Lab 5 Challenges

1) Update the lambda function for sentiment analysis to use AWS Comprehend
    a. First, update the IAM **sentiments-access-policy** to include AWS **Comprehend**
        i. The ONLY action the Lambda function for Sentiments can perform on the **Comprehend** is the read action: **DetectSentiment.**
    b. Then, open the **lambda_function.py** under the **sentiment-analysis** folder.
    c. **ADD** the line below underneath the REGION variable declaration **(REGION="us-east-1"):**
        i. comprehend = boto3.client('comprehend', region_name=REGION)
    d. **DELETE** the section:
        i.
```
TERMS={}
sent_file = open('AFINN-111.txt')
sent_lines = sent_file.readlines()

for line in sent_lines:
        s = line.split("\t")

        TERMS[s[0]] = s[1].strip()

sent_file.close()
```
    e. Then, edit the **get_sentiment(text)** function by first deleting all lines of code in the function:
        i. **DELETE** these lines:
```
splitTweet = text.split()
sentiment = 0.0

for word in splitTweet:
        if word in TERMS:
                sentiment = sentiment+ float(TERMS[word])

return sentiment
```
    f. Then, add the lines of code below in the **get_sentiment(text)** function:
        i. ADD these lines:
```
sentValue = 0
sentiment = comprehend.detect_sentiment(Text=text, LanguageCode='en')
sentString = sentiment['Sentiment']
if sentString == 'POSITIVE':
    sentValue = 3
elif sentString == 'NEGATIVE':
    sentValue = -3
elif sentString == 'MIXED':
    sentValue = 1
else:
    sentValue = 0
return sentValue
```

g. Afterwards, compress all files in the sentiment-analysis file into a zip file again and re-upload the zip file to the **code-lastname-2021-ece4150** S3 bucket under the lambda folder.
h. Lastly, go to the sentiments-analysis lambda function and re-upload the Code source using the Amazon S3 Object URL for the sentiment-analysis.zip file.

2) Cost analysis of your application
    a. Lambda cost analysis:
        i. There are 2 lambda functions used in this application: sentiments-analysis function and kinesis-dynamo function.
        ii. Since both were 2048MB and both had an estimate time of 200ms per request or .2 seconds per request, I solved for both in a similar way, but the sentiment-analysis function was triggered after every 1 tweet and the kinesis-dynamo function was triggered after every 10 tweets.
            1. The total number of seconds in a month for 30 days is 2,592,000 seconds.
            2. To find the total estimate number of tweets per month, I divided the total seconds in a month by the time for one tweet:
                a. 2,592,000 sec / .2 sec = 12,960,000 tweets or 12.96M tweets
            3. Afterwards, I calculated the total number of requests/times the lambda function was triggered.
                a. Sentiment function
                    i. Since it is triggered after every 10 tweets, I divided the tweets by 10.
                    ii. 12.96M tweets / 10 tweets/request = 1.296M requests
                b. Kinesis-dynamo function
                    i. Since it is triggered after every 1 tweet, the total requests is 12.96M requests
        iii. First cost: Monthly request charges
            1. Sentiment function
                a. It cost $0.2 per 1 million requests
                b. 1.296M requests * $0.2/1M requests = **$0.26**
            2. Kinesis-dynamo function
                a. It cost $0.2 per 1 million requests
                b. 12.96M requests * $0.2/1M requests = **$2.59**
        iv. Second cost: monthly duration charges
            1. It cost $0.0000000333 / 1 ms
            2. 2,592,000 sec/month * 1000 ms/sec * $0.0000000333/ms = $86.31
            3. Since there are 2 functions: $86.31 * 2 = **$172.62**
        v. Total cost:
            1. Monthly request charges + monthly duration charges
            2. $0.26 + $2.59 + $172.62 = **$175.47**
    b. DynamoDB cost analysis:
        i. The application only uses one table.

