Question 1

```
transactionLog = [
    {"orderID":"100","customerId":"cust_Ahmed","ProductId":
["A9","B15",'A90',"C19"]},
    {"orderID":"101","customerId":"cust_Mishal","ProductId":
["C19","A9"]},
    {"orderID":"102","customerId":"cust_Muniba","ProductId":
["B54","A9","C19"]},
    {"orderID":"103","customerId":"cust_Faraz","ProductId":
["A90",'B15']}
]

productLog = {"A9":"Slingshot","B15":"China Tea Set","B54":"Paper
Towel","A90":"Bluetooth Fan","C19":"Chai"}

def processTransactions(dict_data):
  newDict = {}

  for custinfo in dict_data:
    newDict[custinfo['customerId']] = custinfo["ProductId"]
  return newDict

# this is data [['A9', 'B15', 'C19'], ['C19', 'A9'], ['B54', 'A9',
'C19'], 'A90']
from collections import Counter

def frequentPairs(transactionLog):
  custData = processTransactions(transactionLog)
  data = list(custData.values())

  pairs = []
  for prodlist in data:

    for i in range(len(prodlist)):
      maxInd = len(prodlist)
      for j in range(maxInd):
        if prodlist[i]!= prodlist[j]:
          pair = (prodlist[i],prodlist[j])

          pairs.append(str(sorted(pair)))




  count = dict(Counter(pairs))
  freq = [eval(key) for key,value in count.items() if value>=4]
  return freq

def reccomended(targetProd,freqPairs):
```

```
  rec=[]
  for pair in freqPairs:
    if targetProd in pair:
      rec.append(pair)
  return rec

freq  = frequentPairs(transactionLog)
print(freq)
reccomended("A9",freq)
```

```
[['A9', 'C19'], ['A90', 'B15']]
```

```
[['A9', 'C19']]
```

```
def genReport(targetId,catalog,freq):
  rec = reccomended(targetId,freq)

  for id,name in catalog.items():
    if catalog[targetId] == catalog[rec[0][0]]:
      print(f'-Your Purchase: {targetId}:{catalog[targetId]}\n-
Recommended Products:  {catalog[rec[0][1]]}')

      return

genReport('A9',productLog,freq)
```

```
-Your Purchase: A9:Slingshot
-Recommended Products:  Chai
```

Question 2

```
allPosts = [{'id':1,'text':'Using #Gulphone % was a crazy experience.
it$$ was so good that it cured my cornea and now I can walk again'},
 {'id':2,'text':'@gulphone has to be the best in market like
seriously, when &i walked in the store i could feel its aura'},
 {'id':3,'text':"Even if you paid me a million ++dollars to leave a
decent review; I'd never take it.thats how bad this phone is!
#disgusting"},
 {'id':4,'text':'my daughter bought this for her 9th ----birthday and
now its her 50th. lasted for decades ^-^! amazing phone ngl'},
 {'id':5,'text':'@GulAli The quality was  =disastrous! never buying
again :('},
 {'id':6,'text':"@Mishoo I got mine from a secondhand source it
was ,,,the worst phone% i've *ever had"}]

punctuationChar = "!$%^&*()_+-=,./][:]{};'`~|"

stopWords =
{'i','a','me','my','this','that','an','is','am','was','and','but','if'
,'or','to','of','at','by','for','with','this','that'}

posWords = {'good','best','amazing','cool'}

negWords = {'bad','disastrous','horrible','worst'}
```

TASKS

1. Clean Text

```
#assume we give the string as input
def processText(text,punct = punctuationChar,stopWordsSet=stopWords):
  for char in punct:
    if char in text:
      text = text.replace(char,"")

  splitText = text.split()
  for word in stopWords:
    newText = [a for a in splitText if a not in stopWordsSet]
    text = " ".join(newText)
  return text
```

2. Score Sentiment

```
def analyzePosts(postsList=allPosts, posWords=posWords,
negWords=negWords):
    allText = [post['text'] for post in postsList]
    processedText = list(map(processText, allText))
    newList = postsList
```

```
    for index, postDict in enumerate(newList):
        postDict['text'] = processedText[index]
        text = postDict['text']
        score = 0

        for word in posWords:
            if word in text:
                score += 1

        for word in negWords:
            if word in text:
                score -= 1

        postDict['score'] = score

    return newList
```

