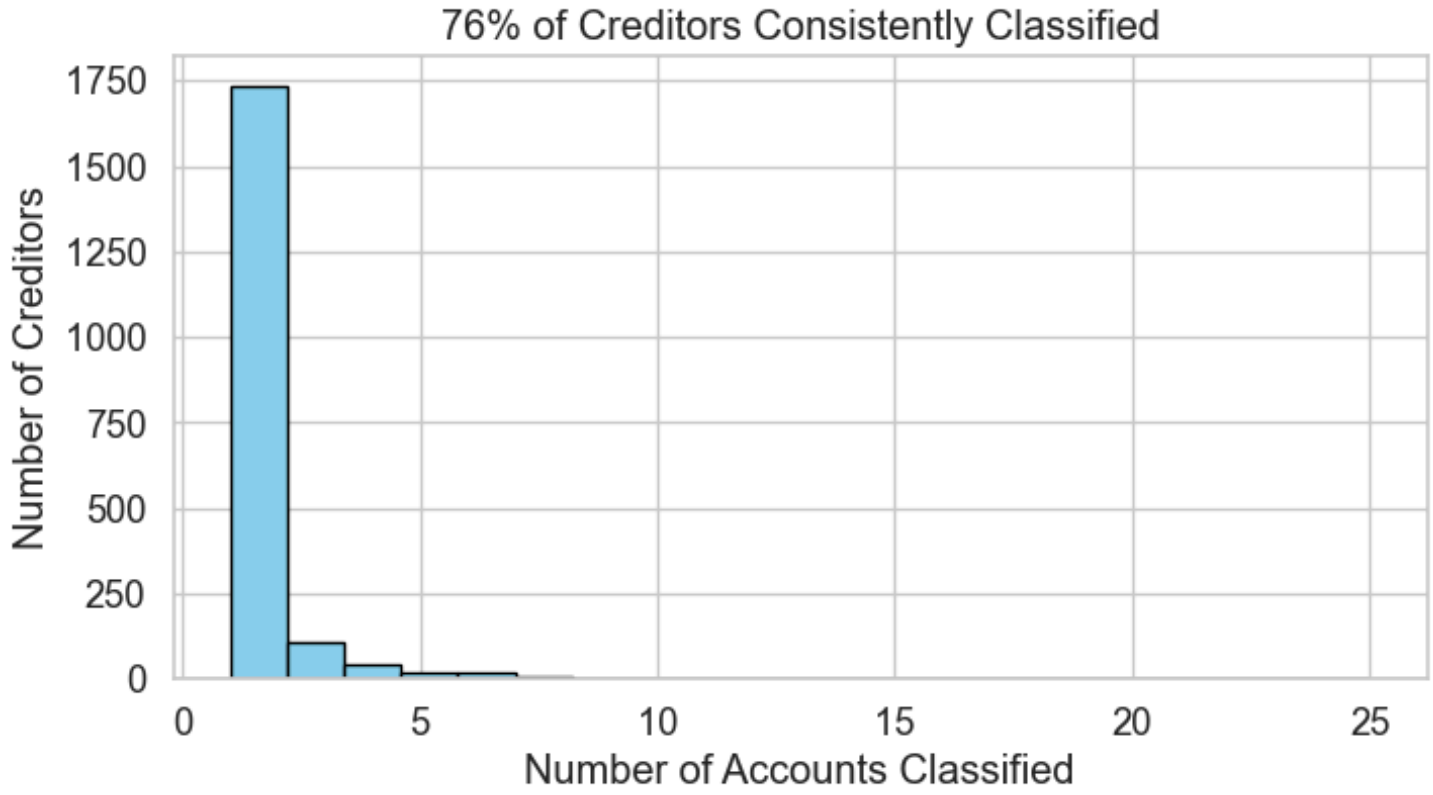
Out[430]:

