TERM PROJECT - ANTHONY ASILO - DATA VISUALIZATION – DUE 4/28/20 11:59PM

a. SOURCE CODE

```
import queue
import statistics
from math import *
from statistics import *
import pandas as pd
import numpy as np
import time
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns
import os
import re
import datetime

#Main Method That Provides Data Calculations and Visualizations
def Main():
   print("Authenticating...")
   time.sleep(1)

   # dataset
   ds = pd.read_csv('/Users/anthonyasilo/Desktop/Data_Visualization/TermP/SUNTRUST_HISTORY/mySuntrustHistory.csv')

   #Description of DataFrame
   print('\n\nDescription of DataFrame:')
   ds.describe()

   #Food Purchase by Date
   print('\n\nFood Purchase by Date:\n')
   print(ds.loc[ds["Type"].isin(["Food"])])

   #Charge Category Value
   print('\n\nCharge Category Value:\n')
   print(ds["Type"].value_counts())

   ds4 = pd.DataFrame(ds["Type"].value_counts())
   print(ds4)
   figa = px.bar(x=ds4.index.values, y=ds4['Type'].values, title="Amount of Items Purchases per Category", labels={'x':'Category', 'y':'Frequency '})
```

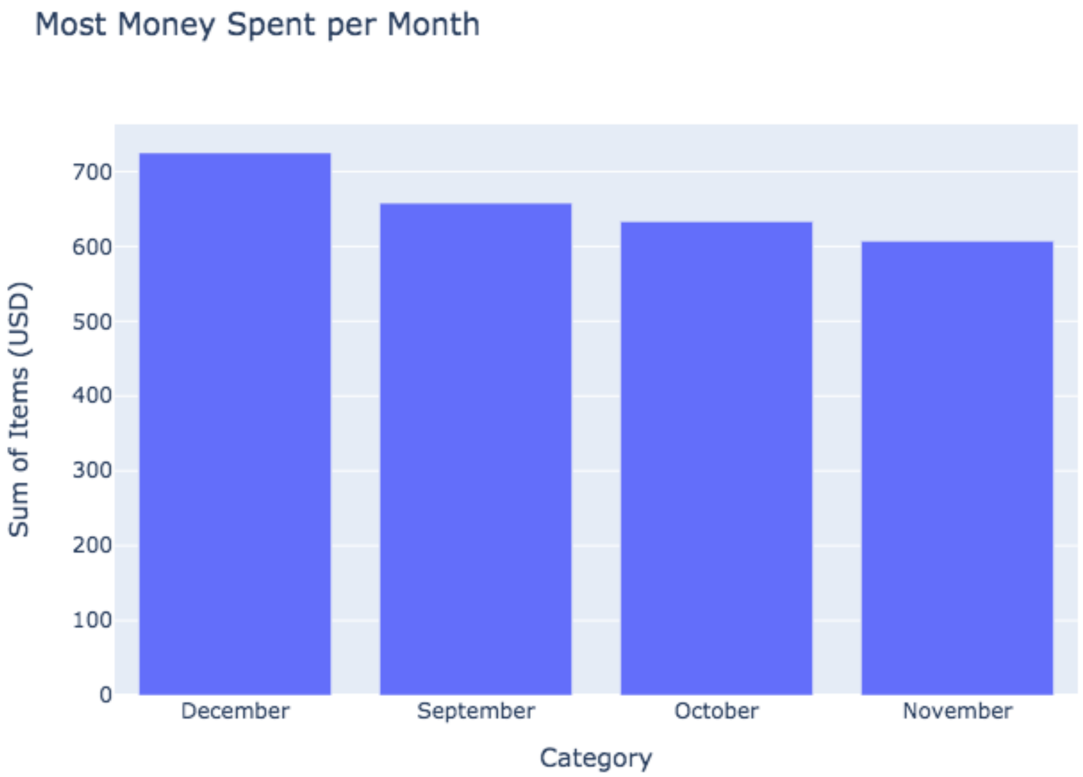
```
    figa.show()
    figa.write_image("./a.png")

    #Sum of Items per Charge Category
    print('\n\nSum of Items per Charge Category:\n')
    print(ds.groupby("Type")["Charge"].sum().sort_values(ascending=False))

    ds5 =
pd.DataFrame(ds.groupby("Type")["Charge"].sum().sort_values(ascending=False
))
    figa = px.bar(x=ds5.index.values, y=ds5['Charge'].values, title="Sum of Items
per Charge Category", labels={'x':'Category', 'y':'Sum of Items (USD)'})
    figa.show()
    figa.write_image("./b.png")

    #Sum of Items per Charge Category
    print('\n\nSum of Items per Charge Category:\n')
    print(ds.groupby("Type")["Charge"].sum().sort_values(ascending=False))

    ds5 =
pd.DataFrame(ds.groupby("Type")["Charge"].sum().sort_values(ascending=False
))
    ds5.head()
    figa = px.bar(x=ds5.index.values, y=ds5['Charge'].values, title="Sum of Items
per Charge Category", labels={'x':'Category', 'y':'Sum of Items (USD)'})
    figa.show()
    figa.write_image("./c.png")
    #Total Balance over time Graph
    totalBal = px.line(ds, x = 'Date', y = 'Running Balance', title='Total Balance over
time (09/2018 - 02/2020)')
    totalBal.show()

    #for x in ds['Date']:
    #    print(x)

    #totalBal2 = px.scatter(ds, x="Date", y="Running Balance", color="Type",
marginal_y="violin",
    #       marginal_x="box", trendline="lowess", title='Total Balance over time
(09/2018 - 02/2020)')

    #totalBal2.show()

    #Month Date Year Insertion
    #REGEX Extract '/' replace with ' ' for easier regex functions
    arr = []
    regex0 = r"/"
    subst0 = " "
    for date in enumerate(ds['Date']):
        arr.insert(int(date[0]), re.sub(regex0, subst0, date[1], 0, re.MULTILINE))

    #REGEX Extract everthing except Year and replace with nothing hence trims
the string and create new column
    year = []
    regex1 = r"[\d]+ [\d]+ "
    subst1 = ""
    for each in enumerate(arr):
        year.insert(int(each[0]), re.sub(regex1, subst1, each[1], 0, re.MULTILINE))
    ds['Year'] = year

    #REGEX Extract everthing except Month and replace with nothing hence trims
the string and create new column.
    month = []
    regex2 = r" [\d]+ [\d]+"
    subst2 = ""
    for each in enumerate(arr):
        month.insert(int(each[0]), re.sub(regex2, subst2, each[1], 0, re.MULTILINE))
    ds['Month'] = month

    #Get month name based on Value and create new column
    monthName = []
    for val in enumerate(month):
        if(val[1] == "1"):
            monthName.insert(int(val[0]), "January" )
        elif(val[1] == "2"):
            monthName.insert(int(val[0]), "February" )
        elif(val[1] == "3"):
            monthName.insert(int(val[0]), "March")
        elif(val[1] == "4"):
            monthName.insert(int(val[0]), "April")
        elif(val[1] == "5"):
            monthName.insert(int(val[0]), "May")
        elif(val[1] == "6"):
            monthName.insert(int(val[0]), "June")
        elif(val[1] == "7"):
            monthName.insert(int(val[0]), "July")
        elif(val[1] == "8"):
            monthName.insert(int(val[0]), "August")
        elif(val[1] == "9"):
            monthName.insert(int(val[0]), "September")
        elif(val[1] == "10"):
            monthName.insert(int(val[0]), "October")
        elif(val[1] == "11"):
            monthName.insert(int(val[0]), "November")
        elif(val[1] == "12"):
            monthName.insert(int(val[0]), "December")
        else:
            monthName.insert(int(val[0]), "N/A")
    ds['MonthName'] = monthName

    #REGEX Extract everthing before Day and replace with nothing hence trims
the string with day and year
    dayyr = []
    regex3 = r"(^[\d]+ )"
    subst3 = ""
    for each in enumerate(arr):
        dayyr.insert(int(each[0]), re.sub(regex3, subst3, each[1], 0, re.MULTILINE))

    #REGEX Extract everthing after Day and replace with nothing hence trims the
string with just day. add to data create new column
    day = []
    regex4 = r"( [\d]+$)"
    subst4 = ""
    for each in enumerate(dayyr):
        day.insert(int(each[0]), re.sub(regex4, subst4, each[1], 0, re.MULTILINE))
    ds['Day'] = day

    #call datetime date function and return day of week for each date.
    day_name = []
    day_code = []
    ROWS = range(0, int(ds.shape[0]))
    for n in ROWS:
        today = datetime.date(2000 + int(ds.iloc[n]['Year']), int(ds.iloc[n]['Month']),
int(ds.iloc[n]['Day']))
        day_name.append(today.strftime("%A"))
        if(today.strftime("%A") == "Sunday"):
            day_code.append(0)
        elif(today.strftime("%A") == "Monday"):
            day_code.append(1)
        elif(today.strftime("%A") == "Tuesday"):
            day_code.append(2)
        elif(today.strftime("%A") == "Wednesday"):
            day_code.append(3)
        elif(today.strftime("%A") == "Thursday"):
            day_code.append(4)
        elif(today.strftime("%A") == "Friday"):
            day_code.append(5)
        elif(today.strftime("%A") == "Saturday"):
            day_code.append(6)
        else:
            day_code.append(None)

    ds['DayName'] = day_name
    ds['DayCode'] = day_code
    #print(day_name)
    display(ds)

    def getNum(x):
        if(x == "Sunday"):
            return 0
        if(x == "Monday"):
            return 1
        if(x == "Tuesday"):
            return 2
        if(x == "Wednesday"):
            return 3
        if(x == "Thursday"):
            return 4
        if(x == "Friday"):
            return 5
        if(x == "Saturday"):
            return 6
        else:
            return None
    #Retrieve weekcode and update in table

    #groupby each year
    wnum = None
    dfwn = []
    dfwn2 = []
    for y in ds['Year'].unique():
        #instantiate the current year scope
        this_year = ds[ds['Year'] == y]
        display(this_year)
        #groupby month in current year
        dfy = []
        dfy2 = []
        print(y)
        for m in this_year['Month'].unique():
            #instantiate the current month scope
            this_month = this_year[this_year['Month'] == m]
            display(this_month)
            this_week = 1
            #first day in dataset of this month
            print('Month: ' + m)
            fom = datetime.date(2000 + int(y), int(m), 1)
            t = fom.strftime("%A")
            weekdaynumoffirstofmonth = getNum(str(t))
            month_vals = []
            month_vals2 = []
            count = 0
            first = 1
            wdnum = None
            for d in this_month['Day']:
                weekdaynumwewant = weekdaynumoffirstofmonth
                start = 1
                print('First & Last')
                print(start)
                print(d)
                while (start <= int(d)):
                    if(weekdaynumwewant < 6):
                        weekdaynumwewant += 1
                    else:
                        weekdaynumwewant = 0
                        this_week  +=1
                    start +=1
                wdnum = weekdaynumwewant
                wnum = this_week
                print('wnum = ' + str(wdnum) )
                month_vals.insert(count, wdnum)
                month_vals2.insert(count, wnum)
                count+=1

            print('\nmonth_vals\n')
            print(month_vals)
            dfy += month_vals
            dfy2 +=month_vals2
            print('\nEDNOFMONTH\n')

        print('\year_vals\n')
        print(dfy)
        dfwn += dfy
        dfwn2 += dfy2
        print('\nENDOFYEAR\n')
    print("\nDOC_vals\n")
    display(dfwn)
    display(dfwn2)
    print("\nENDOFDOC\n")

    ds['WeekNum'] = dfwn
    ds['wn'] = dfwn2
```

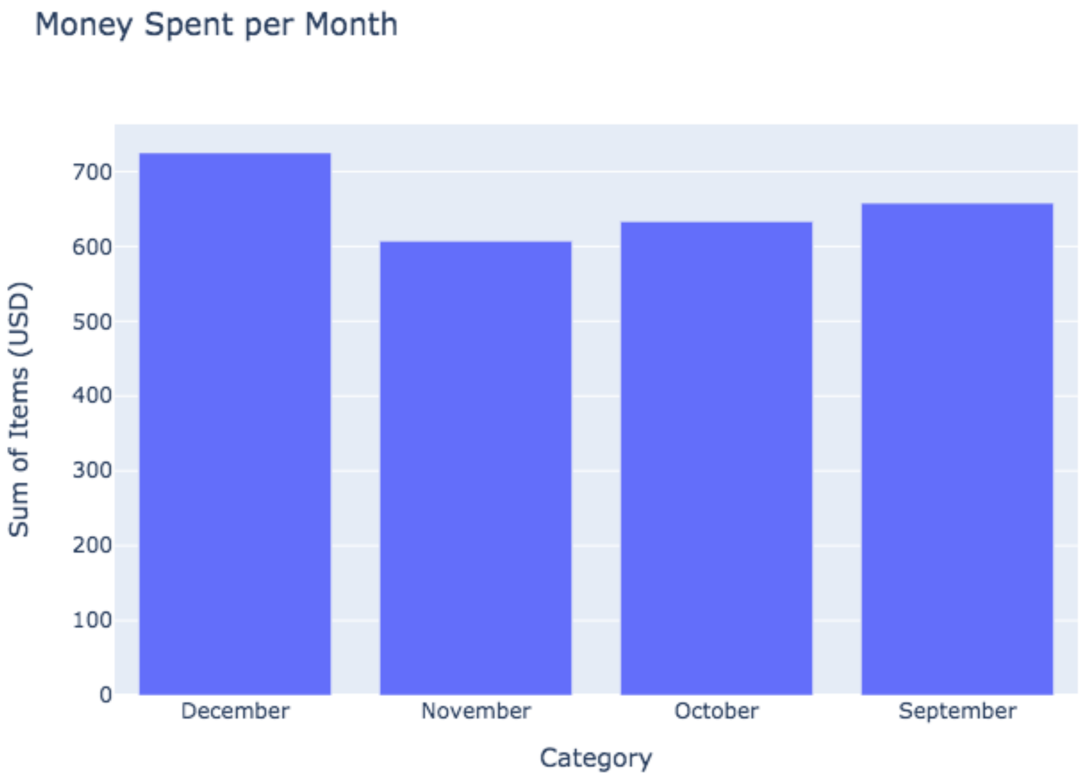
```
    #heatmap fore each month based on what you spent the most on that day per
week
    print("Daily Max Purchase per month")
    #for month in ds['Year'].unique():
    #    for week




    #Total Money Spent Per Month (descending AND chronilogically)
    print('\n\nSum of Items per Charge Category:\n')
    count = 0
    for each in ds['Year'].unique():
        df_sample = ds[ds['Year'] == each]

print(df_sample.groupby("MonthName")["Debit"].sum().sort_values(ascending=F
alse))
        ds5 =
pd.DataFrame(df_sample.groupby("MonthName")["Debit"].sum().sort_values(asc
ending=False))
        figa = px.bar(x=ds5.index.values, y=ds5['Debit'].values, title="Most Money
Spent per Month", labels={'x':'Category', 'y':'Sum of Items (USD)'})
        figa.write_image("./" + str(count) + "aa.png")
        figa.show()

        print(df_sample.groupby("MonthName")["Debit"].sum())
        ds6 = pd.DataFrame(df_sample.groupby("MonthName")["Debit"].sum())
        figb = px.bar(x=ds6.index.values, y=ds6['Debit'].values, title="Money Spent
per Month", labels={'x':'Category', 'y':'Sum of Items (USD)'})
        figb.write_image("./" + str(count) + "bb.png")
        figb.show()

        count+=1



    display(ds)






#Caller
if __name__ == "__main__":
    Main()
```

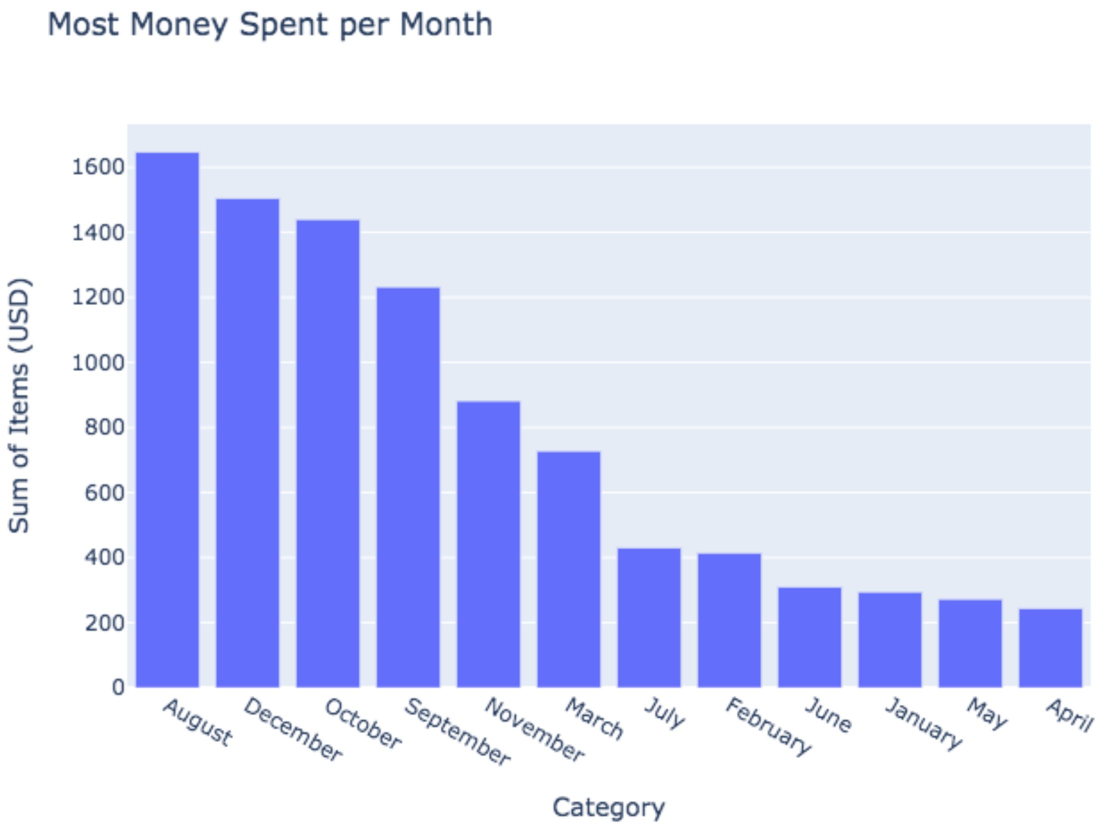
b. A REPORT DESCRIBING THE DATA, THE GOAL, DATA ANALYSIS METHOD, DATA VIS, AND CONCLUSIONS

