CSV TO HORUS

# Utility Start CSV to HORUS =================================

# Standard Tools

#=============================================================

import pandas as pd

# Input Agreement ============================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/Country\_Code.csv'

InputData=pd.read\_csv(sInputFileName,encoding="latin-1")

print('Input Data Values ===================================')

print(InputData)

print('=====================================================')

# Processing Rules ===========================================

ProcessData=InputData

# Remove columns ISO-2-Code and ISO-3-CODE

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)

ProcessData.drop('ISO-3-Code', axis=1,inplace=True)

# Rename Country and ISO-M49

ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)

ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

# Set new Index

ProcessData.set\_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True)

print('Process Data Values =================================')

print(ProcessData)

print('=====================================================')

# Output Agreement ===========================================

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('CSV to HORUS - Done')

print('Rinki\_mscit')

# Utility done ===============================================

**DATABASE TO HORUS**

# Utility Start Database to HORUS =================================

# Standard Tools

#=============================================================

import pandas as pd

import sqlite3 as sq

# Input Agreement ============================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/utility.db'

sInputTable='Country\_Code'

conn = sq.connect(sInputFileName)

sSQL='select \* FROM ' + sInputTable + ';'

InputData=pd.read\_sql\_query(sSQL, conn)

print('Input Data Values ===================================')

print(InputData)

print('=====================================================')

# Processing Rules ===========================================

ProcessData=InputData

# Remove columns ISO-2-Code and ISO-3-CODE

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)

ProcessData.drop('ISO-3-Code', axis=1,inplace=True)

# Rename Country and ISO-M49

ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)

ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

# Set new Index

ProcessData.set\_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True)

print('Process Data Values =================================')

print(ProcessData)

print('=====================================================')

# Output Agreement ===========================================

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('Database to HORUS - Done')

# Utility done ===============================================

**IMAGE TO HORUS**

from skimage import io

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

# Input Agreement

sInputFileName='C:/VKHCG/05-DS/9999-Data/ice-cream.jpg'

InputData = io.imread(sInputFileName, pilmode='RGBA')

plt.imshow(InputData)

InputData. shape

print('Input Data Values')

print('X: ',InputData.shape[0])

print('Y: ', InputData. shape[1])

print('RGBA: ', InputData.shape[2])

ProcessRawData=InputData.flatten()

y=InputData.shape[2] + 2

x=int(ProcessRawData.shape[0]/y)

ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))

ProcessRawData

ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))

sColumns= [ 'XAxis', 'YAxis', 'Red', 'Green', 'Blue','Aplha']

ProcessData.columns=sColumns

ProcessData

print('Rows: ',ProcessData.shape[0])

print('Columns :',ProcessData.shape[1])

OutputData = ProcessData

OutputData.to\_csv('Image to HORUS.csv', index = False)

print('Rinki\_mscit')

**AUDIO TO HORUS**

# Utility Start Audio to HORUS ===============================

# Standard Tools

#=============================================================

from scipy.io import wavfile

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#=============================================================

def show\_info(aname, a,r):

print ('----------------')

print ("Audio:", aname)

print ('----------------')

print ("Rate:", r)

print ('----------------')

print ("shape:", a.shape)

print ("dtype:", a.dtype)

print ("min, max:", a.min(), a.max())

print ('----------------')

plot\_info(aname, a,r)

#=============================================================

def plot\_info(aname, a,r):

sTitle= 'Signal Wave - '+ aname + ' at ' + str(r) + 'hz'

plt.title(sTitle)

sLegend=[]

for c in range(a.shape[1]):

sLabel = 'Ch' + str(c+1)

sLegend=sLegend+[str(c+1)]

plt.plot(a[:,c], label=sLabel)

plt.legend(sLegend)

plt.show()

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'

print('=====================================================')

print('Processing : ', sInputFileName)

print('=====================================================')

InputRate, InputData = wavfile.read(sInputFileName)

show\_info("2 channel", InputData,InputRate)

ProcessData=pd.DataFrame(InputData)

sColumns= ['Ch1','Ch2']

ProcessData.columns=sColumns

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'

OutputData.to\_csv(sOutputFileName, index = False)

