Visual Data Analysis of Fraudulent Transactions

Your CFO has also requested detailed trends data on specific card holders. Use the starter notebook to query your database and generate visualizations that supply the requested information as follows, then add your visualizations and observations to your markdown report.

In [ ]:
```
# Initial imports
import pandas as pd
import calendar
import hvplot.pandas
from sqlalchemy import create_engine
%matplotlib inline
```

In [ ]:
```
# Create a connection to the database

def postgres_connect_str(
    uname    = 'Daniel',
    pass_env = 'POSTGRES_PASSWORD',
    host     = 'pg-2e8191e-instructors-1f45.aivencloud.com',
    database = 'daniel',
    port     = 18645,
):
    from dotenv import load_dotenv
    import os

    load_dotenv()
    password = os.getenv(pass_env)
    return f"postgresql://{uname}:{password}@{host}:{port}/{database}?sslmode=require"

postgres_connect_str()

engine = create_engine(postgres_connect_str())
```

Data Analysis Question 1

The two most important customers of the firm may have been hacked. Verify if there are any fraudulent transactions in their history. For privacy reasons, you only know that their cardholder IDs are 2 and 18.

- Using hvPlot, create a line plot representing the time series of transactions over the course of the year for each cardholder separately.

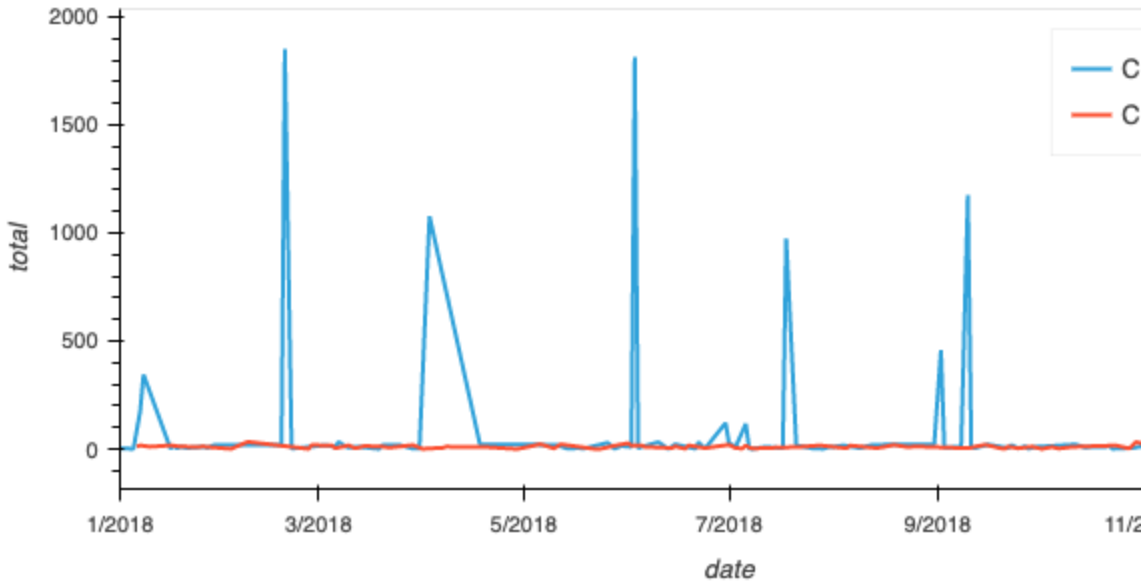
- Next, to better compare their patterns, create a single line plot that containins both card holders' trend data.

- What difference do you observe between the consumption patterns? Does the difference suggest a fraudulent transaction? Explain your rationale in the markdown report.