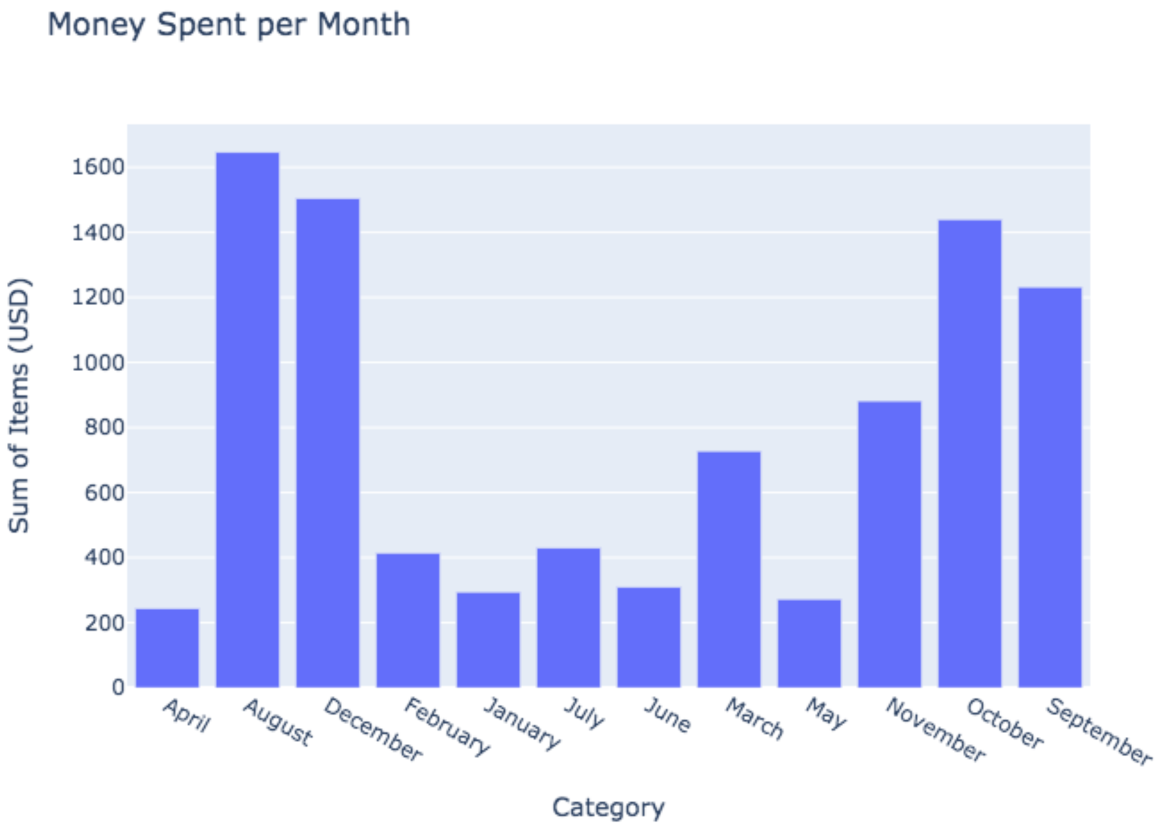
   a. What I tried to do was create a sort of budget for me to look at and create a projection of earnings based on my suntrust history data, but was unable to figure out the projection aspect due to time constraints
   b. I did a regex with the month year and day number and then made a column of each and also a month name column. this helped me to create visualizations based on a month, or a year, or even a day. I wanted to get the day of week based on the date and the week number by month and create a categorical heatmap per month [per year of money spent per day
   c. I had a few other visualizations such as a daily balance, and I wanted to create a projected rate of change for bank account total and was not able to figure that out. I had a sum of items per category
   d. I also had money spent per month per year. One was sorted by most to least and the other I wanted chronologically but it wouldn’t work and it was alphabetic

