Challenge

Another approach to identifying fraudulent transactions is to look for outliers in the data. Standard deviation or quartiles are often used to detect outliers. Using this starter notebook, code two Python functions:

- One that uses standard deviation to identify anomalies for any cardholder.
- Another that uses interquartile range to identify anomalies for any cardholder.

Identifying Outliers using Standard Deviation

In [1]:
```
# Initial imports
import pandas as pd
import numpy as np
import random
from sqlalchemy import create_engine
```

In [2]:
```
# Create a connection to the database
engine = create_engine("postgresql://postgres:postgres@localhost:5432/fraud_detection")
```

In [3]:
```
# Write function that locates outliers using standard deviation
def find_outliers_sd(card_holder=1):
    query = (
        "SELECT t.date, t.amount, t.card "
        + "FROM transaction AS t "
        + "JOIN credit_card AS cc ON cc.card = t.card "
        + "JOIN card_holder AS ch ON ch.id = cc.id_card_holder "
        + "WHERE ch.id = "
        + str(card_holder)
        + " ORDER BY date"
    )
    data = pd.read_sql(query, engine)
    elements = data["amount"]
    mean = np.mean(elements, axis=0)
    sd = np.std(elements, axis=0)
    # 2 standard deviations are taken for analysis purposes
    low_transactions = [x for x in elements if (x < mean - 2 * sd)]
    high_transaction = [x for x in elements if (x > mean + 2 * sd)]
    final_list = low_transactions + high_transaction
    if len(final_list) > 0:
        query = (
            "SELECT t.date, t.amount, t.card "
            + "FROM transaction AS t "
            + "JOIN credit_card AS cc ON cc.card = t.card "
            + "JOIN card_holder AS ch ON ch.id = cc.id_card_holder "
            + "WHERE ch.id = "
            + str(card_holder)
            + " AND t.amount IN ("
            + str(final_list)[1:-1]
            + ") "
            + "ORDER BY date"
        )
        data = pd.read_sql(query, engine)
        return data
    else:
        return "There are no fraudulent transactions identified for this card holder"
```

In [4]:
```
# Find anomalous transactions for 3 random card holders
for i in range(1, 4):
    card_holder = random.randint(1, 25)
    print("*" * 60)
    print(f"Looking for fraudulent transactions for card holder id {card_holder}")
    print(find_outliers_sd(card_holder))
```

```
************************************************************
Looking for fraudulent transactions for card holder id 23
                 date  amount              card
0 2018-06-21 22:11:26   20.65  4150721559116778
************************************************************
Looking for fraudulent transactions for card holder id 20
                 date  amount                card
0 2018-01-14 06:19:11   21.11     3535651398328201
1 2018-05-11 12:43:50   20.56  4586962917519654607
2 2018-08-26 07:15:18   23.13     4506405265172173
3 2018-10-07 08:16:54   20.44  4586962917519654607
4 2018-11-09 19:38:36   20.27     3535651398328201
************************************************************
Looking for fraudulent transactions for card holder id 24
                 date  amount            card
0 2018-03-20 13:05:54  1011.0  30142966699187
1 2018-04-21 18:40:47   525.0  30142966699187
2 2018-05-08 13:21:01  1901.0  30142966699187
3 2018-12-21 09:56:32  1301.0  30142966699187
4 2018-12-25 19:10:42  1035.0  30142966699187
```

Identifying Outliers Using Interquartile Range

In [5]:
```
# Write a function that locates outliers using interquartile range
def find_outliers_iqr(card_holder=1):
    query = (
        "SELECT t.date, t.amount, t.card "
        + "FROM transaction AS t "
        + "JOIN credit_card AS cc ON cc.card = t.card "
        + "JOIN card_holder AS ch ON ch.id = cc.id_card_holder "
        + "WHERE ch.id = "
        + str(card_holder)
        + " ORDER BY date"
    )
    data = pd.read_sql(query, engine)
    # calculate interquartile range
    q25, q75 = np.percentile(data["amount"], 25), np.percentile(data["amount"], 75)
    iqr = q75 - q25
    # calculate the outlier cutoff
    cut_off = iqr * 1.5
    lower, upper = q25 - cut_off, q75 + cut_off
    # identify outliers
    outliers = [x for x in data["amount"] if x < lower or x > upper]
    if len(outliers) > 0:
        query = (
            "SELECT t.date, t.amount, t.card "
            + "FROM transaction AS t "
            + "JOIN credit_card AS cc ON cc.card = t.card "
            + "JOIN card_holder AS ch ON ch.id = cc.id_card_holder "
            + "WHERE ch.id = "
            + str(card_holder)
            + " AND t.amount IN ("
            + str(outliers)[1:-1]
            + ") "
            + "ORDER BY date"
        )
        data = pd.read_sql(query, engine)
        return data
    else:
        return "There are no fraudulent transactions identified for this card holder"
```

In [6]:
```
# Find anomalous transactions for 3 random card holders
for i in range(1, 4):
    card_holder = random.randint(1, 25)
    print("*" * 60)
    print(f"Looking for fraudulent transactions for card holder id {card_holder}")
    print(find_outliers_iqr(card_holder))
```

```
************************************************************
Looking for fraudulent transactions for card holder id 21
There are no fraudulent transactions identified for this card holder
************************************************************
Looking for fraudulent transactions for card holder id 11
There are no fraudulent transactions identified for this card holder
************************************************************
Looking for fraudulent transactions for card holder id 8
There are no fraudulent transactions identified for this card holder
```

In [ ]: