C. Anirudh Bhardwaj

EY Project

Simulation of Bank Transactions

‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑

import numpy as np

import pandas as pd

import math

import matplotlib

import matplotlib.pyplot as plt

plt.show()

branch\_a\_cust\_data = {'Branch' :'A', 'Balance': 100000\*abs((np.random.randn(1000)))};

df = pd.DataFrame(data= branch\_a\_cust\_data);

Branch\_a = df;

branch\_b\_cust\_data = {'Branch' :'B', 'Balance': 100000\*abs((np.random.randn(3000)))};

df = pd.DataFrame(data= branch\_b\_cust\_data);

Branch\_b = df;

branch\_c\_cust\_data = {'Branch' :'C', 'Balance': 100000\*abs((np.random.randn(2000)))};

df = pd.DataFrame(data= branch\_c\_cust\_data);

Branch\_c =df;

Branch\_a\_Record = Branch\_a.copy()

Branch\_b\_Record = Branch\_b.copy()

Branch\_c\_Record = Branch\_c.copy()

Ba\_resrv = 0.1\*(Branch\_a.Balance.sum());

Bb\_resrv = 0.1\*(Branch\_b.Balance.sum());

Bc\_resrv = 0.1\*(Branch\_c.Balance.sum());

print Ba\_resrv,Bb\_resrv,Bc\_resrv

Paa = (1)/((1/10.0000)+(1/5.0000000)+1)

Pab = (1/10.000)/((1/10.000000000000)+(1/5.0000000)+1)

Pac = (1/5.000)/((1/10.000000000000)+(1/5.0000000)+1)

Pba = (1/10.000)/((1/10.0000)+(1/10.0000000)+1)

Pbb = (1)/((1/10.0000)+(1/10.0000000)+1)

Pbc = (1/10.000)/((1/10.0000)+(1/10.0000000)+1)

Pca = (1/5.000)/((1/10.0000)+(1/5.0000000)+1)

Pcb = (1/10.000)/((1/10.0000)+(1/5.0000000)+1)

Pcc = (1)/((1/10.0000)+(1/5.0000000)+1)

print Paa,Pab,Pac,(Paa+Pab+Pac)

print Pba,Pbb,Pbc,(Pba+Pbb+Pbc)

print Pca,Pcb,Pcc,(Pca+Pcb+Pcc)

Branch\_a\_Record = Branch\_a.Balance.copy()

Branch\_b\_Record = Branch\_b.Balance.copy()

Branch\_c\_Record = Branch\_c.Balance.copy()

c=1

s\_ta = {c};

Old\_bal\_ser\_A = pd.Series();

Old\_bal\_ser\_B = pd.Series();

Old\_bal\_ser\_C = pd.Series();

New\_bal\_ser\_A = pd.Series();

New\_bal\_ser\_B = pd.Series();

New\_bal\_ser\_C = pd.Series();

Total\_IO\_ser = pd.Series();

#dftttta = pd.DataFrame({'Old\_A': [c] ,'New\_A': [c] ,'Old\_B': [c],'New\_B': [c],'Old\_C': [c],'New\_C': [c],'Change\_Val': [c]},index=[0]);

for j in range(1,91,1):

for i in range(0,3,1):

a= math.floor(np.random.uniform(low=0, high=1000, size=None))

print a

c=a;

Cb= Branch\_a.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_a.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_a.Balance

Branch\_a.Balance = Branch\_a.Balance.replace(Branch\_a.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Paa):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Paa<x<(Paa+Pab)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

for i in range(0,10,1):

a= math.floor(np.random.uniform(low=0, high=3000, size=None))

print a

c=a;

Cb= Branch\_b.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_b.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_b.Balance

Branch\_b.Balance = Branch\_b.Balance.replace(Branch\_b.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Pba):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Pba<x<(Pba+Pbb)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

for i in range(0,6,1):

a= math.floor(np.random.uniform(low=0, high=1000, size=None))

print a

c=a;

Cb= Branch\_c.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_c.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_c.Balance

Branch\_c.Balance = Branch\_c.Balance.replace(Branch\_c.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Pca):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Pca<x<(Pca+Pcb)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

Branch\_a\_Record = pd.concat([Branch\_a\_Record, Branch\_a.Balance], axis =1);

Branch\_b\_Record = pd.concat([Branch\_b\_Record, Branch\_b.Balance], axis =1);

Branch\_c\_Record = pd.concat([Branch\_c\_Record, Branch\_c.Balance], axis =1);

temp\_a = Ba\_resrv;

temp\_b = Bb\_resrv;

temp\_c = Bc\_resrv;