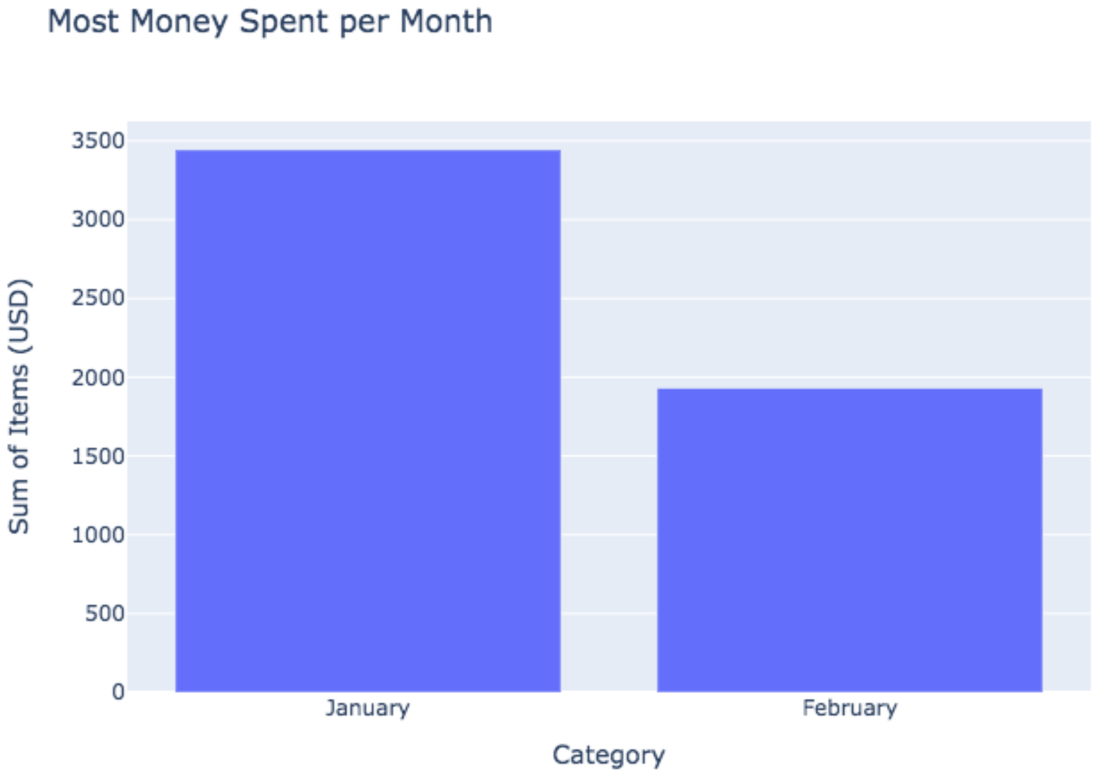
Screenshots below

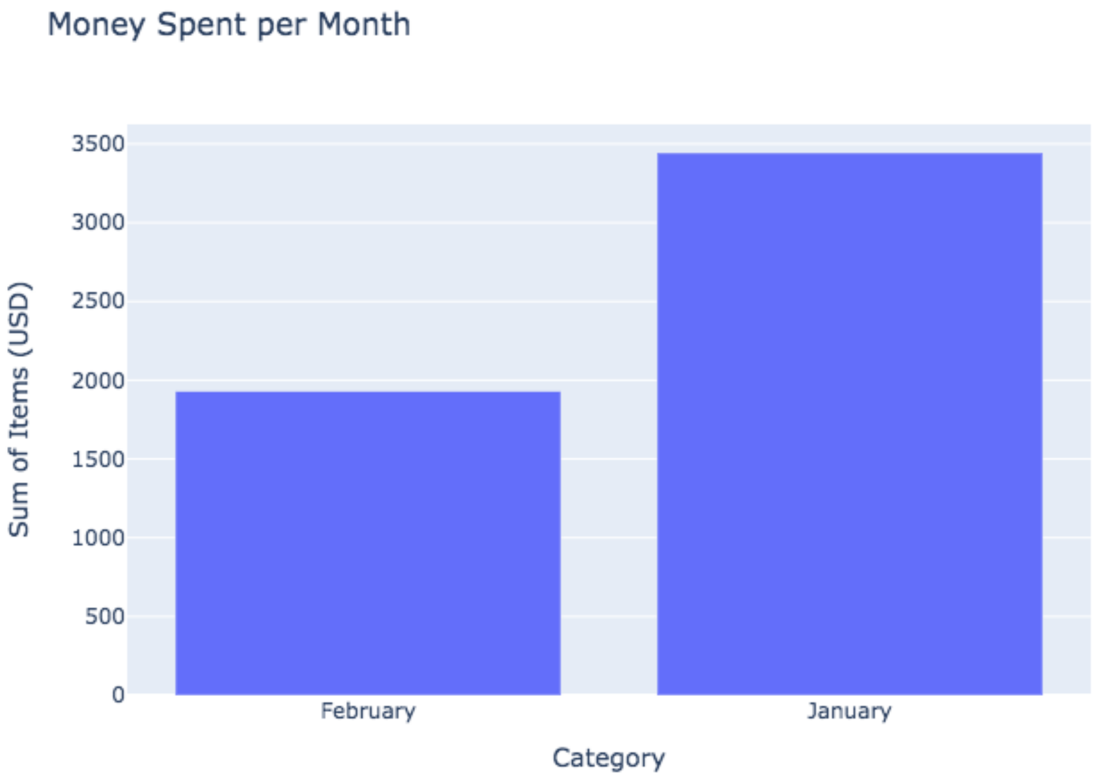

Most Money Spent per Month
Sum of Items (USD)
0
100
200
300
400
500
600
700
December
September
October
November
Category

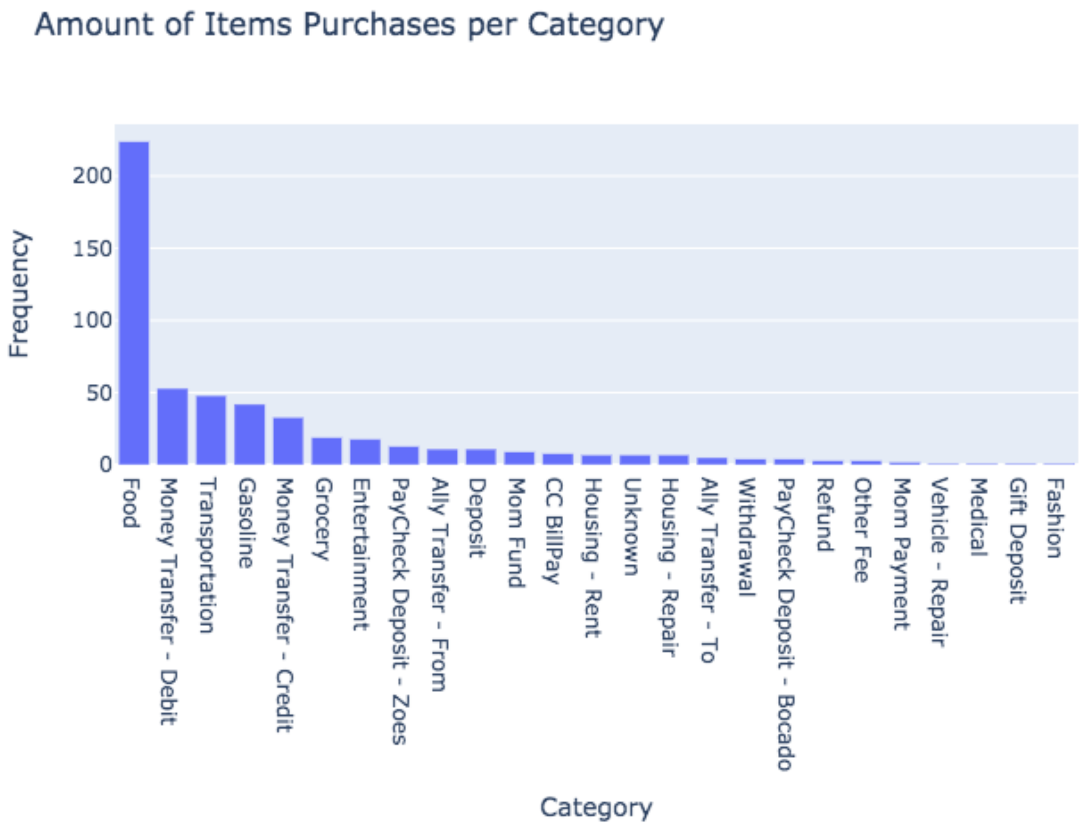

Money Spent per Month
Sum of Items (USD)
0
100
200
300
400
500
600
700
December
November
October
September
Category