In [ ]:

```
# Write function that locates outliers using standard deviation
sql=f'''SELECT card_holder_id, date_trunc('day',date) as date, SUM(amount) as total FROM "Transactions" t
        INNER JOIN "Credit_card" c
        ON t."card_number" = c."cardNumber"
        WHERE card_holder_id = 2 or card_holder_id = 18
         GROUP BY date_trunc('day',date), card_holder_id
        ORDER BY date_trunc('day',date)

'''
df=pd.read_sql_query(sql, engine)

df.index = pd.to_datetime(df["date"])

plot1 = df[
        df["card_holder_id"] == 18 # where card holder id is 18
][["total", "date"]].hvplot(label="Customer id 18")

plot2 = df[
        df["card_holder_id"] == 2 # where card holder id is 18
][["total", "date"]].hvplot(label="Customer id 2") # select all but card_holder_id

plot1 * plot2
```

Out[ ]:



In [ ]:
```
std2= df['total'].std()*2
len(df[df['total'] >= std2])
```

Out[ ]:
```
7
```

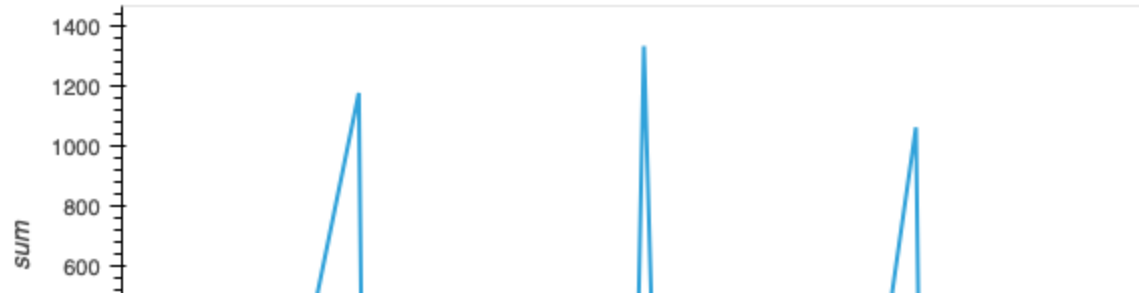
Data Analysis Question 2

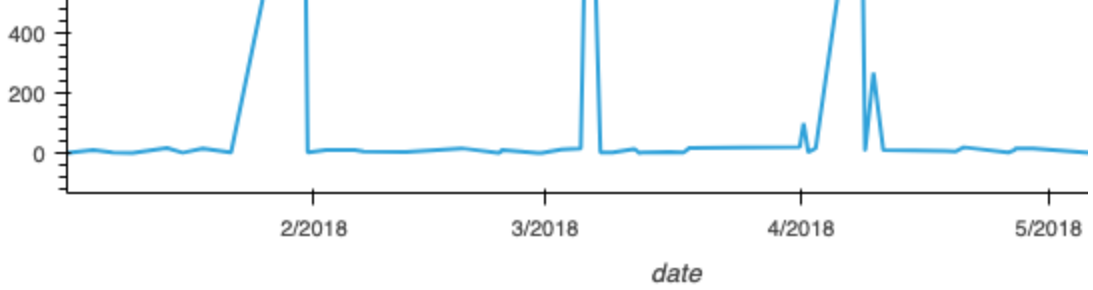
The CEO of the biggest customer of the firm suspects that someone has used her corporate credit card without authorization in the first quarter of 2018 to pay quite expensive restaurant bills. Again, for privacy reasons, you know only that the cardholder ID in question is 25.

- Using Plotly Express, create a box plot, representing the expenditure data from January 2018 to June 2018 for cardholder ID 25.
- Are there any outliers for cardholder ID 25? How many outliers are there per month?
- Do you notice any anomalies? Describe your observations and conclusions in your markdown report.

In [ ]:
```
# loading data of daily transactions from jan to jun 2018 for card holder 25
# Write the query
query = '''
        SELECT date, SUM(amount)
        FROM "Transactions" t
        INNER JOIN "Credit_card" c
        ON t."card_number" = c."cardNumber"
        WHERE card_holder_id = 25 AND date BETWEEN '01/01/2018' AND '06/01/2018'
        GROUP BY date
        '''
# Create a DataFrame from the query result. HINT: Use pd.read_sql(query, engine)
df = pd.read_sql(query, engine)

df.index = pd.to_datetime(df.date)

df.hvplot()
```

Out[ ]:





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

std2up = df.std()*2 + df.mean()
std2low= df.std()*2 - df.mean()
```

```
/Users/dan/opt/anaconda3/envs/dev3/lib/python3.7/site-packages/ipykernel_launcher.py:3: FutureWarning: DataFrame.mean and
DataFrame.median with numeric_only=None will include datetime64 and datetime64tz columns in a future version.
  This is separate from the ipykernel package so we can avoid doing imports until
/Users/dan/opt/anaconda3/envs/dev3/lib/python3.7/site-packages/ipykernel_launcher.py:4: FutureWarning: DataFrame.mean and
DataFrame.median with numeric_only=None will include datetime64 and datetime64tz columns in a future version.
  after removing the cwd from sys.path.
```

In [ ]:
```
df.groupby([df.index.month, df.index.day]).sum()
```

Out[ ]:

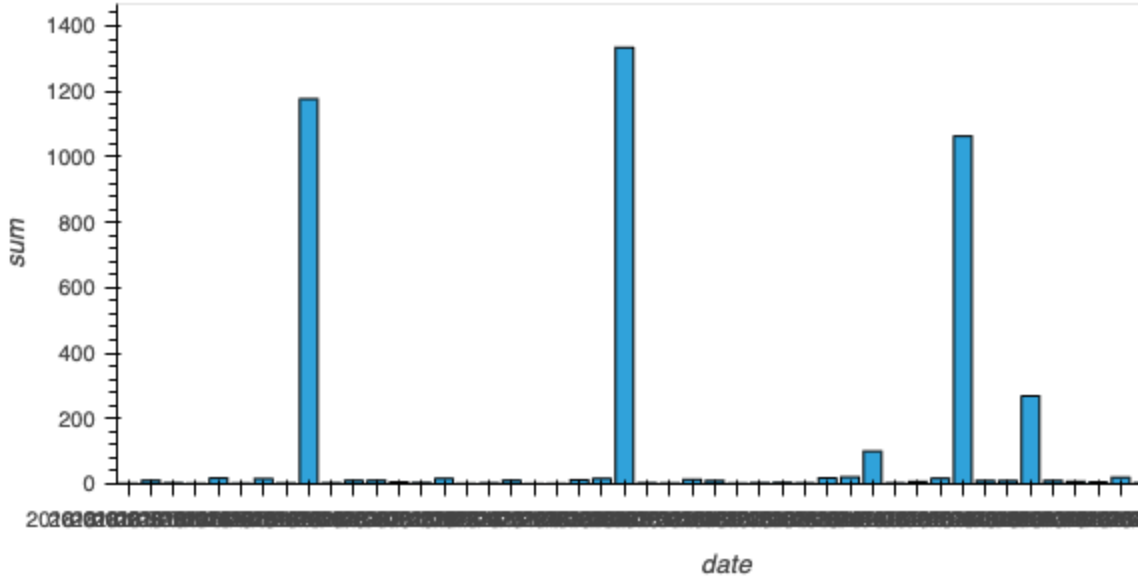
		sum
date	date	
1	2	1.46
	5	10.74
	7	2.93
	10	1.39
	14	17.84
	16	1.65
	18	15.86
	21	2.22
	30	1177.00
	31	2.75
2	2	10.75
	5	10.81

		sum
date	**date**	
	7	5.97
	12	3.69
	18	16.70
	23	14.90
	28	2.09
3	**2**	12.42
	5	16.58
	6	1334.00
	7	2.88
	9	2.04
	11	13.57
	12	14.83
	16	4.20
	17	2.56
	18	18.28
	31	21.04
4	**1**	102.62
	2	24.23
	8	1083.21
	9	269.00
	10	10.24
	18	7.39
	19	6.01
	20	20.03
	26	28.47
	29	16.50
5	**6**	1.10
	13	1046.00

		sum
date	**date**	
	17	12.15
	19	2.27
	29	5.97

In [ ]:
```
monthly_df = df.copy()
#monthly_df.rename(columns={"sum":"monthlySum"}, inplace=True)
monthly_df.groupby([monthly_df.index.month, monthly_df.index.day]).sum()

monthly_df["Month"] = monthly_df["date"].apply(lambda x: calendar.month_name[x.month])
monthly_df["Day"] = monthly_df["date"].apply(lambda x: x.day)

monthly_df[["sum", "Month", "Day"]].drop(columns=["Day"]).hvplot.bar()
```

Out[ ]:



In [ ]:
```
# loop to change the numeric month to month names
```

In [ ]:
```

```

In [ ]:
```
# Creating the six box plots using plotly express
```