	Service Area	Account Description	Creditor	Journal Date	JV Reference	Total	Quarter
0	Childrens Services	IT Services	123-REG.CO.UK	2014-04-23	93.00	143.81	2014Q2
1	Childrens Services	Other Services	ACCESS EXPEDITIONS	2014-04-03	111.00	6,000.00	2014Q2
2	Childrens Services	Equipment and Materials Repair	AFE SERVICELINE	2014-04-02	6.00	309.38	2014Q2

In [431]:
```
### find multi-classified creditors

pd.set_option('display.max_colwidth', 1000)

# count accounts per creditor
num_creditor_accounts = []
for creditor in range(num_creditors):
    accounts = np.unique(data[data['Creditor'] == creditors[creditor]]['Account Description'])
    accounts_str = ', '.join(accounts)
    num_creditor_accounts.append((creditors[creditor], len(accounts), accounts_str))

# find multi-classified creditors
accounts_df = pd.DataFrame(num_creditor_accounts, columns = ['Creditor', '# Accounts',
                                                             'Account'])
multi_accounts_df = accounts_df[accounts_df['# Accounts'] > 1]
multi_accounts_df = multi_accounts_df.sort_values(by = '# Accounts', ascending = False)
```

In [432]:

```
# Calculate the percentage of correctly classified creditors
pct_correctly_classed = 100 - int(round(multi_accounts_df.shape[0] / float(accounts_df.shape[0]) * 100))

# Create a histogram plot for the '# Accounts' column in accounts_df
fig, ax = plt.subplots(figsize=(8, 4))
ax.hist(accounts_df['# Accounts'], bins=20, color='skyblue', edgecolor='black')

# Set labels and title for better clarity
ax.set_xlabel("Number of Accounts Classified")
ax.set_ylabel("Number of Creditors")
ax.set_title(f"{pct_correctly_classed}% of Creditors Consistently Classified")

# Display the plot
plt.show()
```



Creditors Classified to More Than One Account top 10

In [433]:
```
display(multi_accounts_df.reset_index(drop = True))
```

	Creditor	# Accounts	Account
0	AMAZON UK MARKETPLACE	25	Advertising, Books-CDs-Audio-Video, Building Repairs & Maintenance, CSG - IT, Cleaning and domestic material, Consumable Catering Supplies, E19 - Learning Resources, E25 - Catering Supplies, Education CFR Administrative S, Education CFR Other Occupation, Equipment and Materials Purcha, Equipment and Materials Repair, Fees and Charges, Food Costs, General Office Expenses, Grounds maintenance, Hardware Purchases, Miscellaneous Expenses, Other Services, Other Transfer Payments to Soc, Publications, Stationery, Training, Travelling Expenses, Ttl IT & Comms
1	AMAZON EU	18	Books-CDs-Audio-Video, Building Repairs & Maintenance, Cleaning and domestic material, E19 - Learning Resources, Education CFR Administrative S, Education CFR Other Occupation, Equipment and Materials Purcha, Equipment and Materials Repair, General Office Expenses, Miscellaneous Expenses, Other Agencies - Third Party P, Other Services, Other Transfer Payments to Soc, Professional Services, Publications, Stationery, Training, Travelling Expenses
2	AMAZON UK RETAIL AMAZO	15	Books-CDs-Audio-Video, Building Repairs & Maintenance, CSG - IT, Cleaning and domestic material, Equipment and Materials Purcha, Food Costs, General Office Expenses, Hardware Purchases, Miscellaneous Expenses, Other Services, Other Transfer Payments to Soc, Publications, Stationery, Training, Ttl IT & Comms
3	AMAZON SVCS EUROPE,SAR	14	Books-CDs-Audio-Video, Building Repairs & Maintenance, Consumable Catering Supplies, E25 - Catering Supplies, Education CFR Administrative S, Equipment and Materials Purcha, General Office Expenses, Hardware Purchases, Miscellaneous Expenses, Other Agencies - Third Party P, Other Services, Other Transfer Payments to Soc, Publications, Stationery
4	ASDA SUPERSTORE	13	Books-CDs-Audio-Video, Building Repairs & Maintenance, Cleaning and domestic material, E19 - Learning Resources, E25 - Catering Supplies, Equipment and Materials Purcha, Food Costs, Other Services, Other Transfer Payments to Soc, Postage, Publications, Stationery, Training
...	...	...	...
469	Lynette Headley-JoneCO-OP GROUP	2	Food Costs, Other Transfer Payments to Soc
470	LUL TICKET OFFICE.	2	Other Transfer Payments to Soc, Travelling Expenses
471	LONDON BOROUGH OF HARR	2	Operating Leases - Transport, Vehicle Running Costs
472	LB BRENT PARKING PCNS	2	Operating Leases - Transport, Vehicle Running Costs
473	ZAHRA NEWSAGENT	2	Books-CDs-Audio-Video, Food Costs

474 rows × 3 columns

Grouping Service Areas

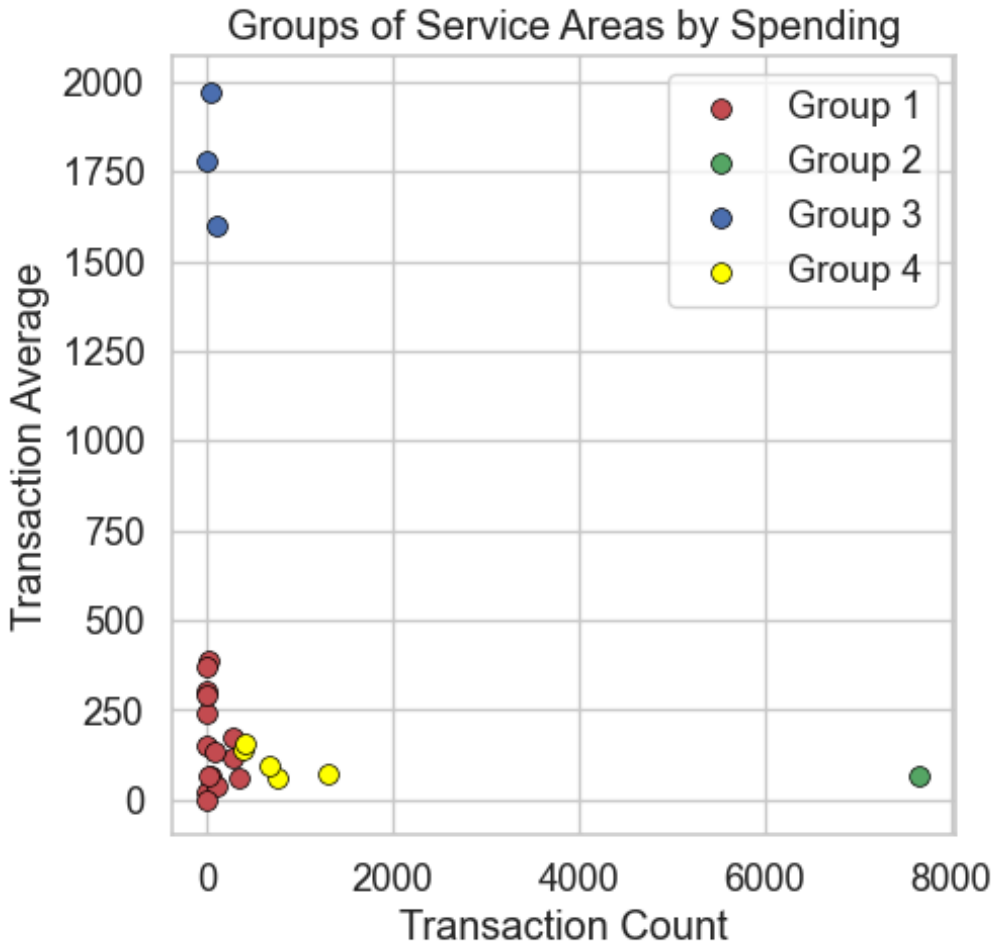
K-means clustering is a popular method for grouping data points into clusters based on their similarity. The algorithm is particularly useful for identifying patterns and trends in data, which is essential for our analysis. We will use the K-means algorithm to group service areas based on their spending behavior. The algorithm will identify clusters of service areas that exhibit similar spending patterns, allowing us to identify groups of service areas that behave similarly.

In [434]:
```
from sklearn.cluster import KMeans

area_counts = np.array(services_areas_counts_qtr.sum(axis = 1)).reshape(-1, 1).astype(int)
area_means = np.array(services_areas_sum_qtr.sum(axis = 1)).reshape(-1, 1).astype(float) / area_counts

## prepare predictors
predictors = np.concatenate((area_counts, area_means), axis = 1)

# create and fit K-means model
n_clusters = 4

model = KMeans(n_clusters = n_clusters)
predict = model.fit_predict(predictors)
cluster_centers = model.cluster_centers_

new_col = np.zeros(data.shape[0]).reshape(-1, 1)
for index in range(num_services_areas):
    area_name = services_ares[index]
    area_indexes = np.where(data['Service Area'] == area_name)
    new_col[area_indexes] = predict[index]

new_col_df = pd.DataFrame(new_col, columns = ['Cluster'])
data_kmeans = pd.concat((data, new_col_df), axis = 1)
```

In [438]:
```
# group clusters with service area names
results = np.concatenate((predict.reshape(-1, 1), services_ares.reshape(-1, 1)), axis = 1)
results_df = pd.DataFrame(results, columns = ['Cluster', 'Service Area'])
clusters = results_df.groupby('Cluster')
```

In [440]:

```
### plot clusters
colors = ['r', 'g', 'b', 'yellow']

fig = plt.figure(figsize = (5, 5))
ax = fig.add_subplot(1, 1, 1)

for cluster in range(n_clusters):
    cluster_counts = area_counts[predict == cluster]
    cluster_means = area_means[predict == cluster]
    ax.scatter(cluster_counts, cluster_means, color = colors[cluster], s = 50,
               edgecolor = 'black', linewidth = 0.5, label = 'Group ' + str(cluster + 1))

ax.set_xlabel("Transaction Count")
ax.set_ylabel("Transaction Average")
ax.set_title("Groups of Service Areas by Spending")
ax.legend(loc = 'best')

plt.show()
```


Groups of Service Areas by Spending
Transaction Average
2000
1750
1500
1250
1000
750
500
250
0
0
2000
4000
6000
8000
Transaction Count
Group 1
Group 2
Group 3
Group 4

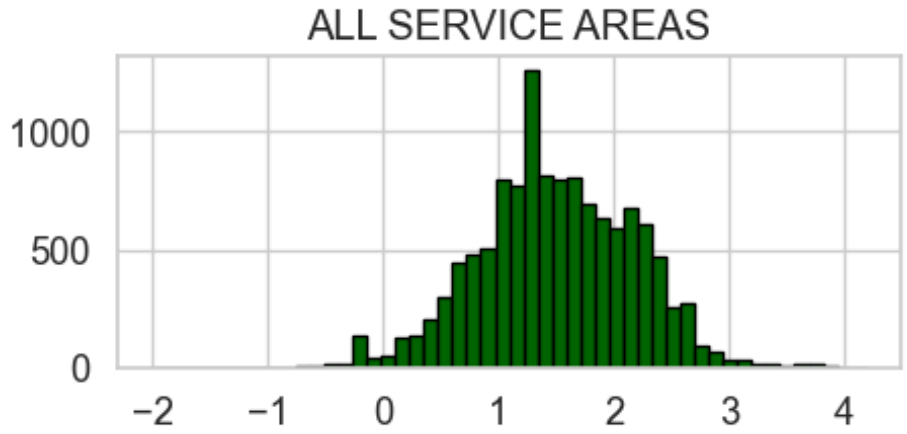
Anomaly Detection

In [447]:
```
# find service areas with < 5 transactions in this time period
min_transactions = 5

many_xact_areas = []
few_xact_areas = []

for index in range(num_services_areas):
    area_name = services_ares[index]
    num_xacts = np.count_nonzero(data[data['Service Area'] == area_name]['Total'])
    if num_xacts >= min_transactions:
        many_xact_areas.append((area_name, num_xacts))
    else:
        few_xact_areas.append((area_name, num_xacts))

many_xact_areas_df = pd.DataFrame(many_xact_areas, columns = \
                                  ['Service Area', 'Number of Transactions'])

few_xact_areas_df = pd.DataFrame(few_xact_areas, columns = \
                                 ['Service Area', 'Number of Transactions'])
display(few_xact_areas_df.reset_index(drop = True))
```

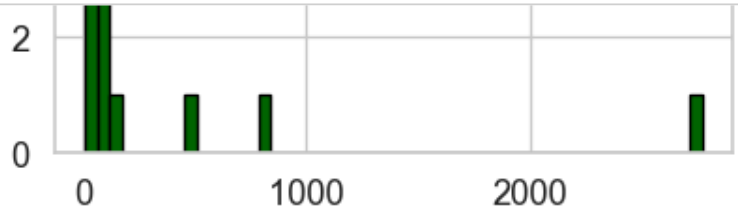
	Service Area	Number of Transactions
0	HRA	1
1	NSCSO	3
2	Public Health	3
3	Strategic Commissioning Board	1

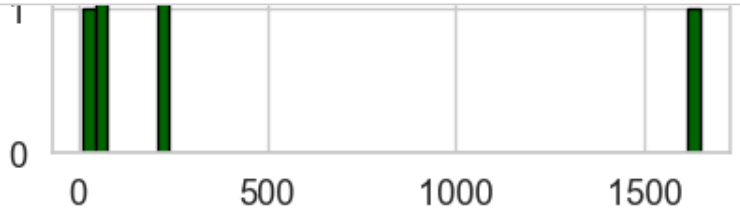
Frequencies of transaction amounts

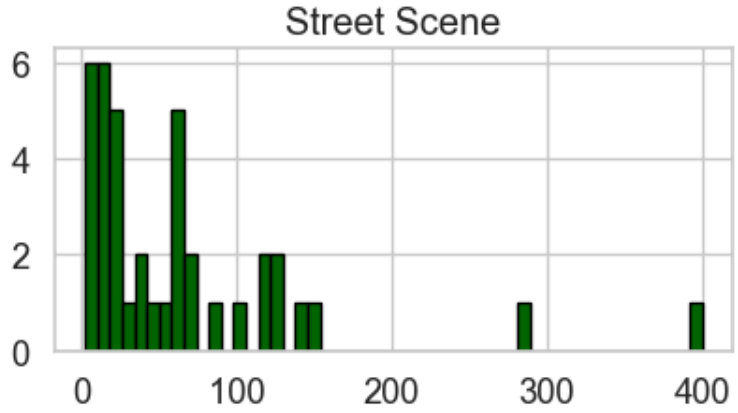
In [448]:
```
fig, ax = plt.subplots(1, 1, figsize = (5, 2))
ax.hist(np.log10(data[data['Total'] > 0]['Total']), bins = 50, color = 'darkgreen', edgecolor = 'black')
ax.set_title("All Service Areas".upper())
plt.show()
```

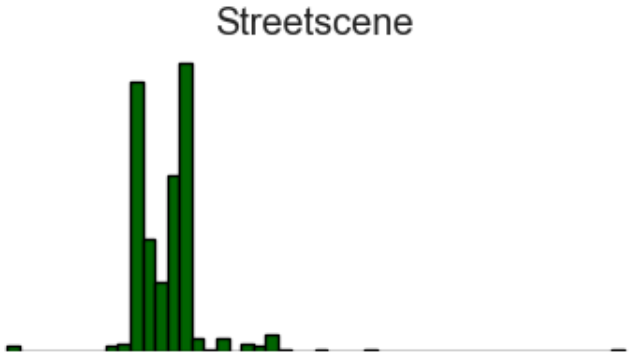


In [457]:
```
num_areas = many_xact_areas_df.shape[0]
fig, ax_ar = plt.subplots((num_areas // 2) + (num_areas % 2), 2, figsize = (9, 24))

for index in range(num_areas):
    area = many_xact_areas_df.iloc[index, 0]
    area_data = data[data['Service Area'] == area]['Total']

    ax_ar[index // 2, index % 2].hist(area_data, bins = 50, color = 'darkgreen', edgecolor = 'black')
    ax_ar[index // 2, index % 2].set_title(area)

ax_ar[-1, -1].axis('off')
plt.tight_layout()
plt.show()
```