Ba\_resrv = 0.1\*(Branch\_a.Balance.sum());

Bb\_resrv = 0.1\*(Branch\_b.Balance.sum());

Bc\_resrv = 0.1\*(Branch\_c.Balance.sum());

t\_val = ( temp\_a[0] - Ba\_resrv) + ( temp\_b[0] - Bb\_resrv) + ( temp\_c[0] - Bc\_resrv);

Old\_bal\_ser\_A = Old\_bal\_ser\_A.set\_value(j,temp\_a[0]);

Old\_bal\_ser\_B = Old\_bal\_ser\_B.set\_value(j,temp\_b[0]);

Old\_bal\_ser\_C = Old\_bal\_ser\_C.set\_value(j,temp\_c[0]);

New\_bal\_ser\_A = New\_bal\_ser\_A.set\_value(j,Ba\_resrv);

New\_bal\_ser\_B = New\_bal\_ser\_B.set\_value(j,Bb\_resrv);

New\_bal\_ser\_C = New\_bal\_ser\_C.set\_value(j,Bc\_resrv);

Total\_IO\_ser = Total\_IO\_ser.set\_value(j,t\_val);

for i in range(0,4,1):

a= math.floor(np.random.uniform(low=0, high=1000, size=None))

print a

c=a;

Cb= Branch\_a.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_a.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_a.Balance

Branch\_a.Balance = Branch\_a.Balance.replace(Branch\_a.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Paa):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Paa<x<(Paa+Pab)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

for i in range(0,10,1):

a= math.floor(np.random.uniform(low=0, high=3000, size=None))

print a

c=a;

Cb= Branch\_b.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_b.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_b.Balance

Branch\_b.Balance = Branch\_b.Balance.replace(Branch\_b.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Pba):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Pba<x<(Pba+Pbb)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

for i in range(0,7,1):

a= math.floor(np.random.uniform(low=0, high=1000, size=None))

print a

c=a;

Cb= Branch\_c.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_c.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_c.Balance

Branch\_c.Balance = Branch\_c.Balance.replace(Branch\_c.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Pca):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Pca<x<(Pca+Pcb)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

Branch\_a\_Record = pd.concat([Branch\_a\_Record, Branch\_a.Balance], axis =1);

Branch\_b\_Record = pd.concat([Branch\_b\_Record, Branch\_b.Balance], axis =1);

Branch\_c\_Record = pd.concat([Branch\_c\_Record, Branch\_c.Balance], axis =1);

temp\_a = Ba\_resrv;

temp\_b = Bb\_resrv;

temp\_c = Bc\_resrv;

Ba\_resrv = 0.1\*(Branch\_a.Balance.sum());

Bb\_resrv = 0.1\*(Branch\_b.Balance.sum());

Bc\_resrv = 0.1\*(Branch\_c.Balance.sum());

t\_val = ( temp\_a[0] - Ba\_resrv) + ( temp\_b[0] - Bb\_resrv) + ( temp\_c[0] - Bc\_resrv);

Old\_bal\_ser\_A = Old\_bal\_ser\_A.set\_value(j,temp\_a[0]);

Old\_bal\_ser\_B = Old\_bal\_ser\_B.set\_value(j,temp\_b[0]);

Old\_bal\_ser\_C = Old\_bal\_ser\_C.set\_value(j,temp\_c[0]);

New\_bal\_ser\_A = New\_bal\_ser\_A.set\_value(j,Ba\_resrv);

New\_bal\_ser\_B = New\_bal\_ser\_B.set\_value(j,Bb\_resrv);

New\_bal\_ser\_C = New\_bal\_ser\_C.set\_value(j,Bc\_resrv);

Total\_IO\_ser = Total\_IO\_ser.set\_value(j,t\_val);

for i in range(0,3,1):

a= math.floor(np.random.uniform(low=0, high=1000, size=None))

print a

c=a;

Cb= Branch\_a.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_a.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_a.Balance

Branch\_a.Balance = Branch\_a.Balance.replace(Branch\_a.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Paa):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Paa<x<(Paa+Pab)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_a.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_a.set\_value(a,'Balance',New\_bal, takeable=False)

for i in range(0,10,1):

a= math.floor(np.random.uniform(low=0, high=3000, size=None))

print a

c=a;

Cb= Branch\_b.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_b.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_b.Balance

Branch\_b.Balance = Branch\_b.Balance.replace(Branch\_b.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Pba):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Pba<x<(Pba+Pbb)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_b.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_b.set\_value(a,'Balance',New\_bal, takeable=False)

for i in range(0,6,1):

a= math.floor(np.random.uniform(low=0, high=1000, size=None))

print a

c=a;

