import unittest  
import warnings  
  
from flask import Flask  
from pandas import pandas as pd  
from sqlalchemy import create\_engine  
from sqlalchemy.orm import sessionmaker  
  
from lib.s3\_file\_helper import S3FileHelper  
  
from models.deal\_setup import DealSetupRepo  
from services.listing\_file\_generator import ListingFileGenerator  
from services.valuation\_model\_factory import ValuationModelFactory  
from models.earning\_trends\_repo import EarningTrendsRepo  
from models.base\_repo import db  
  
  
class TestWizFiles(unittest.TestCase):  
    config = {}  
  
    dburi = xdb'  
    def setUp(self):  
  
        self.config['S3\_BUCKET'] = "valuation-statements-dev"  
        self.config['S3\_KEY'] = "AKIAJY73E7QSIG2M2PNQ"  
        self.config['S3\_SECRET'] = "7rLVLqVyzQg2d4fcGrII0HS+OH+0Mehdu/qbf6Fk"  
        self.config['SQLALCHEMY\_DATABASE\_URI'] = self.dburi  
        self.config['S3\_REGION\_LOCATION'] = '<https://s3-us-west-2.amazonaws.com/>'.format(self.config['S3\_BUCKET'])  
  
        self.engine = create\_engine(  
            self.dburi,  
            echo=False)  
  
        self.ascap\_mismatched\_region\_deal\_slug = 'test2985'  
        self.ascap\_matched\_region\_deal\_slug = '313428110'  
        self.bmi\_deal\_slug = '316119818'  
  
        warnings.filterwarnings("ignore", category=ResourceWarning, message="unclosed.\*<ssl.SSLSocket.\*>")  
  
    def tearDown(self):  
        pass  
  
    # @unittest.skip("skipping test\_ascap mismatched")  
    def test\_ascap\_mismatched\_region\_files(self):  
        print("ascap mismatched region tests")  
        self.check\_wiz\_files(self.ascap\_mismatched\_region\_deal\_slug)  
  
    # @unittest.skip("skipping test\_ascap matched")  
    def test\_ascap\_matched\_region\_files(self):  
        print("ascap matched region tests")  
        self.check\_wiz\_files(self.ascap\_matched\_region\_deal\_slug)  
  
    # @unittest.skip("skipping test bmi")  
    def test\_bmi\_files(self):  
        print("bmi tests")  
        self.check\_wiz\_files(self.bmi\_deal\_slug, True)  
  
    def check\_wiz\_files(self, dealid\_slug, skip\_ascap=False):  
  
        Session = sessionmaker(bind=self.engine)  
        session = Session()  
        dealsetup =  session.query(DealSetupRepo).filter(DealSetupRepo.deal\_id == dealid\_slug).first()  
  
  
        if dealsetup is None:  
            raise FileNotFoundError("Could not find deal")  
  
        if dealsetup.catalog\_pct is None:  
            dealsetup.catalog\_pct = 100  
  
  
        app = Flask(\_\_name\_\_)  
        app.config.from\_object(self.config)  
        with app.app\_context():  
            db.init\_app(app)  
            self.generate\_wiz\_files(self.config, dealid\_slug, dealsetup)  
  
  
        print("1   Top Songs Pie Chart – Lifetime")  
        self.check\_top\_songs\_lifetime(dealid\_slug, dealsetup)  
  
        print("2   Top Songs Pie Chart – LTM")  
        self.check\_top\_songs\_ltm(dealid\_slug, dealsetup)  
  
        print("3   Top Sources Pie Chart – Lifetime")  
        self.check\_top\_sources\_lifetime(dealid\_slug, dealsetup)  
  
       print("4   Top Sources Pie Chart – LTM")  
        self.check\_top\_sources\_ltm(dealid\_slug, dealsetup)  
  
        print("5   Quarterly Bar Chart")  
        self.check\_earnings\_quarters\_life\_time(dealid\_slug, dealsetup)  
  
