**Lab 4 Submission**

**Data Wrangler Script and Screenshot: CMSC**

from wrangler import dw

import sys

if(len(sys.argv) < 3):

sys.exit('Error: Please include an input and output file. Example python script.py input.csv output.csv')

w = dw.DataWrangler()

# Split data repeatedly on newline into rows

w.add(dw.Split(column=["data"],

table=0,

status="active",

drop=True,

result="row",

update=False,

insert\_position="right",

row=None,

on="\n",

before=None,

after=None,

ignore\_between=None,

which=1,

max=0,

positions=None,

quote\_character=None))

# Extract from data on 'CMSC any number '

w.add(dw.Extract(column=["data"],

table=0,

status="active",

drop=False,

result="column",

update=False,

insert\_position="right",

row=None,

on="CMSC\\d+",

before=None,

after=None,

ignore\_between=None,

which=1,

max=1,

positions=None))

# Fill extract with values from above

w.add(dw.Fill(column=["extract"],

table=0,

status="active",

drop=False,

direction="down",

method="copy",

row=None))

# Wrap rows where data starts with '0'

w.add(dw.Wrap(column=[],

table=0,

status="active",

drop=False,

row=dw.Row(column=[],

table=0,

status="active",

drop=False,

conditions=[dw.StartsWith(column=[],

table=0,

status="active",

drop=False,

lcol="data",

value="0",

op\_str="starts with")])))

# Delete row 1

w.add(dw.Filter(column=[],

table=0,

status="active",

drop=False,

row=dw.Row(column=[],

table=0,

status="active",

drop=False,

conditions=[dw.RowIndex(column=[],

table=0,

status="active",

drop=False,

indices=[0])])))

# Drop wrap3

w.add(dw.Drop(column=["wrap3"],

table=0,

status="active",

drop=True))

# Drop wrap13, wrap12, wrap11

w.add(dw.Drop(column=["wrap13","wrap12","wrap11"],

table=0,

status="active",

drop=True))

# Drop wrap5, wrap7, wrap9

w.add(dw.Drop(column=["wrap5","wrap7","wrap9"],

table=0,

status="active",

drop=True))

# Extract from wrap4 between ': ' and ','

w.add(dw.Extract(column=["wrap4"],

table=0,

status="active",

drop=False,

result="column",

update=False,

insert\_position="right",

row=None,

on=".\*",

before=",",

after=": ",

ignore\_between=None,

which=1,

max=1,

positions=None))

# Extract from wrap4 between 'Open: ' and ','

w.add(dw.Extract(column=["wrap4"],

table=0,

status="active",

drop=False,

result="column",

update=False,

insert\_position="right",

row=None,

on=".\*",

before=",",

after="Open: ",

ignore\_between=None,

which=1,

max=1,

positions=None))

# Extract from wrap4 between 'Waitlist: ' and ')'

w.add(dw.Extract(column=["wrap4"],

table=0,

status="active",

drop=False,

result="column",

update=False,

insert\_position="right",

row=None,

on=".\*",

before="\\)",

after="Waitlist: ",

ignore\_between=None,

which=1,

max=1,

positions=None))

# Drop wrap4

w.add(dw.Drop(column=["wrap4"],

table=0,

status="active",

drop=True))

# Extract from wrap6 before ' '

w.add(dw.Extract(column=["wrap6"],

table=0,

status="active",

drop=False,

result="column",

update=False,

insert\_position="right",

row=None,

on=".\*",

before=" ",

after=None,

ignore\_between=None,

which=1,

max=1,

positions=None))

# Extract from wrap6 after ' '

w.add(dw.Extract(column=["wrap6"],

table=0,

status="active",

drop=False,

result="column",

update=False,

insert\_position="right",

row=None,

on=".\*",

before=None,

after=" ",

ignore\_between=None,

which=1,

max=1,

positions=None))

# Drop wrap6

w.add(dw.Drop(column=["wrap6"],

table=0,

status="active",

drop=True))

# Extract from wrap8 before ' '

w.add(dw.Extract(column=["wrap8"],

table=0,

status="active",

drop=False,

result="column",

update=False,

insert\_position="right",

row=None,

on=".\*",

before=" ",

after=None,

ignore\_between=None,

which=1,

max=1,

positions=None))

# Extract from wrap8 after ' '

w.add(dw.Extract(column=["wrap8"],

table=0,

status="active",

drop=False,

result="column",

update=False,

insert\_position="right",

row=None,

on=".\*",

before=None,

after=" ",

ignore\_between=None,

which=1,

max=1,

positions=None))

