# These are important to call in the beginning of the query since they are our tool sets

# Numpy= It is used to integrate a wide variety of database types

# Pandas= This is used for importing, analyzing, and manipulating datasets

import numpy as np

import pandas as pd

# This reads the "Trade\_Table" into Python

trade = pd.read\_csv('Trade\_Table.csv')

# This allows you to view the table we labled "trade"

Trade

# How to select only certain columns/variables to keep

trade\_select = trade[['Pin','Tin']]

trade\_select

# This allows you to see a method of applying multiple filters simultaneously

trade[(trade['Cond\_Filter\_1']>0) & (trade['Cond\_Filter\_2']>='A') & (trade['Cond\_Filter\_2']=='C')]

# Here we make use of a filter and name the output "trade2"

trade2= trade[trade['Cond\_Filter\_1']>0]

trade2

# Here we upload into the kernel the credit score table

score = pd.read\_csv('Score\_Table.csv')

# This creates a new table "full\_table" in which the "score" table is left joined onto "trade2" by Pins

full\_table= pd.merge(trade2,score, how='left', on=['Pin'])

# This shows how to figure out what 'type' of variables each is within the "full\_table" table

full\_table.dtypes

# How to substring Type A1 in Python this is called "Slicing" and we are removing the first part of the balance number

full\_table\_substring= full\_table

full\_table\_substring['intbal'] = full\_table['Bal'].astype(str).str[1:]

full\_table\_substring

# Here it creates a new conditional variable called Debt Type

#Similar to a "Case When" or "If Then" Statement in SQL

full\_table.loc[(trade['Type']=='M1'), 'Debt\_Type']='Mortgage'

full\_table.loc[(trade['Type']=='A1'), 'Debt\_Type']='Auto'

full\_table.loc[(trade['Type']=='C1'), 'Debt\_Type']='CreditCard'

# A 2nd method of relabeling them is the following using a Numpy "Where" statement

full\_table['Type1']= np.where(full\_table['Type']=='M1', 'Mortgage',

np.where(full\_table['Type']=='C1','CreditCard',

np.where(full\_table['Type']=='A1', 'Auto','Other' )))

full\_table

# Using Groupby to perform aggregate function by Debt Type but in a new variable then showing SUM function

bydebt = full\_table.groupby('Debt\_Type')

bydebt.sum()

# If you do not need to make a new variable you can use these

full\_table.groupby('Debt\_Type').sum()

# This one shows sum for a specific Debt Type

full\_table.groupby('Debt\_Type').sum().loc['Auto']

# Here are some other functions which can be used

#Count

full\_table.groupby('Debt\_Type').count()

# Standard Deviation

full\_table.groupby('Debt\_Type').std()

# Mean

full\_table.groupby('Debt\_Type').mean()

# Max

full\_table.groupby('Debt\_Type').max()

# Describe

full\_table.groupby('Debt\_Type').describe()

# Describe in different format

full\_table.groupby('Debt\_Type').describe().transpose()

# Converting String to Integer

s= "1234"

i= int(s)

print (i+1)

# Getting Substring(Slicing)

s[2:]

# Another method to see the variable type within a table

type(full\_table['Type1'])

# How to calculate date diff between 2 dates already in YYYY-MM-DD format

from datetime import date

def diff\_dates(date1, date2):

return abs(date2-date1).days

def main():

d1 = date(2013,1,1)

d2 = date(2013,9,13)

result1 = diff\_dates(d2, d1)

print ('{} days between {} and {}'.format(result1, d1, d2))

print ("Happy Holidays!")

main()

#To compensate for Leap Years you can use this.

from datetime import datetime

from calendar import isleap

start\_date = datetime(2005,4,28,12,33)

end\_date = datetime(2010,5,5,23,14)

diffyears = end\_date.year - start\_date.year

difference = end\_date - start\_date.replace(end\_date.year)

days\_in\_year = isleap(end\_date.year) and 366 or 365

difference\_in\_years = diffyears + (difference.days + difference.seconds/86400.0)/days\_in\_year