import os.path

import time

import getpass

import json

import requests

from requests.auth import HTTPBasicAuth

import sys

import datetime as dt

import pandas as pd

start = 0

naptime = 15

b = 1

issues = []

jqlFilter = "60690"

jira = "https://issues.abc.com/rest/api/2/search"

while True:

query = "?jql=filter=" + jqlFilter + "&startAt=" + str(start) + "&maxResults=100"

result = requests.get(jira + query, auth=HTTPBasicAuth(user, password))

data = json.loads(result.text)

if not data["issues"]:

break

else:

issues += data["issues"]

time.sleep(naptime)

b += 1

start += 100

data["issues"] = issues

b=[];a=[];c=[];d=[];e=[];f=[];g=[];h=[];j=[];k=[];l=[];m=[];n=[];o=[];p=[];q=[];r=[];s=[]

w1=len(data["issues"])

import datetime

for i in range(0,w1):

b=data["issues"][i]['key']

c=data["issues"][i]["fields"]["status"]["name"]

e=data["issues"][i]["fields"]["issuetype"]["name"]

a.append(b)

d.append(c)

f.append(e)

g=data["issues"][i]["fields"]["resolution"]

if g is None:

h.append("none")

else:

g=data["issues"][i]["fields"]["resolution"]["name"]

h.append(g)

j=data["issues"][i]["fields"]["created"]

k.append(j)

l=data["issues"][i]["fields"]["resolutiondate"]

if l is None:

m.append("none")

else:

l=data["issues"][i]["fields"]["resolutiondate"]

m.append(l)

p=data["issues"][i]["fields"]["customfield\_11604"]

if p is None:

q.append("none")

else:

p=data["issues"][i]["fields"]["customfield\_11604"]

q.append(p)

r=data["issues"][i]["fields"]["duedate"]

if r is None:

s.append("none")

else:

r=data["issues"][i]["fields"]["duedate"]

s.append(r)

df = pd.DataFrame({'issue': a,'status':d,'issuetype': f,'resolution':h,'created':k,'resolvdate':m,'custombaselinestart':q,'duedate':s})

import datetime

date\_1=datetime.datetime.today().strftime('%Y-%m-%d')

df1 = df[(df.status != "Triage") & (df.status != "Backlog") & (df.issuetype != "Task") & (df.resolution=="none") ]

dailywip=df1['issue'].count() # daily wip total

df2 = df[(df.status == "Triage") | (df.status == "Backlog") & (df.issuetype != "Task")]

df\_11=df2.count()

daily\_queue=df\_11['issue'] # incoming queue

df\_wrike=pd.read\_excel('hi.xlsx')

df\_n=df.merge(df\_wrike,how='left',left\_on=df['issue'], right\_on=df\_wrike['Issue key'])

df\_n['Baseline Create Date'].fillna(df\_n['created'],inplace=True)

dfz= df\_n[(df\_n.status == "Backlog") & (df\_n.issuetype != "Task")]

dff=dfz['Baseline Create Date']

import numpy as np

mean = (np.array(dff, dtype='datetime64[s]')

.view('i8')

.mean()

.astype('datetime64[s]'))

hh=pd.to\_datetime(date\_1)-pd.to\_datetime(mean)

avgageq=hh.days #avgageq

dfa=pd.to\_datetime((dfz['Baseline Create Date']))

kj=(pd.to\_datetime(date\_1)-dfa).sum()

avgagequeue=kj.days/len(dfa)

j=df[(df.resolvdate =='none') & (df.issuetype != "Task")]

g=j[j['custombaselinestart'] !='none']

s=(pd.to\_datetime(date\_1)-pd.to\_datetime(g['custombaselinestart']))

pp=s.sum()

pp1=pp.days

avgagewip=pp1/len(s) #avgagewip

df1 = df[(df.resolution == "Done") & (df.issuetype != "Task")]

date\_2=datetime.date.fromordinal(datetime.date.today().toordinal()-1)

df111=df1[(pd.to\_datetime(df1['resolvdate'])<pd.to\_datetime(date\_1)) & (pd.to\_datetime(df1['resolvdate'])>pd.to\_datetime(date\_2))]

dailycomplete=df111.count().issue #dailycomplete

ts='9/27/2017'

jj=df[(df['issuetype']!='Subtask') & (df['issuetype']!='Epic') & (df.issuetype != "Task") & (pd.to\_datetime(df['created'])>=pd.to\_datetime(ts))]

ff1=jj[jj['custombaselinestart'] !='none']

jj1=df\_n[(df\_n['issuetype']!='Subtask') & (df.issuetype != "Task") & (df\_n['issuetype']!='Epic') & (pd.to\_datetime(df\_n['created'])>pd.to\_datetime(ts))]

ff3=jj1[jj1['custombaselinestart'] !='none']

a=pd.to\_datetime(ff1['custombaselinestart'])-pd.to\_datetime(ff3['Baseline Create Date'])

c=a.sum()

histavg\_q=c.days/len(a) #historical avg queue time

df45=df[df.resolvdate !='none'].custombaselinestart

dfggg=df45[df45!='none']

import numpy as np

mean1 = (np.array(dfggg, dtype='datetime64[s]')

.view('i8')

.mean()

.astype('datetime64[s]'))

df46= df[(df.resolvdate !='none')]

mean2 = (np.array(df46.resolvdate, dtype='datetime64[s]')

.view('i8')

.mean()

.astype('datetime64[s]'))

zz=pd.to\_datetime(mean2)-pd.to\_datetime(mean1)

avgwip\_cycletime=zz.days

#wip status breakdown

df3 = df[(df.status == "Investigation") & (df.issuetype != "Task")].count()

inv\_status=df3['issue'] #investigation status count

df4 = df[(df.status == "SUBLET: INVESTIGATION") & (df.issuetype != "Task")].count()

sub\_inv\_status=df4['issue']

df5 = df[(df.status == "In Development") & (df.issuetype != "Task")].count()

dev\_status=df5['issue']

df6 = df[(df.status == "SUBLET: APPROVAL") & (df.issuetype != "Task")].count()

sub\_dev\_status=df6['issue']

df7 = df[(df.status == "In Validation") & (df.issuetype != "Task")].count()

val\_status=df7['issue']

df8 = df[(df.status == "SUBLET: VALIDATION") & (df.issuetype != "Task")].count()

sub\_val\_status=df8['issue']

df9 = df[(df.status == "Product Manager Review") & (df.status == "In Review") ].count()

df11=df9

tech\_pub\_status=df11['issue']

df12 = df[(df.status == "Waiting for Documentation") & (df.issuetype != "Task")].count()

sublet\_status\_tech=df12['issue'] #sublet technical publications count

#wip type breakdown

df13 = df[(df.issuetype == "Repair Procedure Development") & (df.resolvdate =='none') & (df.status != "Triage") & (df.status != "Backlog")].count()

repproc\_type\_misc=df13['issue']

df14 = df[(df.issuetype == "Documentation") & (df.resolvdate =='none') & (df.status != "Triage") & (df.status != "Backlog")].count()

tech\_type\_misc=df14['issue']

df15 = df[(df.issuetype == "Tooling Request") & (df.resolvdate =='none') & (df.status != "Triage") & (df.status != "Backlog")].count()

tool\_type\_misc=df15['issue']

df16 = df[(df.issuetype == "Correction") & (df.resolvdate =='none') & (df.status != "Triage") & (df.status != "Backlog")].count()

syscorr\_type\_misc=df16['issue']

df17 = df[(df.issuetype == "Miscellaneous") & (df.resolvdate =='none') & (df.status != "Triage") & (df.status != "Backlog")].count()

wip\_type\_misc=df17['issue']

#queue type breakdown

df30 = df[(df.issuetype == "Repair Procedure Development") & (df.resolvdate =='none') & (df.status == "Triage")].count()

df31= df[(df.issuetype == "Repair Procedure Development") & (df.resolvdate =='none')& (df.status == "Backlog")].count()

todayq\_rep\_proc=df30+df31

todayq\_rep\_proc=todayq\_rep\_proc.issue

df32= df[(df.issuetype == "Documentation") & (df.resolvdate =='none')& (df.status == "Backlog")].count()

df33= df[(df.issuetype == "Documentation") & (df.resolvdate =='none')& (df.status == "Triage")].count()

todayq\_technotes=df32+df33

todayq\_technotes=todayq\_technotes.issue

df34= df[(df.issuetype == "Tooling Request") & (df.resolvdate =='none')& (df.status == "Backlog")].count()

df35= df[(df.issuetype == "Tooling Reques") & (df.resolvdate =='none')& (df.status == "Triage")].count()

todayq\_tooling=df34+df35

todayq\_tooling=todayq\_tooling.issue

df36= df[(df.issuetype == "Correction") & (df.duedate =='none')& (df.status == "Backlog")].count()

df37= df[(df.issuetype == "Correction") & (df.duedate =='none')& (df.status == "Triage")].count()

todayq\_sys\_corr=df36+df37

todayq\_sys\_corr=todayq\_sys\_corr.issue

df38= df[(df.issuetype == "Miscellaneous") & (df.resolvdate =='none')& (df.status == "Backlog")].count()

df39= df[(df.issuetype == "Miscellaneous") & (df.resolvdate =='none')& (df.status == "Triage")].count()

todayq\_misc=df38+df39

todayq\_misc=todayq\_misc.issue

df40= df[(df.issuetype == "Work Request") & (df.resolvdate =='none')& (df.status == "Backlog")].count()

df41= df[(df.issuetype == "Work Request") & (df.resolvdate =='none')& (df.status == "Triage")].count()

todayq\_wrkreq=df40+df41

todayq\_wrkreq=todayq\_wrkreq.issue

df\_daily=pd.DataFrame(

{'date\_1': date\_1,'sublet\_status\_tech':sublet\_status\_tech,'tech\_pub\_status': tech\_pub\_status,'sub\_val\_status':sub\_val\_status,'val\_status':val\_status,'sub\_dev\_status':sub\_dev\_status,'dev\_status':dev\_status,'sub\_inv\_status':sub\_inv\_status,'inv\_status':inv\_status,'wip\_type\_misc':wip\_type\_misc,'syscorr\_type\_misc':syscorr\_type\_misc,'tool\_type\_misc':tool\_type\_misc,'tech\_type\_misc':tech\_type\_misc,'repproc\_type\_misc':repproc\_type\_misc,'todayq\_misc':todayq\_misc, 'todayq\_sys\_corr':todayq\_sys\_corr,'todayq\_tooling':todayq\_tooling,'todayq\_technotes':todayq\_technotes ,'todayq\_rep\_proc':todayq\_rep\_proc ,'todayq\_wrkreq':todayq\_wrkreq},index=[0])

import pymysql

dbPassword = "WRSFE3432FWRDF3@#"

dbName = "rshinyapps"

connectionObject = pymysql.connect(host=dbServerName, user=dbUser, password=dbPassword,

db=dbName)

cursorObject = connectionObject.cursor()

cursorObject.execute("insert into abc\_Historical VALUES('%s', '%s', '%s', '%s','%s', '%s', '%s', '%s','%s', '%s', '%s', '%s','%s','%s','%s','%s')" % \

(date\_2, daily\_queue , dailywip, avgageq,avgagewip,histavg\_q,avgwip\_cycletime,dailycomplete,inv\_status,sub\_inv\_status,dev\_status,sub\_dev\_status,val\_status,sub\_val\_status,tech\_pub\_status,sublet\_status\_tech))

connectionObject.commit()

import sqlalchemy

month=(pd.to\_datetime(df\_n[df\_n.issuetype != "Task"]['Baseline Create Date'])).dt.month

year=(pd.to\_datetime(df\_n['Baseline Create Date'])).dt.year

gg=df\_n['Baseline Create Date'][pd.to\_datetime(df\_n['Baseline Create Date']).dt.year>=2018].groupby([year.rename('yearr'),month.rename('monthh')]).agg('count')

df\_m=df\_n[df\_n['resolvdate']!='none']

dfff=df\_m[(df\_m['issuetype'] != "Task") & (df\_m['resolution']=="Done")]

month1=(pd.to\_datetime(dfff['resolvdate'])).dt.month

year1=(pd.to\_datetime(dfff['resolvdate'])).dt.year

jj=dfff['resolvdate'][pd.to\_datetime(dfff['resolvdate']).dt.year>=2018].groupby([year1.rename('yearrr'),month1.rename('monthhh')]).agg('count')

d = {'submit' :gg,'resolve' : jj}

dfp = pd.DataFrame(d)

ds=dfp.reset\_index()

df\_daily.to\_sql(name=abc\_daily', con=engine,if\_exists='replace', index=True,index\_label=None, chunksize=None)

ds.to\_sql(name=abc, con=engine,if\_exists='replace', index=True,index\_label=None, chunksize=None)

Issues faced:;

You can prevent future connections:

REVOKE CONNECT ON DATABASE thedb FROM public;

(and possibly other users/roles; see \l+ in psql)

You can then terminate all connections to this db except your own:

SELECT pid, pg\_terminate\_backend(pid)

FROM pg\_stat\_activity

WHERE datname = current\_database() AND pid <> pg\_backend\_pid();