Cb= Branch\_c.Balance[a]

Sdb= Cb/3.0000

MV = Sdb \* (np.random.randn(1))

New\_bal = Branch\_c.Balance[a] + MV

if(New\_bal<0):

New\_bal = Branch\_c.Balance

Branch\_c.Balance = Branch\_c.Balance.replace(Branch\_c.Balance[i],New\_bal)

x= np.random.uniform(0,1)

if(x<Pca):

print 'Branch A';

if((Ba\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Ba\_resrv = Ba\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

elif(Pca<x<(Pca+Pcb)):

print 'Branch B';

if((Bb\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Bb\_resrv = Bb\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

else:

print 'Branch C';

if((Bc\_resrv+MV)>0 and ((Branch\_c.Balance[a] + MV)>0)):

Bc\_resrv = Bc\_resrv + MV;

Branch\_c.set\_value(a,'Balance',New\_bal, takeable=False)

Branch\_a\_Record = pd.concat([Branch\_a\_Record, Branch\_a.Balance], axis =1);

Branch\_b\_Record = pd.concat([Branch\_b\_Record, Branch\_b.Balance], axis =1);

Branch\_c\_Record = pd.concat([Branch\_c\_Record, Branch\_c.Balance], axis =1);

temp\_a = Ba\_resrv;

temp\_b = Bb\_resrv;

temp\_c = Bc\_resrv;

Ba\_resrv = 0.1\*(Branch\_a.Balance.sum());

Bb\_resrv = 0.1\*(Branch\_b.Balance.sum());

Bc\_resrv = 0.1\*(Branch\_c.Balance.sum());

t\_val = ( temp\_a[0] - Ba\_resrv) + ( temp\_b[0] - Bb\_resrv) + ( temp\_c[0] - Bc\_resrv);

Old\_bal\_ser\_A = Old\_bal\_ser\_A.set\_value(j,temp\_a[0]);

Old\_bal\_ser\_B = Old\_bal\_ser\_B.set\_value(j,temp\_b[0]);

Old\_bal\_ser\_C = Old\_bal\_ser\_C.set\_value(j,temp\_c[0]);

New\_bal\_ser\_A = New\_bal\_ser\_A.set\_value(j,Ba\_resrv);

New\_bal\_ser\_B = New\_bal\_ser\_B.set\_value(j,Bb\_resrv);

New\_bal\_ser\_C = New\_bal\_ser\_C.set\_value(j,Bc\_resrv);

Total\_IO\_ser = Total\_IO\_ser.set\_value(j,t\_val);

Net\_inventory\_change\_A = pd.DataFrame(Old\_bal\_ser\_A)

New\_inventory\_change\_A = pd.DataFrame(New\_bal\_ser\_A)

result\_A = pd.concat([Net\_inventory\_change\_A,New\_inventory\_change\_A,(New\_inventory\_change\_A - Net\_inventory\_change\_A)], axis=1);

Net\_inventory\_change\_B = pd.DataFrame(Old\_bal\_ser\_B)

New\_inventory\_change\_B = pd.DataFrame(New\_bal\_ser\_B)

result\_B = pd.concat([Net\_inventory\_change\_B,New\_inventory\_change\_B,(New\_inventory\_change\_B - Net\_inventory\_change\_B)], axis=1);

Net\_inventory\_change\_C = pd.DataFrame(Old\_bal\_ser\_C)

New\_inventory\_change\_C = pd.DataFrame(New\_bal\_ser\_C)

result\_C = pd.concat([Net\_inventory\_change\_C,New\_inventory\_change\_C,(New\_inventory\_change\_C - Net\_inventory\_change\_C)], axis=1);

t\_val = result\_A.iloc[:, 2] + result\_B.iloc[: , 2] + result\_C.iloc[: , 2]

Total\_Change = pd.concat([result\_A.iloc[:, 2],result\_B.iloc[: , 2],result\_C.iloc[: , 2],t\_val], axis=1);

Total\_Change.columns = ['A', 'B','C','Total']

Total\_Change.to\_csv('Total\_Change\_V2.csv')

Branch\_a\_Record.to\_csv('Branch\_A\_V2.csv')

Branch\_b\_Record.to\_csv('Branch\_B\_V2.csv')

Branch\_c\_Record.to\_csv('Branch\_C\_V2.csv')

result\_A.to\_csv('Transaction\_log\_A\_V2.csv')

result\_B.to\_csv('Transaction\_log\_B\_V2.csv')

result\_C.to\_csv('Transaction\_log\_C\_V2.csv')