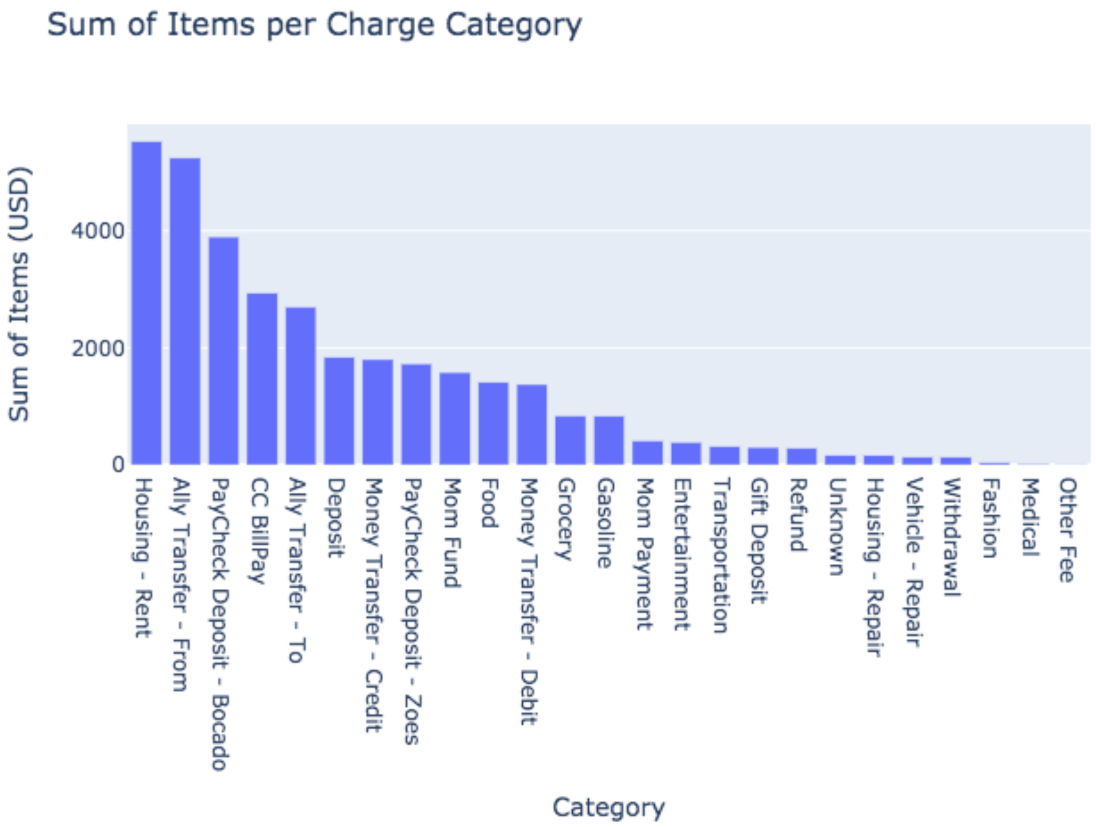

Most Money Spent per Month
Sum of Items (USD)
0
200
400
600
800
1000
1200
1400
1600
August
December
October
September
November
March
July
February
June
January
May
April
Category

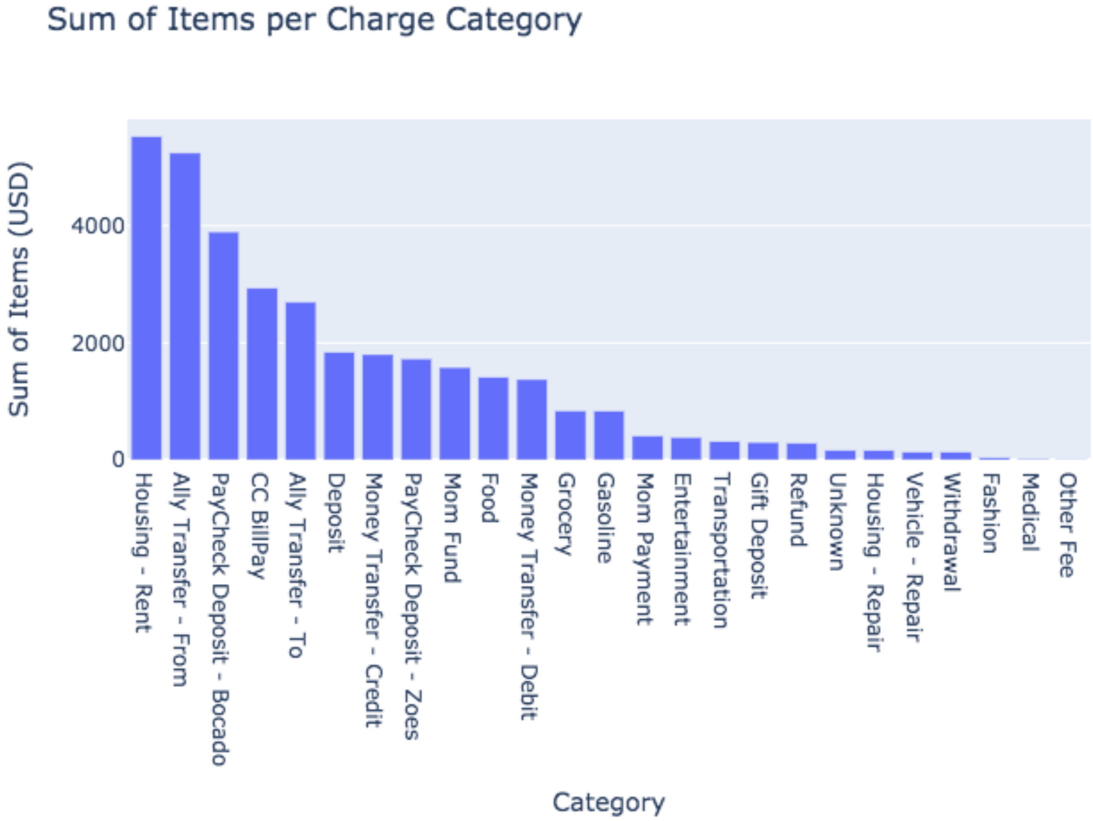

Money Spent per Month
Sum of Items (USD)
1600
1400
1200
1000
800
600
400
200
0
April
August
December
February
January
July
June
March
May
November
October
September
Category


Most Money Spent per Month
Sum of Items (USD)
0
500
1000
1500
2000
2500
3000
3500
January
February
Category

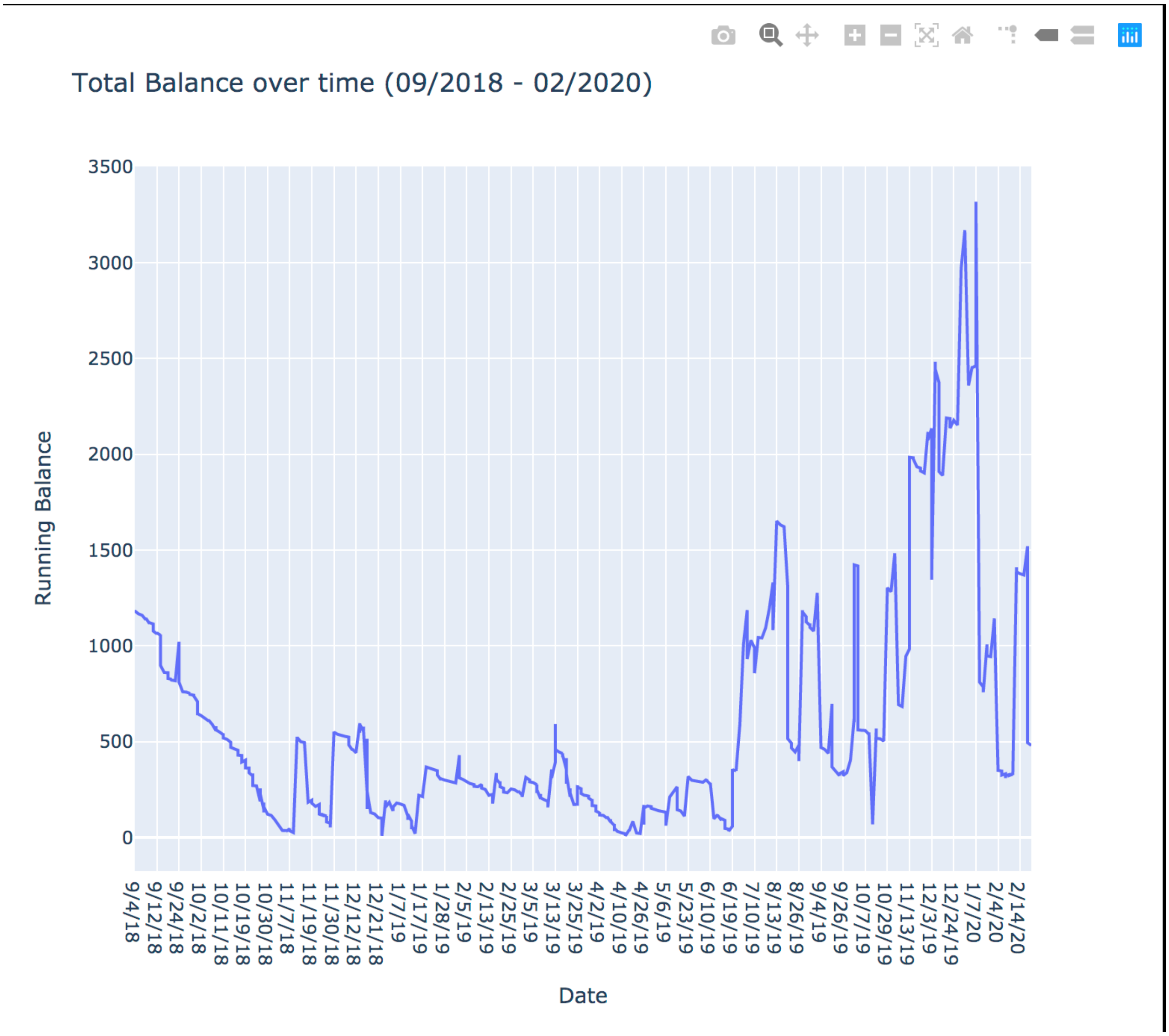

Money Spent per Month
Sum of Items (USD)
3500
3000
2500
2000
1500
1000
500
0
February
January
Category