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'

print('=====================================================')

print('Processing : ', sInputFileName)

print('=====================================================')

InputRate, InputData = wavfile.read(sInputFileName)

show\_info("4 channel", InputData,InputRate)

ProcessData=pd.DataFrame(InputData)

sColumns= ['Ch1','Ch2','Ch3', 'Ch4']

ProcessData.columns=sColumns

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'

OutputData.to\_csv(sOutputFileName, index = False)

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'

print('=====================================================')

print('Processing : ', sInputFileName)

print('=====================================================')

InputRate, InputData = wavfile.read(sInputFileName)

show\_info("6 channel", InputData,InputRate)

ProcessData=pd.DataFrame(InputData)

sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']

ProcessData.columns=sColumns

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'

OutputData.to\_csv(sOutputFileName, index = False)

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/8ch-sound.wav'

print('=====================================================')

print('Processing : ', sInputFileName)

print('=====================================================')

InputRate, InputData = wavfile.read(sInputFileName)

show\_info("8 channel", InputData,InputRate)

ProcessData=pd.DataFrame(InputData)

sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']

**FIXER UTILITIES**

print('Rinki sokhi practical 3')

print('#1 Removing leading or lagging spaces from a data entry');

baddata = " Data Science with too many spaces is bad!!! "

print('>',baddata,'<')

cleandata=baddata.strip()

print('>',cleandata,'<')

import string

print('#2 Removing nonprintable characters from a data entry')

printable = set(string.printable)

baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!"

cleandata=''.join(filter(lambda x: x in string.printable,baddata))

print('Bad Data : ',baddata);

print('Clean Data : ',cleandata)

import datetime

print('# 3 Reformatting data entry to match specific formatting criteria.')

baddate = datetime.date(2019, 10, 31)

baddata=format(baddate,'%Y-%m-%d')

gooddate = datetime.datetime.strptime(baddata,'%Y-%m-%d')

gooddata=format(gooddate,'%d %B %Y')

print('Bad Data : ',baddata)

print('Good Data : ',gooddata)

**DATA BINNING AND BUCKETING**

import numpy as np

import matplotlib.mlab as mlab

import matplotlib.pyplot as plt

import scipy.stats as stats

np.random.seed(0)

# example data

mu = 90 # mean of distribution

sigma = 25 # standard deviation of distribution

x = mu + sigma \* np.random.randn(5000)

num\_bins = 25

fig, ax = plt.subplots()

# the histogram of the data

n, bins, patches = ax.hist(x, num\_bins, density=1)

# add a 'best fit' line

y = stats.norm.pdf(bins, mu, sigma)

# mlab.normpdf(bins, mu, sigma)

ax.plot(bins, y, '--')

ax.set\_xlabel('Example Data')

ax.set\_ylabel('Probability density')

sTitle=r'Histogram ' + str(len(x)) + ' entries into ' + str(num\_bins) + ' Bins: $\mu=' + str(mu) + '$,$\sigma=' +str(sigma) + '$'

ax.set\_title(sTitle)

fig.tight\_layout()

sPathFig='C:/VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png'

fig.savefig(sPathFig)

**AVEARAGING DATA .UTILITIES AND AUDUTING NP**

import pandas as pd

################################################################

InputFileName='IP\_DATA\_CORE.csv'

OutputFileName='Retrieve\_Router\_Location.csv'

Base='C:/VKHCG'

print('################################')

print('Working Base :',Base, ' using ')

print('################################')

sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName

print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False,usecols=['Country','Place

Name','Latitude','Longitude'], encoding="latin-1")

IP\_DATA\_ALL.rename(columns={'Place Name': 'Place\_Name'}, inplace=True)

AllData=IP\_DATA\_ALL[['Country', 'Place\_Name','Latitude']]

print(AllData)

MeanData=AllData.groupby(['Country', 'Place\_Name'])['Latitude'].mean()

print(MeanData)

################################################################

**LOGGING**

VKHCG\_77 YOKE/YOKE\_logging.py

**LOADING IP DATA**

"C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-IP\_DATA\_ALL.py"

**FORECASTING DATA**

import sys

import os

import sqlite3 as sq

import quandl

import pandas as pd

################################################################

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

print('Working Base :',Base, ' using ', sys.platform)

print('################################')