        print("6   Quarterly Bar Chart – Last 12 Quarters")  
        # self.check\_earnings\_last\_12\_quarters\_life\_time(dealid\_slug, dealid)  
  
        print("7   Cashflow - skipping")  
        # No need to test  
  
        print("8   Raw Data for Listing")  
        self.check\_eligible\_raw\_data\_life\_time(dealid\_slug, dealsetup)  
  
        print("9   Top Songs Table")  
        self.check\_top\_5\_earning\_songs\_ltm(dealid\_slug, dealsetup)  
  
        print("10  Top Sources Table")  
        self.check\_top\_5\_earning\_sources\_ltm(dealid\_slug, dealsetup)  
  
        print("11  Comparison of Top Sources Table and Top Sources Pie Chart – LTM Files")  
        self.compare\_top\_5\_earning\_sources\_ltm\_with\_topsources\_ltm(dealid\_slug, dealsetup)  
  
        if skip\_ascap:  
            print("11  ASCAP Last 4 Quarters Table - Skipping since bmi file")  
        else:  
            print("11  ASCAP Last 4 Quarters Table")  
            self.check\_earnings\_last\_4\_quarters\_ascap\_ltm(dealid\_slug, dealsetup)  
  
  
    def check\_top\_songs\_ltm(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'topsongs\_ltm.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_ltm = df['Amount'].sum()  
        self.assertAlmostEqual(total\_ltm, self.get\_trends\_page\_ltm(dealid\_slug, dealsetup), msg='topsongs\_ltm.csv')  
  
    def check\_top\_sources\_ltm(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'topsources\_ltm.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_ltm = df['Amount'].sum()  
        self.assertAlmostEqual(total\_ltm, self.get\_trends\_page\_ltm(dealid\_slug, dealsetup), msg='topsongs\_ltm.csv')  
  
    def check\_top\_songs\_lifetime(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'topsongs.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_lifetime = df['Amount'].sum()  
        self.assertAlmostEqual(total\_lifetime, self.get\_trends\_page\_lifetime(dealid\_slug, dealsetup), msg='topsongs.csv')  
  
    def check\_top\_sources\_lifetime(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'topsources.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_lifetime = df['Amount'].sum()  
        self.assertAlmostEqual(total\_lifetime, self.get\_trends\_page\_lifetime(dealid\_slug, dealsetup), msg='topsources.csv')  
  
    def check\_earnings\_quarters\_life\_time(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'earnings\_by\_quarter.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_lifetime = df['Amount'].sum()  
        self.assertAlmostEqual(total\_lifetime, self.get\_trends\_page\_lifetime(dealid\_slug, dealsetup), msg='earnings\_by\_quarter.csv')  
  
    def check\_eligible\_raw\_data\_life\_time(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'raw\_data\_for\_listing.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_lifetime = df['amount'].sum()  
        self.assertAlmostEqual(total\_lifetime, self.get\_trends\_page\_lifetime(dealid\_slug, dealsetup), msg='raw\_data\_for\_listing.csv')  
  
    def check\_top\_5\_earning\_songs\_ltm(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'top\_5\_earning\_songs.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_ltm = df['LAST 12 MONTHS'].apply(lambda x: float(x.split()[0].replace(',', '').replace('$', ''))).sum()  
        self.assertNotEqual(total\_ltm, 0)  
  
    def check\_top\_5\_earning\_sources\_ltm(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'top\_5\_earning\_sources.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_ltm = df['LAST 12 MONTHS'].apply(lambda x: float(x.split()[0].replace(',', '').replace('$', ''))).sum()  
        self.assertAlmostEqual(total\_ltm, round(self.get\_trends\_page\_ltm(dealid\_slug, dealsetup)), msg='top\_5\_earning\_sources.csv')  
  
    def check\_earnings\_last\_4\_quarters\_ascap\_ltm(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'earnings\_last\_4\_quarters.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_ltm = df['TOTAL'].apply(lambda x: float(x.split()[0].replace(',', '').replace('$', ''))).sum()  
  