3. Flag Posts

```
def getFlaggedPosts(postsList = allPosts,sentimentThreshold = -1):
  postsList = analyzePosts(postsList)
  flaggedPosts = [post for post in postsList if post['score'] == -1]
  return flaggedPosts
```

4. Find Topics

```
def findNegTopics(flaggedPosts):
  postText = [post['text'] for post in flaggedPosts]
  negTopics = []
  for text in postText:
    words = text.split()
    for word in words:
      if word.startswith('@') or word.startswith('#'):
        negTopics.append(word)

  return negTopics
```

test

```
analyzePosts()

[{'id': 1,
  'text': 'Using #Gulphone crazy experience it so good it cured cornea
now I can walk again',
  'score': 1},
 {'id': 2,
  'text': '@gulphone has be the best in market like seriously when
walked in the store could feel its aura',
```

```
  'score': 1},
 {'id': 3,
  'text': 'Even you paid million dollars leave decent review Id never
take itthats how bad phone #disgusting',
  'score': -1},
 {'id': 4,
  'text': 'daughter bought her 9th birthday now its her 50th lasted
decades amazing phone ngl',
  'score': 1},
 {'id': 5,
  'text': '@GulAli The quality disastrous never buying again',
  'score': -1},
 {'id': 6,
  'text': '@Mishoo I got mine from secondhand source it the worst
phone ive ever had',
  'score': -1}]

getFlaggedPosts(allPosts)

[{'id': 3,
  'text': 'Even you paid million dollars leave decent review Id never
take itthats how bad phone #disgusting',
  'score': -1},
 {'id': 5,
  'text': '@GulAli The quality disastrous never buying again',
  'score': -1},
 {'id': 6,
  'text': '@Mishoo I got mine from secondhand source it the worst
phone ive ever had',
  'score': -1}]

findNegTopics(getFlaggedPosts(allPosts))

['#disgusting', '@GulAli', '@Mishoo']
```

QUESTION 3

```
class Package:

  def __init__(self,packageId,weightKg):
    self.assigned = False
    self.packageId = packageId
    self.weightKg = weightKg
```

TASKS

```
Alldrones = []

class Drone:
  def __init__(self,droneId,maxLoadKg):
    self.droneId = droneId
    self.maxLoadKg = maxLoadKg
    self.__status = 'idle'
    Alldrones.append(self)

  def getStatus(self):
    return self.__status

  def setStatus(self,status):
    validStatus = ('idle','delivering','charging')
    if status.lower() in validStatus:
      self.__status = status
      print(f'-STATUS UPDATED TO {status}')
    else:
      return f'NOT A VALID STATUS'

  def assignPackage(self,Package):
    if Package.weightKg <= self.maxLoadKg and self.__status == 'idle':
      print(f'-PACKAGE ASSIGNED WITH ID:{Package.packageId}')
      Package.assigned = True
      self.setStatus('delivering')

    elif self.__status != 'idle':
      return f'-FAILED TO ASSIGN PACKAGE DRONE STATUS NOT IDLE'
    elif Package.weightKg > self.maxLoadKg:
      return f'-FAILED TO ASSIGN! MAX WEIGHT IS {self.maxLoadKg}'

import time

class FleetManager:
  def __init__(self):
    self.drones = []
    self.packages = []
```

```
    self.pendingPack = []

  def assignPackage(self,*Package):
    for pack in Package:
      self.packages.append(pack)
      print(f'-PACKAGE ASSIGNED ID:{pack.packageId}')
    return '-ALL PACKAGES ASSIGNED TO FLEET'

  def assignDrone(self,*Drone):
    for drone in Drone:
      self.drones.append(drone)
      print(f'-DRONE ASSIGNED ID:{drone.droneId}')
    return '-ALL DRONES ASSIGNED TO FLEET'

  def getPendingPackages(self):
    for package in self.packages:
      if not(package.assigned):
        self.pendingPack.append(package)


  def dispatchJobs(self):
    for drone in self.drones:
      if drone.getStatus() == 'idle':
        if len(self.pendingPack)==0:
          return f'-NO PENDING PACKAGES TO ASSIGN'
        else:
          drone.assignPackage(self.pendingPack[-1])
          self.pendingPack.pop()

  def simulationTick(self):
    for drone in self.drones:
      if drone.getStatus() == 'delivering':
        time.sleep(2)
        drone.setStatus('charging')
    return '-ALL STATUS UPDATED'
```

TEST

```
p1 = Package('12df',53)
p2 = Package('14df',43)
p3 = Package('16df',312)
p4 = Package('167df',23)
p5 = Package('12df',332)

d1 = Drone('aff13',214)
d2 = Drone('fsafa',214)
d3 = Drone('sfad',214)
d4 = Drone('asdasf',214)
d5 = Drone('sfasf',214)
```

```
d4.setStatus('charging')

d3.assignPackage(p2)

-STATUS UPDATED TO charging
-PACKAGE ASSIGNED WITH ID:14df
-STATUS UPDATED TO delivering

d5.setStatus('lol')

{"type":"string"}

manage = FleetManager()

manage.assignDrone(d1,d2,d3,d4)
manage.assignPackage(p1,p2,p3,p4)

-DRONE ASSIGNED ID:aff13
-DRONE ASSIGNED ID:fsafa
-DRONE ASSIGNED ID:sfad
-DRONE ASSIGNED ID:asdasf
-PACKAGE ASSIGNED ID:12df
-PACKAGE ASSIGNED ID:14df
-PACKAGE ASSIGNED ID:16df
-PACKAGE ASSIGNED ID:167df

{"type":"string"}

manage.drones

[<__main__.Drone at 0x7d1a5a318350>,
 <__main__.Drone at 0x7d1a5a318410>,
 <__main__.Drone at 0x7d1a5a318380>,
 <__main__.Drone at 0x7d1a5a318590>]

manage.packages

[<__main__.Package at 0x7d1a5a3193a0>,
 <__main__.Package at 0x7d1a5b7b83e0>,
 <__main__.Package at 0x7d1a5a3195b0>,
 <__main__.Package at 0x7d1a5a3195e0>]

manage.dispatchJobs()

{"type":"string"}

manage.getPendingPackages()

manage.dispatchJobs()

-PACKAGE ASSIGNED WITH ID:167df
-STATUS UPDATED TO delivering
```

```
manage.simulationTick()

-STATUS UPDATED TO charging
-STATUS UPDATED TO charging

{"type":"string"}
```

QUESTION 4

```
import numpy as np

def flipHorizontal(pixelData):
    return np.flip(pixelData,axis=0)

def rotate90(pixelData):
    return np.flip(np.transpose(pixelData),axis=0)

def transformationFuncs():
  return [flipHorizontal,rotate90,adjustBrightness]

def adjustBrightness(pixelData,brightVal=80):
  return (pixelData+brightVal)

class image:
  def __init__(self):
    self.imgPixel = np.random.randint(low=0,high=255,size=(2,3))

  def applyTransformation(self,transformationFunc):
    copyImg = self.makeCopy(self.imgPixel)
    for func in transformationFunc:
      func(copyImg)

    return copyImg

  def makeCopy(self):
    copyPix = self.imgPixel.copy()
    return copyPix


class AugmentationPipeline(image):
    def __init__(self):
        super().__init__()
        self.steps = []


    def addStep(self,func):
        self.steps.append(func)

    def processImage(self, originalImage):
        for func in self.steps:
            func(originalImage)
        return originalImage

img1 = image()

img1.imgPixel
```

```
array([[181, 108,  22],
       [133, 243, 116]])

img1 = image()

pipeline = AugmentationPipeline()
pipeline.addStep(flipHorizontal)
pipeline.addStep(rotate90)

results = pipeline.processImage(img1.imgPixel)


results

array([[233, 186, 193],
       [ 35, 225, 167]])
```