################################################################

Company='04-Clark'

sInputFileName='00-RawData/VKHCG\_Shares.csv'

sOutputFileName='Shares.csv'

################################################################

sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

################################################################

sFileDir1=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'

if not os.path.exists(sFileDir1):

os.makedirs(sFileDir1)

################################################################

sFileDir2=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'

if not os.path.exists(sFileDir2):

os.makedirs(sFileDir2)

################################################################

sFileDir3=Base + '/' + Company + '/03-Process/01-EDS/02-Python'

if not os.path.exists(sFileDir3):

os.makedirs(sFileDir3)

################################################################

sDatabaseName=sDataBaseDir + '/clark.db'

conn = sq.connect(sDatabaseName)

################################################################

### Import Share Names Data

################################################################

sFileName=Base + '/' + Company + '/' + sInputFileName

print('################################')

print('Loading :',sFileName)

print('################################')

RawData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1")

RawData.drop\_duplicates(subset=None, keep='first', inplace=True)

print('Rows :',RawData.shape[0])

print('Columns:',RawData.shape[1])

print('################')

################################################################

sFileName=sFileDir1 + '/Retrieve\_' + sOutputFileName

print('################################')

print('Storing :', sFileName)

print('################################')

RawData.to\_csv(sFileName, index = False)

print('################################')

################################################################

sFileName=sFileDir2 + '/Assess\_' + sOutputFileName

print('################################')

print('Storing :', sFileName)

print('################################')

RawData.to\_csv(sFileName, index = False)

print('################################')

################################################################

sFileName=sFileDir3 + '/Process\_' + sOutputFileName

print('################################')

print('Storing :', sFileName)

print('################################')

RawData.to\_csv(sFileName, index = False)

print('################################')

################################################################

### Import Shares Data Details

################################################################

nShares=RawData.shape[0]

#nShares=6

for sShare in range(nShares):

sShareName=str(RawData['Shares'][sShare])

ShareData = quandl.get(sShareName)

UnitsOwn=RawData['Units'][sShare]

ShareData['UnitsOwn']=ShareData.apply(lambda row:(UnitsOwn),axis=1)

ShareData['ShareCode']=ShareData.apply(lambda row:(sShareName),axis=1)

print('################')

print('Share :',sShareName)

print('Rows :',ShareData.shape[0])

print('Columns:',ShareData.shape[1])

print('################')

#################################################################

print('################')

sTable=str(RawData['sTable'][sShare])

print('Storing :',sDatabaseName,' Table:',sTable)

ShareData.to\_sql(sTable, conn, if\_exists="replace")

print('################')

################################################################

sOutputFileName = sTable.replace("/","-") + '.csv'

sFileName=sFileDir1 + '/Retrieve\_' + sOutputFileName

print('################################')

print('Storing :', sFileName)

print('################################')

ShareData.to\_csv(sFileName, index = False)

print('################################')

################################################################

sOutputFileName = sTable.replace("/","-") + '.csv'

sFileName=sFileDir2 + '/Assess\_' + sOutputFileName

print('################################')

print('Storing :', sFileName)

print('################################')

ShareData.to\_csv(sFileName, index = False)

print('################################')

################################################################

sOutputFileName = sTable.replace("/","-") + '.csv'

sFileName=sFileDir3 + '/Process\_' + sOutputFileName

print('################################')

print('Storing :', sFileName)

print('################################')

ShareData.to\_csv(sFileName, index = False)

print('################################')

################################################################

################################################################

print('### Done!! ############################################')

################################################################

NETWORK ROUTING

"C:\VKHCG\01-Vermeulen\02-Assess\Assess-Network-Routing-Company.py"

**LINEAR REGRESSION**

import sys

import os

import pandas as pd

import sqlite3 as sq

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear\_model

from sklearn.metrics import mean\_squared\_error, r2\_score

################################################################

Base='C:/VKHCG'

print('################################')

print('Working Base :',Base, ' using ', sys.platform)

print('################################')

################################################################

################################################################

Company='01-Vermeulen'