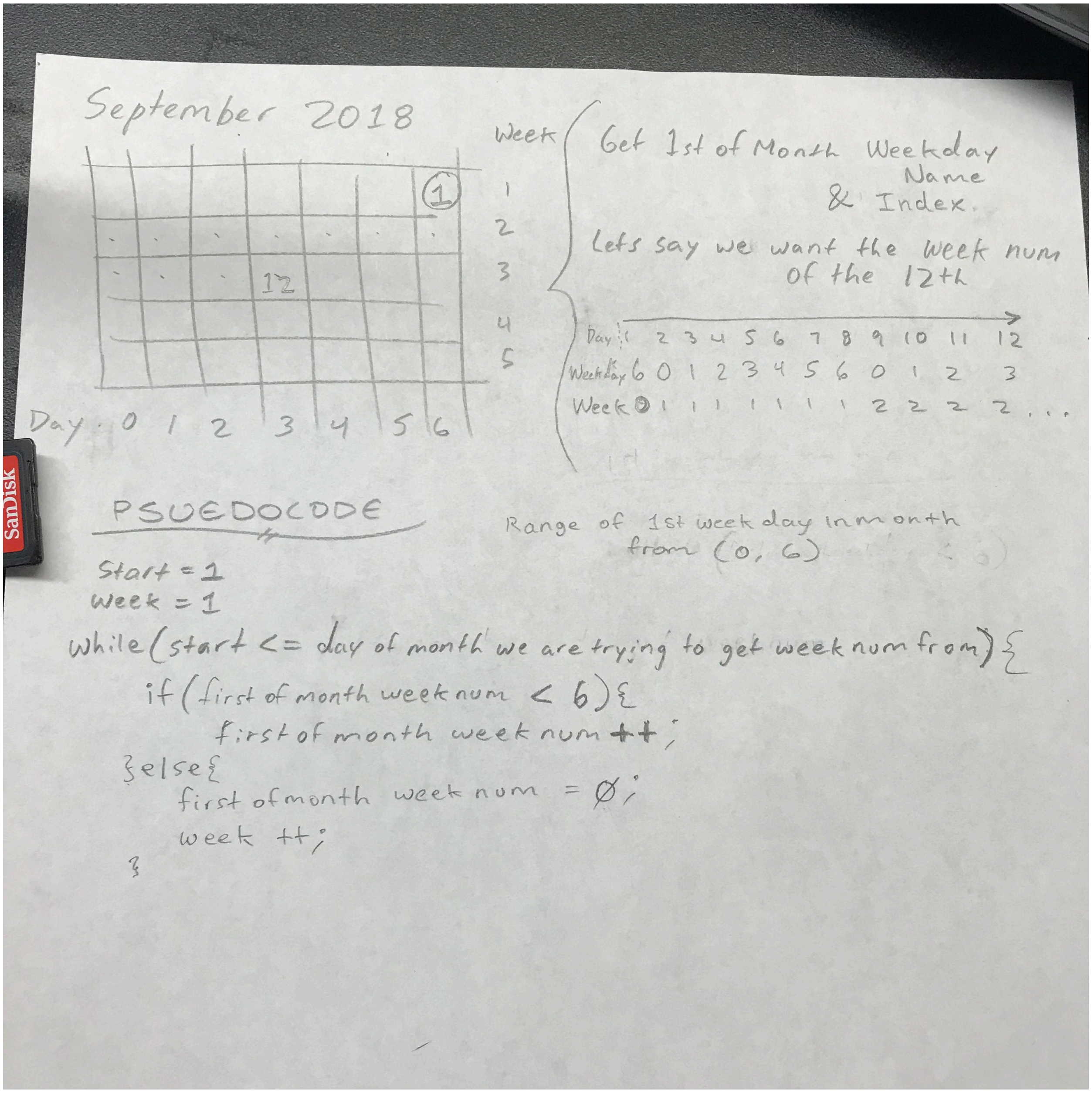

Amount of Items Purchases per Category
Frequency
0
50
100
150
200
Category
Food
Money Transfer - Debit
Transportation
Gasoline
Money Transfer - Credit
Grocery
Entertainment
PayCheck Deposit - Zoes
Ally Transfer - From
Deposit
Mom Fund
CC BillPay
Housing - Rent
Unknown
Housing - Repair
Ally Transfer - To
Withdrawal
PayCheck Deposit - Bocado
Refund
Other Fee
Mom Payment
Vehicle - Repair
Medical
Gift Deposit
Fashion


Sum of Items per Charge Category
Sum of Items (USD)
0
2000
4000
Housing - Rent
Ally Transfer - From
PayCheck Deposit - Bocado
CC BillPay
Ally Transfer - To
Deposit
Money Transfer - Credit
PayCheck Deposit - Zoes
Mom Fund
Food
Money Transfer - Debit
Grocery
Gasoline
Mom Payment
Entertainment
Transportation
Gift Deposit
Refund
Unknown
Housing - Repair
Vehicle - Repair
Withdrawal
Fashion
Medical
Other Fee
Category



Sum of Items per Charge Category


Total Balance over time (09/2018 - 02/2020)



September 2018

Day	0	1	2	3	4	5	6	Week
							1	1
								2
				12				3
								4
								5

Get 1st of Month Weekday Name & Index.

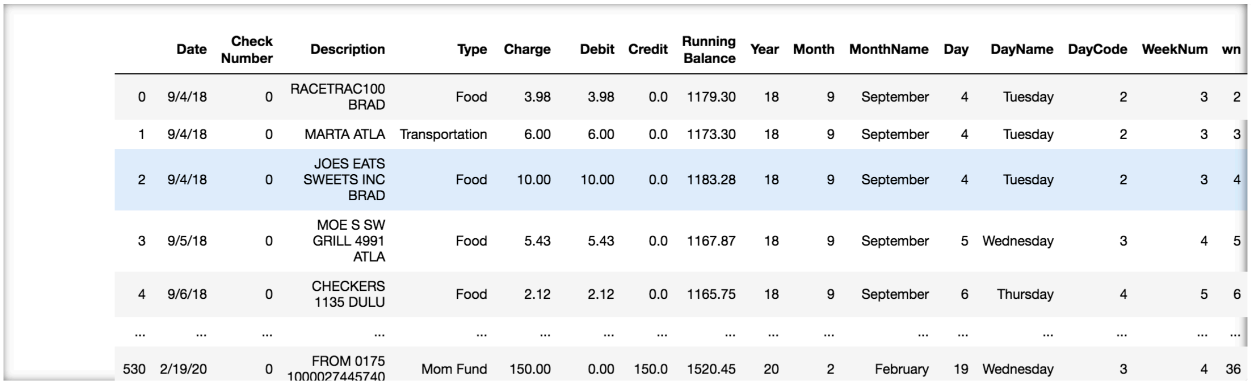
Lets say we want the week num of the 12th

Day	1	2	3	4	5	6	7	8	9	10	11	12	
Weekday	6	0	1	2	3	4	5	6	0	1	2	3	
Week	0	1	1	1	1	1	1	1	2	2	2	2	...

Range of 1st week day in month from (0, 6)

PSUEDOCODE

```
Start = 1
week = 1
while(start <= day of month we are trying to get week num from){
    if(first of month week num < 6){
        first of month week num ++;
    }else{
        first of month week num = Ø;
        week ++;
    }
```



	Date	Check Number	Description	Type	Charge	Debit	Credit	Running Balance	Year	Month	MonthName	Day	DayName	DayCode	WeekNum	wn
0	9/4/18	0	RACETRAC100 BRAD	Food	3.98	3.98	0.0	1179.30	18	9	September	4	Tuesday	2	3	2
1	9/4/18	0	MARTA ATLA	Transportation	6.00	6.00	0.0	1173.30	18	9	September	4	Tuesday	2	3	3
2	9/4/18	0	JOES EATS SWEETS INC BRAD	Food	10.00	10.00	0.0	1183.28	18	9	September	4	Tuesday	2	3	4
3	9/5/18	0	MOE S SW GRILL 4991 ATLA	Food	5.43	5.43	0.0	1167.87	18	9	September	5	Wednesday	3	4	5
4	9/6/18	0	CHECKERS 1135 DULU	Food	2.12	2.12	0.0	1165.75	18	9	September	6	Thursday	4	5	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
530	2/19/20	0	FROM 0175 1000027445740	Mom Fund	150.00	0.00	150.0	1520.45	20	2	February	19	Wednesday	3	4	36