# Drop wrap8

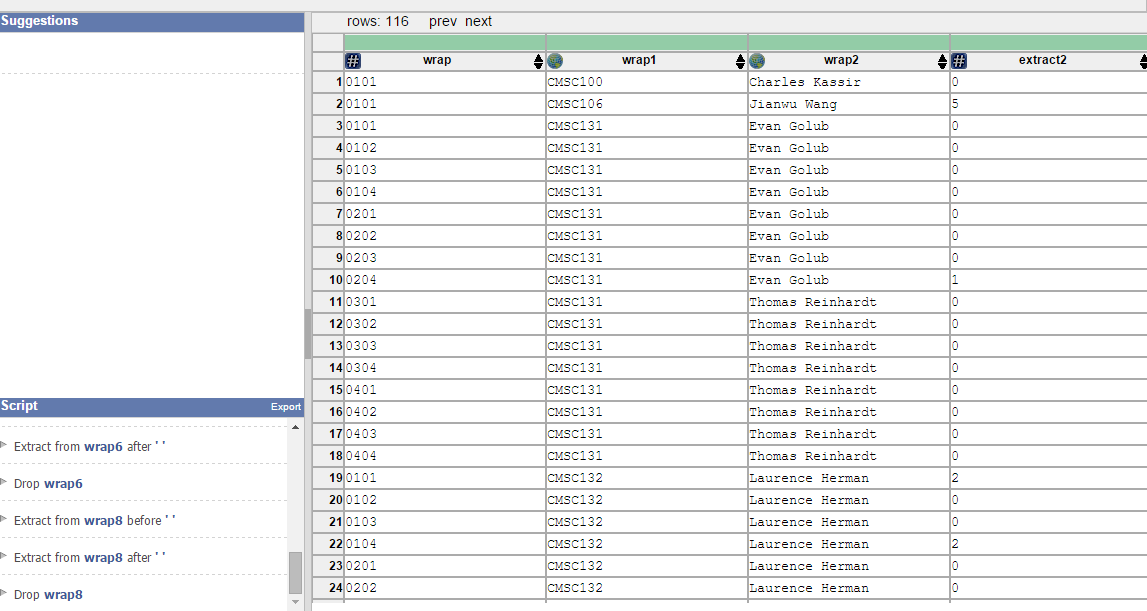
w.add(dw.Drop(column=["wrap8"],

table=0,

status="active",

drop=True))

w.apply\_to\_file(sys.argv[1]).print\_csv(sys.argv[2])



**Data Wrangler Script and Screenshot: World Cup 1**

**from wrangler import dw**

**import sys**

**if(len(sys.argv) < 3):**

**sys.exit('Error: Please include an input and output file. Example python script.py input.csv output.csv')**

**w = dw.DataWrangler()**

**# Split data repeatedly on newline into rows**

**w.add(dw.Split(column=["data"],**

**table=0,**

**status="active",**

**drop=True,**

**result="row",**

**update=False,**

**insert\_position="right",**

**row=None,**

**on="\n",**

**before=None,**

**after=None,**

**ignore\_between=None,**

**which=1,**

**max=0,**

**positions=None,**

**quote\_character=None))**

**# Delete row 1**

**w.add(dw.Filter(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**row=dw.Row(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**conditions=[dw.RowIndex(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**indices=[0])])))**

**# Wrap rows where data = '|-'**

**w.add(dw.Wrap(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**row=dw.Row(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**conditions=[dw.Eq(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**lcol="data",**

