visual_data_analysis

February 10, 2021

1 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.

[46]:
```
# Initial imports
import pandas as pd
import calendar
import plotly.express as px
import hvplot.pandas
from sqlalchemy import create_engine
from dotenv import load_dotenv
import os
import datetime
import calendar
```

[12]:
```
load_dotenv()
pgpw = os.getenv("pgpw")
```

[13]:
```
# Create a connection to the database
engine = create_engine(f"postgresql://postgres:{pgpw}@localhost:5432/
 ↪fraud_detection")
```

1.1 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.

[14]:
```
# loading data for card holder 2 and 18 from the database
# Write the query
query = "SELECT credit_card.cardholder_id, transaction.date, transaction.amount␣
 ↪ FROM transaction \
INNER JOIN credit_card ON transaction.card = credit_card.card \
WHERE credit_card.cardholder_id IN ('2','18');"
# Create a DataFrame from the query result. HINT: Use pd.read_sql(query, engine)
df = pd.read_sql(query, engine)
df.head()
```

[14]:

	cardholder_id	date	amount
0	18	2018-01-01 23:15:10	2.95
1	18	2018-01-05 07:19:27	1.36
2	2	2018-01-06 02:16:41	1.33
3	2	2018-01-06 05:13:20	10.82
4	18	2018-01-07 01:10:54	175.00

[26]:
```
# Plot for cardholder 2
df_cardholder2 = df.loc[df["cardholder_id"]==2].set_index("date").sort_index()
df_cardholder2["amount"].hvplot.line(rot=90,title="Transaction Amount by␣
 ↪Month",width=800,height=400)
```

[26]:
```
:Curve   [date]   (amount)
```

[27]:
```
# Plot for cardholder 18
df_cardholder18 = df.loc[df["cardholder_id"]==18].set_index("date").sort_index()
df_cardholder18["amount"].hvplot.line(rot=90,title="Transaction Amount by␣
 ↪Month",width=800,height=400)
```

[27]:
```
:Curve   [date]   (amount)
```

[28]:
```
# Combined plot for card holders 2 and 18
df_cardholder2["amount"].hvplot.line(rot=90,title="Transaction Amount by␣
 ↪Month",width=800,height=400) * df_cardholder18["amount"].hvplot.
 ↪line(rot=90,title="Transaction Amount by Month",width=800,height=400)
```

[28]:
```
:Overlay
   .Curve.Amount.I   :Curve   [date]   (amount)
   .Curve.Amount.II  :Curve   [date]   (amount)
```

1.2 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.

[49]:
```
# loading data of daily transactions from jan to jun 2018 for card holder 25
# Write the query
query2 = "SELECT transaction.date, SUM(transaction.amount) FROM transaction \
INNER JOIN credit_card ON transaction.card = credit_card.card \
WHERE credit_card.cardholder_id = '25' AND transaction.date >= '01/01/2018' AND␣
 ↪transaction.date <= '06/30/2018' \
GROUP BY transaction.date \
ORDER BY transaction.date;"
# Create a DataFrame from the query result. HINT: Use pd.read_sql(query, engine)
df2 = pd.read_sql(query2, engine)
df2["date"] = pd.to_datetime(df2["date"])
df2["month"] = pd.DatetimeIndex(df2["date"]).month
df2["day"] = pd.DatetimeIndex(df2["date"]).day
df2["month_name"] = df2["month"].apply(lambda x: calendar.month_name[x])
df2.rename(columns={"sum":"amount"},inplace=True)
df_cardholder25 = df2.
 ↪drop(columns=["date","month_name"])[["month","day","amount"]]
df_cardholder25.head()
```

[49]:

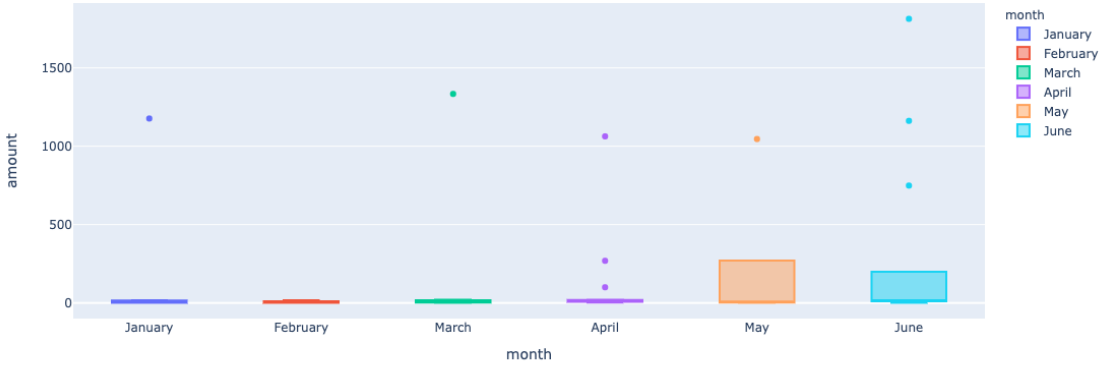
	month	day	amount
0	1	2	1.46
1	1	5	10.74
2	1	7	2.93
3	1	10	1.39
4	1	14	17.84

[52]:
```
# loop to change the numeric month to month names
df_cardholder25_2 = df2.drop(columns=["date","month"]).
 ↪rename(columns={"month_name":"month"})[["month","day","amount"]]
df_cardholder25_2.head()
```

[52]:

	month	day	amount
0	January	2	1.46
1	January	5	10.74
2	January	7	2.93
3	January	10	1.39
4	January	14	17.84

[57]:
```
# Creating the six box plots using plotly express
px.box(df_cardholder25_2, y="amount", x="month", color="month")
```


amount
1500
1000
500
0
January
February
March
April
May
June
month
month
January
February
March
April
May
June

[ ]: