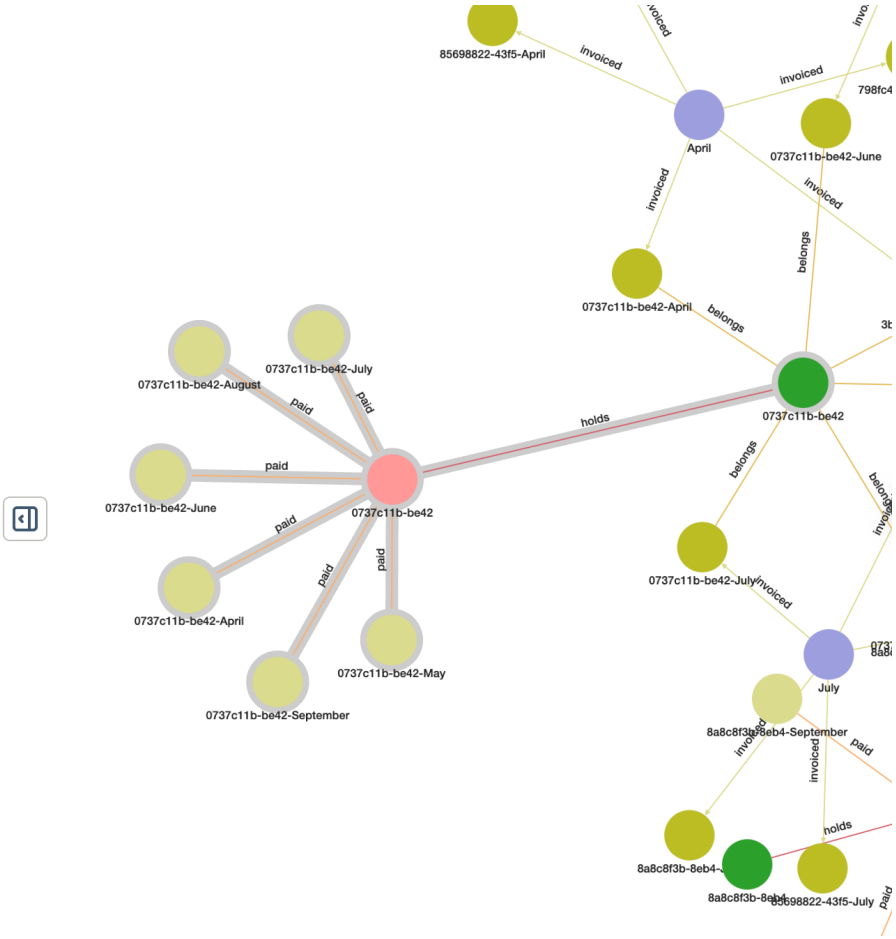
Vertex	Account
(PRIMARY ID) id	0737c11b-be42
limit_bal	50000
status	false



One instance of Bills paid each month from one account held by one customer

```
import pandas as pd
import pyTigerGraph as tg

host = "https://3314d527106244578c3eff59e7a1ce42.i.tgcloud.io"
graphname = "MLCC_Lab"
username = "user_2"
password = "Tb1Yb8Kc6Vt6Jf3_"
secret = "s800no94cutspdqlaae55qfurvr7hsf1"

conn = tg.TigerGraphConnection(host=host, graphname=graphname,
username=username, password=password)
conn.apiToken = conn.getToken(secret)
print("TOKEN: ", conn.apiToken)

data = pd.read_csv("/Users/gideoncrawley/MSBA/MSBA/Machine Learning
5505/Chapter_1_cleaned_data.csv")
data = data.head()
```

```
def create_month_vertex(month):
    month_id = f"{month}"
    attributes = {
        "name": f"{month}"
    }
    conn.upsertVertex("Month", month_id, attributes)
    return(month_id)

def create_account_vertex(account_id, limit_bal, status):
    account_id = f"{account_id}"
    attributes = {
        "limit_bal": limit_bal,
        "status" : status
    }
    conn.upsertVertex("Account", account_id, attributes)
    return(account_id)

def create_customer_vertex(customer_id, sex, age):
    customer_id = f"{customer_id}"
    attributes = {
        "sex": "male" if sex == 1 else "female",
        "age": age,
    }
    conn.upsertVertex("Customer", customer_id, attributes)
    return(customer_id)

def create_billing_vertex(bill_id, bill_amt):
    billing_id = f"{bill_id}"
    attributes = {
        "bill_amt": bill_amt
    }
    conn.upsertVertex("Billing", billing_id, attributes)
    return(billing_id)

def create_payment_vertex(payment_id, pay_amt):
    payment_id = f"{payment_id}"
    attributes = {
        "pay_amt": pay_amt
    }
    conn.upsertVertex("Payment", payment_id, attributes)
    return(payment_id)

def populate_month(month):
    month_id = month
    name = month
    month_id = create_month_vertex(month)
    return(month_id)

def populate_billing(data, month):
    billing_id = f"{data['ID']}-{month}"
    if month == "April":
        bill_amt = int(data['BILL_AMT6'])
    elif month == "May":
```

```
        bill_amt = int(data['BILL_AMT5'])
    elif month == "June":
        bill_amt = int(data['BILL_AMT4'])
    elif month == "July":
        bill_amt = int(data['BILL_AMT3'])
    elif month == "August":
        bill_amt = int(data['BILL_AMT2'])
    elif month == "September":
        bill_amt = int(data['BILL_AMT1'])

    bill_id = create_billing_vertex(billing_id, bill_amt)
    conn.upsertEdge("Month", f"{month}", "invoiced", "Billing",
f"{billing_id}")
    return(billing_id)

def populate_customer(data):
    customer_id = f"{data['ID']}"
    sex = f"{data['SEX']}"
    age = int(data['AGE'])
    customer_id = create_customer_vertex(customer_id, sex, age)
    return(customer_id)

def populate_account(data):
    account_id = f"{data['ID']}"
    limit_bal = int(data['LIMIT_BAL'])
    status = bool(data['default payment next month'])
    account_id = create_account_vertex(account_id, limit_bal, status)
    return(account_id)

def populate_payment(data, month):
    payment_id = f"{data['ID']}-{month}"
    if month == "April":
        pay_amt = int(data['PAY_AMT6'])
    elif month == "May":
        pay_amt = int(data['PAY_AMT5'])
    elif month == "June":
        pay_amt = int(data['PAY_AMT4'])
    elif month == "July":
        pay_amt = int(data['PAY_AMT3'])
    elif month == "August":
        pay_amt = int(data['PAY_AMT2'])
    elif month == "September":
        pay_amt = int(data['PAY_AMT1'])
    payment_id = create_payment_vertex(payment_id, pay_amt)
    return(payment_id)

def populate(data):
    month_list = ['April', 'May', 'June', 'July', 'August', 'September']
    for month in month_list:
        month = populate_month(month)
        for index, row in data.iterrows():
            billing_id = populate_billing(row, month)
            customer_id = populate_customer(row)
            account_id = populate_account(row)
```

```
            payment_id = populate_payment(row, month)
            conn.upsertEdge("Billing", f"{billing_id}",
"billed","Account", f"{account_id}")
            conn.upsertEdge("Customer", f"{customer_id}", "holds",
"Account", f"{account_id}")
            conn.upsertEdge( "Account", f"{account_id}", "paid",
"Payment", f"{payment_id}")
            conn.upsertEdge("Customer", f"{customer_id}", "belongs",
"Billing", f"{billing_id}")


populate(data)
```