################################################################

sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

################################################################

sDatabaseName=sDataBaseDir + '/Vermeulen.db'

conn1 = sq.connect(sDatabaseName)

################################################################

sDataVaultDir=Base + '/88-DV'

if not os.path.exists(sDataVaultDir):

os.makedirs(sDataVaultDir)

################################################################

sDatabaseName=sDataVaultDir + '/datavault.db'

conn2 = sq.connect(sDatabaseName)

################################################################

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir):

os.makedirs(sDataWarehouseDir)

################################################################

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'

conn3 = sq.connect(sDatabaseName)

################################################################

t=0

tMax=((300-100)/10)\*((300-30)/5)

for heightSelect in range(100,300,10):

for weightSelect in range(30,300,5):

height = round(heightSelect/100,3)

weight = int(weightSelect)

bmi = weight/(height\*height)

if bmi <= 18.5:

BMI\_Result=1

elif bmi > 18.5 and bmi < 25:

BMI\_Result=2

elif bmi > 25 and bmi < 30:

BMI\_Result=3

elif bmi > 30:

BMI\_Result=4

else:

BMI\_Result=0

PersonLine=[('PersonID', [str(t)]),

('Height', [height]),

('Weight', [weight]),

('bmi', [bmi]),

('Indicator', [BMI\_Result])]

t+=1

print('Row:',t,'of',tMax)

if t==1:

PersonFrame = pd.DataFrame.from\_dict(PersonLine)

else:

PersonRow = pd.DataFrame.from\_dict(PersonLine)

PersonFrame = PersonFrame.append(PersonRow)

################################################################

DimPerson=PersonFrame

DimPersonIndex=DimPerson.set\_index(['PersonID'],inplace=False)

################################################################

sTable = 'Transform-BMI'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn1, if\_exists="replace")

################################################################

################################################################

sTable = 'Person-Satellite-BMI'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace")

################################################################

################################################################

sTable = 'Dim-BMI'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn3, if\_exists="replace")

################################################################

fig = plt.figure()

PlotPerson=DimPerson[DimPerson['Indicator']==1]

x=PlotPerson['Height']

y=PlotPerson['Weight']

plt.plot(x, y, ".")

PlotPerson=DimPerson[DimPerson['Indicator']==2]

x=PlotPerson['Height']

y=PlotPerson['Weight']

plt.plot(x, y, "o")

PlotPerson=DimPerson[DimPerson['Indicator']==3]

x=PlotPerson['Height']

y=PlotPerson['Weight']

plt.plot(x, y, "+")

PlotPerson=DimPerson[DimPerson['Indicator']==4]

x=PlotPerson['Height']

y=PlotPerson['Weight']

plt.plot(x, y, "^")

plt.axis('tight')

plt.title("BMI Curve")

plt.xlabel("Height(meters)")

plt.ylabel("Weight(kg)")

plt.plot()

# Load the diabetes dataset

diabetes = datasets.load\_diabetes()

# Use only one feature

diabetes\_X = diabetes.data[:, np.newaxis, 2]

diabetes\_X\_train = diabetes\_X[:-30]

diabetes\_X\_test = diabetes\_X[-50:]

diabetes\_y\_train = diabetes.target[:-30]

diabetes\_y\_test = diabetes.target[-50:]

regr = linear\_model.LinearRegression()

regr.fit(diabetes\_X\_train, diabetes\_y\_train)

diabetes\_y\_pred = regr.predict(diabetes\_X\_test)

print('Coefficients: \n', regr.coef\_)

print("Mean squared error: %.2f"

% mean\_squared\_error(diabetes\_y\_test, diabetes\_y\_pred))

print('Variance score: %.2f' % r2\_score(diabetes\_y\_test, diabetes\_y\_pred))

plt.scatter(diabetes\_X\_test, diabetes\_y\_test, color='black')

plt.plot(diabetes\_X\_test, diabetes\_y\_pred, color='blue', linewidth=3)

plt.xticks(())

plt.yticks(())

plt.axis('tight')

plt.title("Diabetes")

plt.xlabel("BMI")

plt.ylabel("Age")

plt.show()

**http://services.odata.org/V3/Northwind/Northwind.svc/**