Transactions that are anomalous for high transaction amount

In [460]:
```
### calculate percentiles of transaction amounts for each many-transaction service area

avg_col = np.zeros(num_areas).reshape(-1, 1)
var_col = np.zeros(num_areas).reshape(-1, 1)
percentile_col = np.zeros(data.shape[0])

for index in range(num_areas):
    area_name = many_xact_areas_df.iloc[index, 0]
    area_indexes = np.where(data['Service Area'] == area_name)
    area_totals = data.iloc[area_indexes]['Total'].values

    avg_col[index] = area_totals.mean()
    var_col[index] = area_totals.var()

    dist = norm(avg_col[index], np.power(var_col[index], 0.5))
    percentile_col[area_indexes] = dist.cdf(data.iloc[area_indexes]['Total'].values)

avg_col_df = pd.DataFrame(avg_col, columns = ['Average'])
var_col_df = pd.DataFrame(var_col, columns = ['Variance'])
stat_cols_df = pd.concat((avg_col_df, var_col_df), axis = 1)
many_xact_areas_df = pd.concat((many_xact_areas_df, stat_cols_df), axis = 1)

percentile_col_df = pd.DataFrame(percentile_col.reshape(-1, 1), columns = ['Percentile'])
data = pd.concat((data, percentile_col_df), axis = 1)
```

In [470]:
```
pd.options.display.float_format = '{:,.2f}'.format

anomaly_threshold = 0.98
anomalous_data = data[data['Percentile'] > anomaly_threshold]


for index in range(num_areas):
    area_name = many_xact_areas_df.iloc[index, 0]

    area_data = anomalous_data[anomalous_data['Service Area'] == area_name]
    if (area_data.shape[0] >= min_transactions):
        area_data = area_data.sort_values(by = 'Percentile', ascending = False)
        caveat_str = ""
    else:
        extra_transactions = min_transactions - area_data.shape[0]
        caveat_str = "  (" + str(extra_transactions) + \
                     " extra transactions listed to meet analysis requests)"

        area_data = data[data['Service Area'] == area_name].sort_values(by = ['Percentile', 'Journal Date'],
                                                                        ascending = False)
        area_data = area_data.iloc[:min_transactions, :]

    display(area_data.head(min_transactions))



#Table with anomalous data just 5 transactions per service area
```

	Service Area	Account Description	Creditor	Journal Date	JV Reference	Total	Quarter	Percentile
8013	Adults and Communities	CSG - IT	AMAZON UK RETAIL AMAZO	2017-02-01	10,975.00	62.40	2017Q1	1.00
8064	Adults and Communities	Postage	WWW.ROYALMAIL.COM	2016-06-29	8,573.00	128.00	2016Q2	1.00
8046	Adults and Communities	Other Agencies - Third Party P	HOLIDAY INNS	2016-12-09	10,578.00	830.00	2016Q4	1.00
8047	Adults and Communities	Other Agencies - Third Party P	HOLIDAY INNS	2016-11-22	10,349.00	1,670.30	2016Q4	1.00
5689	Adults and Communities	Publications	GUARDIAN NEWS & MEDIA	2015-12-04	6,273.00	1,391.04	2015Q4	1.00

	Service Area	Account Description	Creditor	Journal Date	JV Reference	Total	Quarter	Percentile
8368	Assurance	Vehicle Running Costs	TFL ROAD CHARGE	2017-01-30	10,837.00	11.50	2017Q1	1.00

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2