        # account for the extra "TOTAL" row  
        adjusted\_total\_ltm = total\_ltm / 2  
        self.assertAlmostEqual(adjusted\_total\_ltm, self.get\_trends\_page\_ltm(dealid\_slug, dealsetup), msg='earnings\_last\_4\_quarters.csv')  
  
    def check\_earnings\_last\_12\_quarters\_life\_time(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        fileKey = 'earnings\_last\_12\_quarters.csv'  
        df = self.load\_wiz\_file(fileKey, dealid\_slug)  
        total\_ltm = df['Amount'].iloc[-4:].sum()  
        self.assertAlmostEqual(total\_ltm, self.get\_trends\_page\_ltm(dealid\_slug, dealsetup), msg='earnings\_last\_12\_quarters.csv')  
  
    def compare\_top\_5\_earning\_sources\_ltm\_with\_topsources\_ltm(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
  
        top\_5\_sources\_file\_key = 'top\_5\_earning\_sources.csv'  
        top\_5\_sources\_df = self.load\_wiz\_file(top\_5\_sources\_file\_key, dealid\_slug)  
        top\_5\_sources\_ltm = top\_5\_sources\_df['LAST 12 MONTHS'].apply(lambda x: float(x.split()[0].replace(',', '').replace('$',''))).sum()  
  
        top\_sources\_ltm\_fileKey = 'topsources\_ltm.csv'  
        top\_sources\_ltm\_df = self.load\_wiz\_file(top\_sources\_ltm\_fileKey, dealid\_slug)  
        top\_sources\_ltm = top\_sources\_ltm\_df['Amount'].sum()  
  
        self.assertAlmostEqual(top\_5\_sources\_ltm, round(top\_sources\_ltm), msg='Comparison of top\_5\_earning\_sources.csv and topsources\_ltm.csv Files')  
  
  
    def load\_wiz\_file(self, fileKey, dealid\_slug):  
        s3 = S3FileHelper(self.config['S3\_KEY'], self.config['S3\_SECRET'])  
        wiz\_files = s3.get\_wiz\_files(self.config['S3\_BUCKET'], dealid\_slug)  
        for wiz\_file in wiz\_files:  
            key = wiz\_file['Key']  
            s3\_bucket\_url = self.config['S3\_REGION\_LOCATION'] + self.config['S3\_BUCKET'] + '/' + key  
            if fileKey in key:  
               return pd.read\_csv(s3\_bucket\_url)  
        return ''  
  
    def get\_trends\_page\_ltm(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        et\_result = self.get\_earning\_trends\_2(dealid\_slug, dealid, dealsetup)  
        snapshottrends = et\_result['snapshot\_trends']  
        return snapshottrends['ltm']  
  
    def get\_trends\_page\_lifetime(self, dealid\_slug, dealsetup):  
        dealid = [dealsetup.id](http://dealsetup.id/)  
        et\_result = self.get\_earning\_trends\_2(dealid\_slug, dealid, dealsetup)  
        snapshottrends = et\_result['snapshot\_trends']  
        return snapshottrends['lifetime\_amount']  
  
    def get\_earning\_trends\_2(self, dealid\_slug, dealid, dealsetup):  
        Session = sessionmaker(bind=self.engine)  
        session = Session()  
        catalog\_pct = 1  
  
        # ve = ValuationModelFactory(self.config['SQLALCHEMY\_DATABASE\_URI'])  
        # dollar\_age = ve.calculate\_dollar\_age(dealid)  
  
        et = session.query(EarningTrendsRepo).filter(EarningTrendsRepo.deal\_id == dealid) \  
            .order\_by(EarningTrendsRepo.dollar\_age\_bucket).all()  
        if et.

...