**value="|-",**

**op\_str="=")])))**

**# Extract from wrap1 on ' any uppercase word '**

**w.add(dw.Extract(column=["wrap1"],**

**table=0,**

**status="active",**

**drop=False,**

**result="column",**

**update=False,**

**insert\_position="right",**

**row=None,**

**on="[A-Z]+",**

**before=None,**

**after=None,**

**ignore\_between=None,**

**which=1,**

**max=1,**

**positions=None))**

**# Drop wrap1**

**w.add(dw.Drop(column=["wrap1"],**

**table=0,**

**status="active",**

**drop=True))**

**# Drop wrap**

**w.add(dw.Drop(column=["wrap"],**

**table=0,**

**status="active",**

**drop=True))**

**# Set wrap2 name to 1**

**w.add(dw.SetName(column=["wrap2"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["1"],**

**header\_row=None))**

**# Set wrap3 name to 2**

**w.add(dw.SetName(column=["wrap3"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["2"],**

**header\_row=None))**

**# Set wrap4 name to 3**

**w.add(dw.SetName(column=["wrap4"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["3"],**

**header\_row=None))**

**# Set wrap5 name to 4**

**w.add(dw.SetName(column=["wrap5"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["4"],**

**header\_row=None))**

**# Drop wrap6**

**w.add(dw.Drop(column=["wrap6"],**

**table=0,**

**status="active",**

**drop=True))**

**# Drop wrap7**

**w.add(dw.Drop(column=["wrap7"],**

**table=0,**

**status="active",**

**drop=True))**

**# Fold 1, 2, 3, 4 using header as a key**

**w.add(dw.Fold(column=["\_1","\_2","\_3","\_4"],**

**table=0,**

**status="active",**

**drop=False,**

**keys=[-1]))**

**# Split value repeatedly on ',' into rows**

**w.add(dw.Split(column=["value"],**

**table=0,**

**status="active",**

**drop=True,**

**result="row",**

**update=False,**

**insert\_position="right",**

**row=None,**

**on=",",**

**before=None,**

**after=None,**

**ignore\_between=None,**

**which=1,**

**max="0",**

**positions=None,**

**quote\_character=None))**

**# Extract from value between ' any word any word |' and ']'**

**w.add(dw.Extract(column=["value"],**

**table=0,**

**status="active",**

**drop=False,**

**result="column",**

**update=False,**

**insert\_position="right",**

**row=None,**

**on=".\*",**

**before="]",**

**after="[a-zA-Z]+[a-zA-Z]+\\|",**

**ignore\_between=None,**

**which=1,**

**max=1,**

**positions=None))**

**# Drop value**

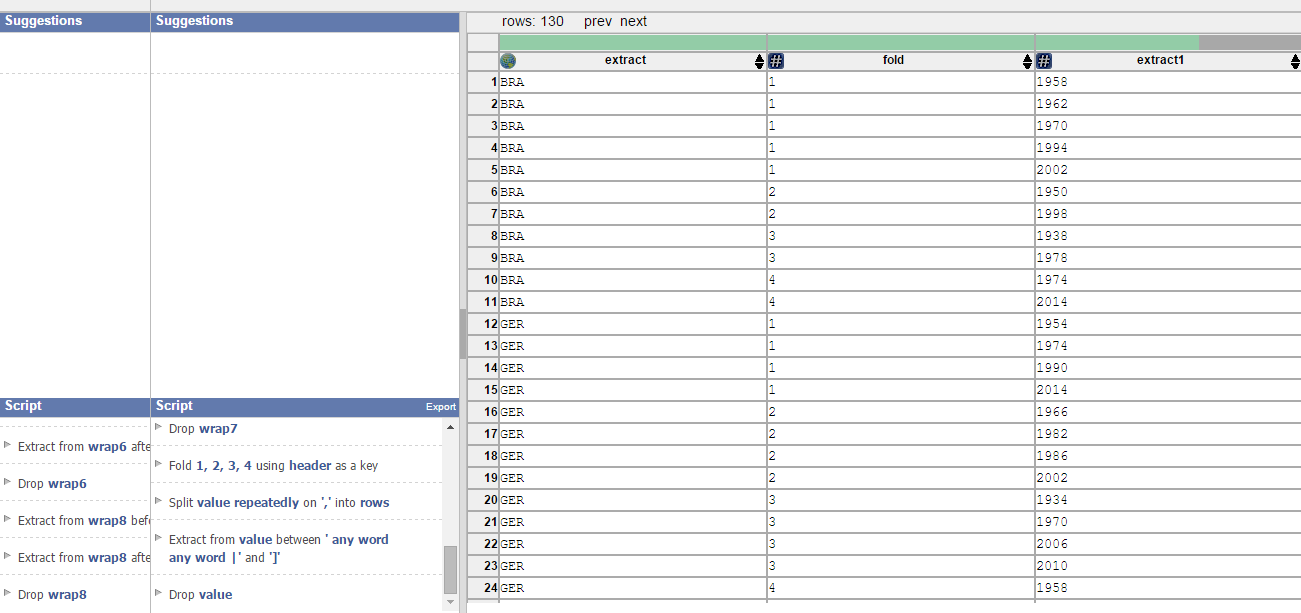
**w.add(dw.Drop(column=["value"],**

**table=0,**

**status="active",**

**drop=True))**

**w.apply\_to\_file(sys.argv[1]).print\_csv(sys.argv[2])**



**Data Wrangler Script and Screenshot: World Cup 2**

**from wrangler import dw**

**import sys**

**if(len(sys.argv) < 3):**

**sys.exit('Error: Please include an input and output file. Example python script.py input.csv output.csv')**

**w = dw.DataWrangler()**

**# Split data repeatedly on newline into rows**

**w.add(dw.Split(column=["data"],**

**table=0,**

**status="active",**

**drop=True,**

**result="row",**

**update=False,**

**insert\_position="right",**

**row=None,**

**on="\n",**

**before=None,**

**after=None,**

**ignore\_between=None,**

**which=1,**

**max=0,**

**positions=None,**

**quote\_character=None))**

**# Delete row 1**

**w.add(dw.Filter(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**row=dw.Row(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**conditions=[dw.RowIndex(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**indices=[0])])))**

**# Wrap rows where data = '|-'**

**w.add(dw.Wrap(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**row=dw.Row(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**conditions=[dw.Eq(column=[],**

**table=0,**

**status="active",**

**drop=False,**

**lcol="data",**

**value="|-",**

**op\_str="=")])))**

**# Extract from wrap1 on ' any uppercase word '**

**w.add(dw.Extract(column=["wrap1"],**

**table=0,**

**status="active",**

**drop=False,**

**result="column",**

**update=False,**

**insert\_position="right",**

**row=None,**

**on="[A-Z]+",**

**before=None,**

**after=None,**

**ignore\_between=None,**

**which=1,**

**max=1,**

**positions=None))**

**# Drop wrap1**

**w.add(dw.Drop(column=["wrap1"],**

**table=0,**

**status="active",**

**drop=True))**

**# Drop wrap**

**w.add(dw.Drop(column=["wrap"],**

**table=0,**

**status="active",**

**drop=True))**

**# Set wrap2 name to 1**

**w.add(dw.SetName(column=["wrap2"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["1"],**

**header\_row=None))**

**# Set wrap3 name to 2**

**w.add(dw.SetName(column=["wrap3"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["2"],**

**header\_row=None))**

**# Set wrap4 name to 3**

**w.add(dw.SetName(column=["wrap4"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["3"],**

**header\_row=None))**

**# Set wrap5 name to 4**

**w.add(dw.SetName(column=["wrap5"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["4"],**

**header\_row=None))**

**# Drop wrap6**

**w.add(dw.Drop(column=["wrap6"],**

**table=0,**

**status="active",**

**drop=True))**

**# Drop wrap7**

**w.add(dw.Drop(column=["wrap7"],**

**table=0,**

**status="active",**

**drop=True))**

**# Fold 1, 2, 3, 4 using header as a key**

**w.add(dw.Fold(column=["\_1","\_2","\_3","\_4"],**

**table=0,**

**status="active",**

**drop=False,**

**keys=[-1]))**

**# Split value repeatedly on ',' into rows**

**w.add(dw.Split(column=["value"],**

**table=0,**

**status="active",**

**drop=True,**

**result="row",**

**update=False,**

**insert\_position="right",**

**row=None,**

**on=",",**

**before=None,**

**after=None,**

**ignore\_between=None,**

**which=1,**

**max="0",**

**positions=None,**

**quote\_character=None))**

**# Extract from value between ' any word any word |' and ']'**

**w.add(dw.Extract(column=["value"],**

**table=0,**

**status="active",**

**drop=False,**

**result="column",**

**update=False,**

**insert\_position="right",**

**row=None,**

**on=".\*",**

**before="]",**

**after="[a-zA-Z]+[a-zA-Z]+\\|",**

**ignore\_between=None,**

**which=1,**

**max=1,**

**positions=None))**

**# Drop value**

**w.add(dw.Drop(column=["value"],**

**table=0,**

**status="active",**

**drop=True))**

**# Set fold name to place**

**w.add(dw.SetName(column=["fold"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["place"],**

**header\_row=None))**

**# Set extract1 name to year**

**w.add(dw.SetName(column=["extract1"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["year"],**

**header\_row=None))**

**# Set extract name to country**

**w.add(dw.SetName(column=["extract"],**

**table=0,**

**status="active",**

**drop=True,**

**names=["country"],**

**header\_row=None))**

**# Unfold year, place on place**

**w.add(dw.Unfold(column=["year","place"],**

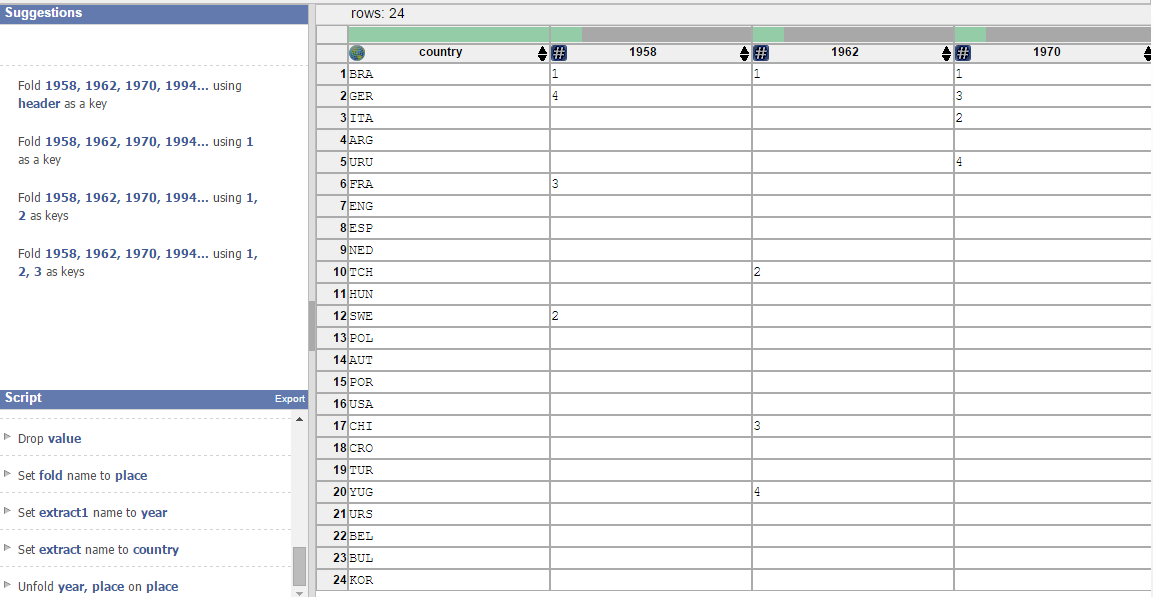
**table=0,**

**status="active",**

**drop=False,**

**measure="place"))**

**w.apply\_to\_file(sys.argv[1]).print\_csv(sys.argv[2])**



**UNIX Tools Command: CMSC**

**#!/bin/bash**

**cat cmsc.txt | awk '/^CMSC[0-9]\*$/ {class=$0}**

**/^0/ {print rest; rest = class", "$0}**

**!/^0|^CMSC[0-9]\*$/ {rest = rest", "$0}' |**

**sed 's/Seats (Total://; s/Open: //; s/Waitlist: \([0-9]\*\))/\1/; s/\(M\*Tu\*W\*Th\*F\*\)/\1,/; s/CSI\|ITV\|HBK\|AVW\|MTH\|JMP/\0,/; s/, $//'**

**UNIX Tools Command: World Cup 1**

#!/bin/bash

cat worldcup.txt | sed 's/^! Team.\*//; s/^.\*fb|\([A-Z]\{3\}\).\*/\1/' |

awk '/^\|-/ {print info; info = $0}

!/^\|-/ {info = info", "$0;}

END {print info}' | sed 's/^|-, //;' |

awk -F', \\|' '/^.\*$/ {

print $1", "$2", "$3", "$4", "$5;

}'

couldn't figure out this one, currently I have it printing each country with its 1st, 2nd, 3rd, and 4th place in the same line followed by commas

**Python Script: CMSC**

**import re**

**cmsc = open('cmsc.txt','r')**

**p = re.compile('^CMSC[0-9]+')**

**m = re.compile('^0[0-9]+')**

**t = re.compile('Seats \(Total: ([0-9]+), Open: ([0-9]+), Waitlist: ([0-9]+)\)')**

**d = re.compile('(M|Tu|W|Th|F)\* (.\*)$')**

**b = re.compile('(CSI|ITV|HBK|AVW|MTH|JMP) ([0-9]\*)$')**

**n = re.compile('.\*')**

**info = ""**

**classname = ""**

**for line in cmsc:**

**if p.match(line):**

**classname = p.match(line).group()**

**elif m.match(line):**

**print info**

**info = classname + ", " + m.match(line).group()**

**elif t.match(line):**

**matchgroup = t.match(line)**

**info += ", " + matchgroup.group(1) + ", " + matchgroup.group(2) + ", " + matchgroup.group(3)**

**elif d.match(line):**

**matchgroup = d.match(line)**

**info += ", " + matchgroup.group(1) + ", " + matchgroup.group(2)**

**elif b.match(line):**

**matchgroup = b.match(line)**

**info += ", " + matchgroup.group(1) + ", " + matchgroup.group(2)**

**else:**

**info += ", " + n.match(line).group()**

**CMSC100, 0101, Charles Kassir, 45, 4, 0, M, 4:00pm - 4:50pm, CSI, 2117,**

**CMSC106, 0101, Jianwu Wang, 45, 0, 5, Th, 9:30am - 10:45am, CSI, 2117,**

**CMSC131, 0101, Evan Golub , 31, 0, 0, F, 2:00pm - 2:50pm, CSI, 1115**

**CMSC131, 0102, Evan Golub , 31, 0, 0, F, 2:00pm - 2:50pm, CSI, 1115**

**CMSC131, 0103, Evan Golub , 31, 1, 0, F, 2:00pm - 2:50pm, CSI, 1115**

**CMSC131, 0104, Evan Golub , 31, 1, 0, F, 2:00pm - 2:50pm, CSI, 1115**

**CMSC131, 0201, Evan Golub , 31, 1, 0, F, 3:00pm - 3:50pm, CSI, 1115**

**CMSC131, 0202, Evan Golub , 31, 1, 0, F, 3:00pm - 3:50pm, CSI, 1115**

**CMSC131, 0203, Evan Golub , 31, 0, 0, F, 3:00pm - 3:50pm, CSI, 1115**

**CMSC131, 0204, Evan Golub , 31, 0, 1, F, 3:00pm - 3:50pm, CSI, 1115**

**CMSC131, 0301, Thomas Reinhardt, 31, 0, 0, F, 11:00am - 11:50am, CSI, 1115**

**CMSC131, 0302, Thomas Reinhardt, 31, 0, 0, F, 11:00am - 11:50am, CSI, 1115**

**CMSC131, 0303, Thomas Reinhardt, 31, 0, 0, F, 11:00am - 11:50am, CSI, 1115**

**CMSC131, 0304, Thomas Reinhardt, 31, 0, 0, F, 11:00am - 11:50am, CSI, 1115**

**CMSC131, 0401, Thomas Reinhardt, 31, 0, 0, F, 12:00pm - 12:50pm, CSI, 1115**

**CMSC131, 0402, Thomas Reinhardt, 31, 7, 0, F, 12:00pm - 12:50pm, CSI, 1115**

**CMSC131, 0403, Thomas Reinhardt, 31, 0, 0, F, 12:00pm - 12:50pm, CSI, 1115**

**CMSC131, 0404, Thomas Reinhardt, 31, 7, 0, F, 12:00pm - 12:50pm, CSI, 1115,**

**CMSC132, 0101, Laurence Herman , 34, 0, 2, F, 10:00am - 10:50am, CSI, 1115**

**CMSC132, 0102, Laurence Herman , 34, 0, 0, F, 10:00am - 10:50am, CSI, 1115**

**CMSC132, 0103, Laurence Herman , 34, 0, 0, F, 10:00am - 10:50am, CSI, 1115**

**CMSC132, 0104, Laurence Herman , 34, 0, 2, F, 10:00am - 10:50am, CSI, 1115**

**CMSC132, 0201, Laurence Herman , 34, 6, 0, F, 1:00pm - 1:50pm, CSI, 1115**

**CMSC132, 0202, Laurence Herman , 34, 1, 0, F, 1:00pm - 1:50pm, CSI, 1115**

**CMSC132, 0203, Laurence Herman , 34, 0, 0, F, 1:00pm - 1:50pm, CSI, 1115**

**CMSC132, 0204, Laurence Herman , 34, 0, 0, F, 1:00pm - 1:50pm, CSI, 1115**

**CMSC132, 0301, Laurence Herman , 30, 3, 0, F, 2:00pm - 2:50pm, CSI, 2117**

**CMSC132, 0302, Laurence Herman , 29, 0, 1, F, 2:00pm - 2:50pm, CSI, 2117**

**CMSC132, 0303, Laurence Herman , 29, 8, 0, F, 2:00pm - 2:50pm, CSI, 2117,**

**CMSC216, 0101, Nelson Padua-Perez , 28, 2, 0, Th, 9:30am - 10:45am, CSI, 1115**

**CMSC216, 0102, Nelson Padua-Perez , 28, 0, 0, Th, 9:30am - 10:45am, CSI, 1115**

**CMSC216, 0103, Nelson Padua-Perez , 28, 0, 1, Th, 9:30am - 10:45am, CSI, 1115**

**CMSC216, 0104, Nelson Padua-Perez , 28, 0, 0, Th, 9:30am - 10:45am, CSI, 1115**

**CMSC216, 0201, Nelson Padua-Perez , 28, 0, 0, Th, 11:00am - 12:15pm, CSI, 1115**

**CMSC216, 0202, Nelson Padua-Perez , 28, 0, 1, Th, 11:00am - 12:15pm, CSI, 1115**

**CMSC216, 0203, Nelson Padua-Perez , 28, 0, 1, Th, 11:00am - 12:15pm, CSI, 1115**

**CMSC216, 0204, Nelson Padua-Perez , 28, 0, 0, Th, 11:00am - 12:15pm, CSI, 1115**

**CMSC216, 0301, Nelson Padua-Perez , 28, 0, 0, Th, 2:00pm - 3:15pm, CSI, 1115**

**CMSC216, 0302, Nelson Padua-Perez , 28, 2, 0, Th, 2:00pm - 3:15pm, CSI, 1115**

**CMSC216, 0303, Nelson Padua-Perez , 28, 0, 0, Th, 2:00pm - 3:15pm, CSI, 1115**

**CMSC216, 0304, Nelson Padua-Perez , 28, 0, 0, Th, 2:00pm - 3:15pm, CSI, 1115,**

**CMSC250, 0101, Clyde Kruskal , 29, 0, 4, Th, 2:00pm - 3:15pm, CSI, 2117**

**CMSC250, 0102, Clyde Kruskal , 29, 0, 5, Th, 2:00pm - 3:15pm, CSI, 2117**

**CMSC250, 0103, Clyde Kruskal , 29, 0, 3, Th, 2:00pm - 3:15pm, CSI, 2117**

**CMSC250, 0201, Clyde Kruskal , 29, 0, 5, Th, 3:30pm - 4:45pm, CSI, 1115**

**CMSC250, 0202, Clyde Kruskal , 29, 0, 3, Th, 3:30pm - 4:45pm, CSI, 1115**

**CMSC250, 0203, Clyde Kruskal , 29, 0, 3, Th, 3:30pm - 4:45pm, CSI, 1115**

**CMSC250, 0204, Clyde Kruskal , 29, 14, 0, Th, 3:30pm - 4:45pm, CSI, 1115**

**CMSC250, 0301, Thomas Reinhardt, 29, 0, 1, Th, 11:00am - 12:15pm, CSI, 3117**

**CMSC250, 0302, Thomas Reinhardt, 29, 1, 0, Th, 11:00am - 12:15pm, CSI, 3117**

**CMSC250, 0303, Thomas Reinhardt, 29, 0, 5, Th, 11:00am - 12:15pm, CSI, 3117,**

**CMSC250, 0101, Thomas Goldstein, 25, 0, 0, Th, 12:30pm - 1:45pm, CSI, 3118,**

**CMSC289, 0101, James Reggia , 60, 0, 36, Th, 11:00am - 12:15pm, CSI, 2117,**

**CMSC330, 0101, Chau-Wen Tseng , 27, 0, 0, Th, 3:30pm - 4:45pm, CSI, 3117**

**CMSC330, 0102, Chau-Wen Tseng , 27, 0, 0, Th, 3:30pm - 4:45pm, CSI, 3117**

**CMSC330, 0103, Chau-Wen Tseng , 27, 1, 0, Th, 3:30pm - 4:45pm, CSI, 3117**

**CMSC330, 0201, Chau-Wen Tseng , 27, 0, 0, Th, 12:30pm - 1:45pm, CSI, 3117**

**CMSC330, 0202, Chau-Wen Tseng , 27, 0, 1, Th, 12:30pm - 1:45pm, CSI, 3117**

**CMSC330, 0203, Chau-Wen Tseng , 27, 13, 0, Th, 12:30pm - 1:45pm, CSI, 3117**

**CMSC330, 0301, Chau-Wen Tseng , 27, 0, 0, Th, 2:00pm - 3:15pm, CSI, 3117**

**CMSC330, 0302, Chau-Wen Tseng , 27, 0, 0, Th, 2:00pm - 3:15pm, CSI, 3117**

**CMSC330, 0303, Chau-Wen Tseng , 27, 1, 0, Th, 2:00pm - 3:15pm, CSI, 3117,**

**CMSC351, 0101, Hamid Mahini, 88, 0, 2, F, 10:00am - 10:50am, CSI, 3117**

**CMSC351, 0201, Hamid Mahini, 88, 0, 4, F, 11:00am - 11:50am, CSI, 3117**

**CMSC351, 0301, Hamid Mahini, 88, 0, 0, F, 3:00pm - 3:50pm, CSI, 2117,**

**CMSC396, 0101, Atif Memon , Neil Spring, 25, 0, 0, W, 1:00pm - 1:50pm, AVW, 4172,**

**CMSC411, 0101, Michelle Hugue , 45, 0, 3, Th, 3:30pm - 4:45pm, CSI, 1122**

**CMSC411, 0201, Michelle Hugue , 45, 0, 3, Th, 2:00pm - 3:15pm, CSI, 1122,**

**CMSC412, 0101, Neil Spring, 25, 0, 4, Th, 11:00am - 12:15pm, CSI, 1122**

**CMSC412, 0102, Neil Spring, 25, 0, 3, Th, 11:00am - 12:15pm, CSI, 1122,**

**CMSC414, 0101, A.U. Shankar , 40, 0, 6, Th, 12:30pm - 1:45pm, CSI, 1122**

**CMSC414, 0201, Elaine Shi, 50, 1, 0, W, 3:30pm - 4:45pm, CSI, 1122,**

**CMSC417, 0101, Ashok Agrawala , 40, 5, 0, Th, 11:00am - 12:15pm, CSI, 3120**

**CMSC417, 0201, Instructor: TBA, 40, 27, 0, Th, 3:30pm - 4:45pm, CSI, 2107,**

**CMSC420, 0101, Hanan Samet , 40, 0, 0, Th, 12:30pm - 1:45pm, CSI, 2120**

**CMSC420, 0201, Michelle Hugue , 40, 0, 10, W, 3:30pm - 4:45pm, CSI, 3120,**

**CMSC421, 0101, Donald Perlis , 50, 0, 2, Th, 2:00pm - 3:15pm, CSI, 1121**

**CMSC421, 0201, Donald Perlis , 46, 0, 3, Th, 9:30am - 10:45am, CSI, 3117,**

**CMSC423, 0101, Hector Corrada Bravo, 48, 20, 0, W, 3:30pm - 4:45pm, CSI, 1121,**

**CMSC424, 0101, Nicholas Roussopoulos , 40, 0, 2, Th, 2:00pm - 3:15pm, CSI, 3120,**

**CMSC426, 0101, John Aloimonos, 40, 7, 0, Th, 11:00am - 12:15pm, CSI, 2120,**

**CMSC427, 0101, Zia Khan, 40, 2, 0, W, 2:00pm - 3:15pm, CSI, 1121,**

**CMSC430, 0101, Jeffrey Foster , 40, 1, 0, W, 3:30pm - 4:45pm, CSI, 2107,**

**CMSC433, 0101, Michael Hicks , 50, 0, 15, Th, 3:30pm - 4:45pm, CSI, 1121,**

**CMSC434, 0101, Jon Froehlich, 44, 0, 8, Th, 9:30am - 10:45am, CSI, 1122**

**CMSC434, 0201, Vibha Sazawal, 44, 0, 6, Th, 11:00am - 12:15pm, CSI, 1121,**

**CMSC435, 0101, James Purtilo , 50, 0, 4, Th, 12:30pm - 1:45pm, CSI, 1121,**

**CMSC436, 0101, Adam Porter , 50, 0, 14, Th, 9:30am - 10:45am, CSI, 1121**

**CMSC436, 0201, Atif Memon , 60, 0, 11, Th, 12:30pm - 1:45pm, CSI, 2117,**

**CMSC451, 0101, Aravind Srinivasan, 40, 0, 1, Th, 12:30pm - 1:45pm, CSI, 3120,**

**CMSC452, 0101, William Gasarch , 32, 11, 0, Th, 2:00pm - 3:15pm, CSI, 3118,**

**CMSC456, 0101, Jonathan Katz, 40, 0, 3, F, 12:00pm - 12:50pm, CSI, 1121,**

**CMSC460, 0101, Harland Glaz , 33, 0, 2, Th, 12:30pm - 1:45pm, MTH, 0304**

**CMSC460, 0201, Changhui Tan, 25, 2, 0, Th, 2:00pm - 3:15pm, MTH, 0409,**

**CMSC466, 0101, Maria Cameron, 25, 11, 0, F, 11:00am - 11:50am, MTH, 1311,**

**CMSC474, 0101, Dana Nau , 40, 9, 0, Th, 3:30pm - 4:45pm, CSI, 2120,**

**CMSC498, 0101, Uzi Vishkin , 10, 7, 0, W, 11:00am - 12:15pm, ITV, 1100,**

**CMSC498, 0101, Amol Deshpande, 40, 8, 0, Th, 11:00am - 12:15pm, CSI, 2107,**

**CMSC498, 0101, Marshini Chetty , 10, 0, 4, Th, 6:00pm - 8:45pm, HBK, 0123,**

**CMSC631, 0101, David Van Horn, 20, 3, 0, Th, 3:30pm - 4:45pm, CSI, 3118,**

**CMSC660, 0101, Ramani Duraiswami, 25, 0, 3, Th, 3:30pm - 4:45pm, CSI, 3120,**

**CMSC663, 0101, Howard Elman, 20, 15, 0, Th, 5:00pm - 6:15pm, CSI, 4122,**

**CMSC666, 0101, Tobias von Petersdorff , 25, 8, 0, Th, 2:00pm - 3:15pm, CSI, 4122,**

**CMSC701, 0101, Mihai Pop, 40, 9, 0, Th, 9:30am - 10:45am, CSI, 3120,**

**CMSC712, 0101, A.U. Shankar , 20, 0, 0, Th, 2:00pm - 3:30pm, AVW, 3258,**

**CMSC723, 0101, Hal Daume, 40, 0, 10, Th, 2:00pm - 3:15pm, CSI, 2120,**

**CMSC733, 0101, David Jacobs , 32, 0, 17, Th, 11:00am - 12:15pm, CSI, 3118,**

**CMSC737, 0101, Atif Memon , 20, 0, 0, Th, 9:30am - 10:45am, CSI, 2118,**

**CMSC754, 0101, David Mount , 32, 0, 15, Th, 12:30pm - 1:45pm, CSI, 2118,**

**CMSC798, 0101, Jeffrey Foster , 40, 10, 0, F, 11:00am - 12:50pm, CSI, 2117,**

**CMSC818, 0101, Peter Keleher , 20, 1, 0, Th, 11:00am - 12:15pm, AVW, 3258,**

**CMSC818, 0101, Instructor: TBA, 145, 145, 0, W, 4:00pm - 5:00pm, CSI, 1115,**

**CMSC818, 0101, Tudor Dumitras, 15, 13, 0, W, 12:30pm - 1:45pm, JMP, 1202,**

**CMSC828, 0101, Larry Davis , 25, 2, 0, Th, 2:00pm - 3:15pm, CSI, 2118,**

**CMSC828, 0101, Venkatramanan Subrahmanian , 25, 12, 0, Th, 9:30am - 10:45am, CSI, 3118,**

**Python Script: World Cup 1**

import re

data = open('worldcup.txt','r')

results = open('pandas2.csv','w')

place = "1"

c = re.compile('.\*{{fb\|([A-Z]+)}}.\*')

y = re.compile('.\*Cup\|([0-9]+).\*')

country = ""

for line in data:

split = line.split(',')

for year in split:

if y.match(year):

results.write(country + ", " + y.match(year).group(1) + ", " + place + "\n")

if place == "1":

place = "2"

elif place == "2":

place = "3"

elif place == "3":

place = "4"

else:

place = "1"

if c.match(line):

country = c.match(line).group(1)

place = "1"

results.close()

BRA, 1958, 1

BRA, 1962, 1

BRA, 1970, 1

BRA, 1994, 1

BRA, 2002, 1

BRA, 1950, 2

BRA, 1998, 2

BRA, 1938, 3

BRA, 1978, 3

BRA, 1974, 4

BRA, 2014, 4

GER, 1954, 1

GER, 1974, 1

GER, 1990, 1

GER, 2014, 1

GER, 1966, 2

GER, 1982, 2

GER, 1986, 2

GER, 2002, 2

GER, 1934, 3

GER, 1970, 3

GER, 2006, 3

GER, 2010, 3

GER, 1958, 4

ITA, 1934, 1

ITA, 1938, 1

ITA, 1982, 1

ITA, 2006, 1

ITA, 1970, 2

ITA, 1994, 2

ITA, 1990, 3

ITA, 1978, 4

ARG, 1978, 1

ARG, 1986, 1

ARG, 1930, 2

ARG, 1990, 2

ARG, 2014, 2

URU, 1930, 1

URU, 1950, 1

URU, 1954, 4

URU, 1970, 4

URU, 2010, 4

FRA, 1998, 1

FRA, 2006, 2

FRA, 1958, 3

FRA, 1986, 3

FRA, 1982, 4

ENG, 1966, 1

ENG, 1990, 4

ESP, 2010, 1

ESP, 1950, 4

NED, 1974, 2

NED, 1978, 2

NED, 2010, 2

NED, 2014, 3

NED, 1998, 4

TCH, 1934, 2

TCH, 1962, 2

HUN, 1938, 2

HUN, 1954, 2

SWE, 1958, 2

SWE, 1950, 3

SWE, 1994, 3

SWE, 1938, 4

POL, 1974, 3

POL, 1982, 3

AUT, 1954, 3

AUT, 1934, 4

POR, 1966, 3

POR, 2006, 4

USA, 1930, 3

CHI, 1962, 3

CRO, 1998, 3

TUR, 2002, 3

YUG, 1930, 4

YUG, 1962, 4

URS, 1966, 4

BEL, 1986, 4

BUL, 1994, 4

KOR, 2002, 4

**Python Script: World Cup 2**

import re

import numpy

import pandas as pd

data = pd.read\_csv("pandas2.csv", names=['country', 'year', 'place'])

result = data.pivot('country','year','place')

print result

year 1930 1934 1938 1950 1954 1958 1962 1966 1970 1974 1978 \

country

ARG 2 NaN NaN NaN NaN NaN NaN NaN NaN NaN 1

AUT NaN 4 NaN NaN 3 NaN NaN NaN NaN NaN NaN

BEL NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN

BRA NaN NaN 3 2 NaN 1 1 NaN 1 4 3

BUL NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN

CHI NaN NaN NaN NaN NaN NaN 3 NaN NaN NaN NaN

CRO NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN

ENG NaN NaN NaN NaN NaN NaN NaN 1 NaN NaN NaN

ESP NaN NaN NaN 4 NaN NaN NaN NaN NaN NaN NaN

FRA NaN NaN NaN NaN NaN 3 NaN NaN NaN NaN NaN

GER NaN 3 NaN NaN 1 4 NaN 2 3 1 NaN

HUN NaN NaN 2 NaN 2 NaN NaN NaN NaN NaN NaN

ITA NaN 1 1 NaN NaN NaN NaN NaN 2 NaN 4

KOR NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN

NED NaN NaN NaN NaN NaN NaN NaN NaN NaN 2 2

POL NaN NaN NaN NaN NaN NaN NaN NaN NaN 3 NaN

POR NaN NaN NaN NaN NaN NaN NaN 3 NaN NaN NaN

SWE NaN NaN 4 3 NaN 2 NaN NaN NaN NaN NaN

TCH NaN 2 NaN NaN NaN NaN 2 NaN NaN NaN NaN

TUR NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN

URS NaN NaN NaN NaN NaN NaN NaN 4 NaN NaN NaN

URU 1 NaN NaN 1 4 NaN NaN NaN 4 NaN NaN

USA 3 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN

YUG 4 NaN NaN NaN NaN NaN 4 NaN NaN NaN NaN

year 1982 1986 1990 1994 1998 2002 2006 2010 2014

country

ARG NaN 1 2 NaN NaN NaN NaN NaN 2

AUT NaN NaN NaN NaN NaN NaN NaN NaN NaN

BEL NaN 4 NaN NaN NaN NaN NaN NaN NaN

BRA NaN NaN NaN 1 2 1 NaN NaN 4

BUL NaN NaN NaN 4 NaN NaN NaN NaN NaN

CHI NaN NaN NaN NaN NaN NaN NaN NaN NaN

CRO NaN NaN NaN NaN 3 NaN NaN NaN NaN

ENG NaN NaN 4 NaN NaN NaN NaN NaN NaN

ESP NaN NaN NaN NaN NaN NaN NaN 1 NaN

FRA 4 3 NaN NaN 1 NaN 2 NaN NaN

GER 2 2 1 NaN NaN 2 3 3 1

HUN NaN NaN NaN NaN NaN NaN NaN NaN NaN

ITA 1 NaN 3 2 NaN NaN 1 NaN NaN

KOR NaN NaN NaN NaN NaN 4 NaN NaN NaN

NED NaN NaN NaN NaN 4 NaN NaN 2 3

POL 3 NaN NaN NaN NaN NaN NaN NaN NaN

POR NaN NaN NaN NaN NaN NaN 4 NaN NaN

SWE NaN NaN NaN 3 NaN NaN NaN NaN NaN

TCH NaN NaN NaN NaN NaN NaN NaN NaN NaN

TUR NaN NaN NaN NaN NaN 3 NaN NaN NaN

URS NaN NaN NaN NaN NaN NaN NaN NaN NaN

URU NaN NaN NaN NaN NaN NaN NaN 4 NaN

USA NaN NaN NaN NaN NaN NaN NaN NaN NaN

YUG NaN NaN NaN NaN NaN NaN NaN NaN NaN

[24 rows x 20 columns]