ii. Based on the Tweet table, the total size of around 100 items is 50 KB, so I estimated the data size of each tweet to be .5 KB or 5*10^-7 GB.
iii. We estimated the total number of requests from the lambda function above to be around 12,960,000 requests which translates to the same number of items in the table and streams.
iv. First cost: DynamoDB Stream costs
    1. It cost $0.02 per 100,000 streams
    2. 12,960,000 streams / 100,000 streams = 129.6
    3. 129.6 * $0.02 = **$2.59**
v. Second cost: Data Storage costs
    1. It cost $0.25/GB
    2. Total size of all items: 12,960,000 items * 5*10^-7 GB = 6.48 GB
    3. 6.48 GB * $0.25/GB = **$1.62**
vi. Third cost: cost of writes
    1. It costs $1.25 per 1 million writes
    2. Total number of writes is made up of when both lambda functions write to the table, since one lambda function takes the tweet creates an entry in the table and then the other lambda function rewrites the entry after getting the sentiment value.
    3. Therefore, 12,960,000 items * 2 writes = 25,920,000 writes or 25.92M writes
    4. 25.92M writes * $1.25/1M writes = **$32.40**
vii. Fourth cost: cost of reads
    1. It costs $0.25 per 1 million reads
    2. Total number of reads is made up of when the sentiment-analysis lambda function reads the table entry to get the tweet for that entry and evaluate what the sentiment value is.
    3. Therefore, 12,960,000 items * 1 writes = 12,960,000 writes or 12.96M writes
    4. 12.96 M writes * $0.25/1M writes = **$3.24**
viii. Total cost
    1. DynamoDB stream costs + Data storage costs + cost of writes + cost of reads
    2. $2.59 + $1.62 + $32.40 +$3.24 = **$39.85**

c. EC2 cost analysis:
    i. The EC2 used for application was a Linux operating system and used a t2.micro instance
        1. The cost for this instance is $0.0116 / hour
    ii. Since we estimated a month to be 30 days, the total hours is 720 hours for a month.
    iii. 720 hours * $0.0116/hour = **$8.35**
    iv. Total cost
        1. **$8.35**
d. Kinesis data stream cost analysis:

i. The Kinesis data stream used only 1 shard
ii. It cost $0.015/hour for 1 shard.
iii. Each PUT payload unit is up to 25KB, since each record/tweet is about .5 KB estimated from the DynamoDB cost analysis. So each record/tweet is equal to 1 PUT payload unit.
iv. It cost $0.014 per 1 million PUT payload units
v. First cost: Shard hour
    1. Since a month with an estimate of 30 days is 720 hours / month
    2. 720 hours * $0.015/hour = **$10.80**
vi. Second cost: PUT payload unit
    1. Total PUT payload units = 12,960,000 tweets * 1 PUT payload units = 12,960,000 PUT payload units or 12.96M payload units
    2. 12.96M payload units * $0.014/1M payload units = **$0.18**
vii. Total cost
    1. Shard hour + PUT payload unit
    2. $10.80 + $0.18 = **$10.98**

e. Kinesis firehose cost analysis:
i. Since each record is around .5KB which was estimated in the DynamoDB cost analysis, rounding it to the closest multiple of 5KB is 5KB.
ii. The total records of streaming data per second is found by the estimate of the time it takes for each tweet which was .2 sec from the lambda cost analysis.
iii. Since each tweet takes around .2 seconds, the total records of streaming data per second is 5 records/second.
iv. It cost $0.029 per GB
v. The total data ingested (GB/sec) =
    (5 records/sec * 5KB/record) / 1,048,576 KB/GB = 2.38*10^-5 GB/sec
vi. The total data ingested (GB/month) =
    2.38*10^-5 GB/sec * 86,400 sec/day * 30 days/month = 61.80 GB/month
vii. 61.80 GB/month * $0.029/GB = **$1.79**
viii. Total cost:
    1. **$1.79**

f. Elasticsearch cost analysis:
i. The instance used was a t2.small.elasticsearch instance which costs $0.036/hour
ii. Since it uses a EBS storage, it costs $0.135/GB for 1 instance for the General Purpose SSD storage.
iii. First cost: instance per hour cost
    1. Since 1 month has an estimate of 30 days, the total number of hours per month is 720 hours
    2. 720 hours * $0.036/hour = **$25.92**
iv. Second cost: EBS storage cost
    1. There is only 1 instance used and 30 GB of storage for the EBS General Purpose SSD Storage
    2. 30GB * $0.135/GB * 1 instance = **$4.05**

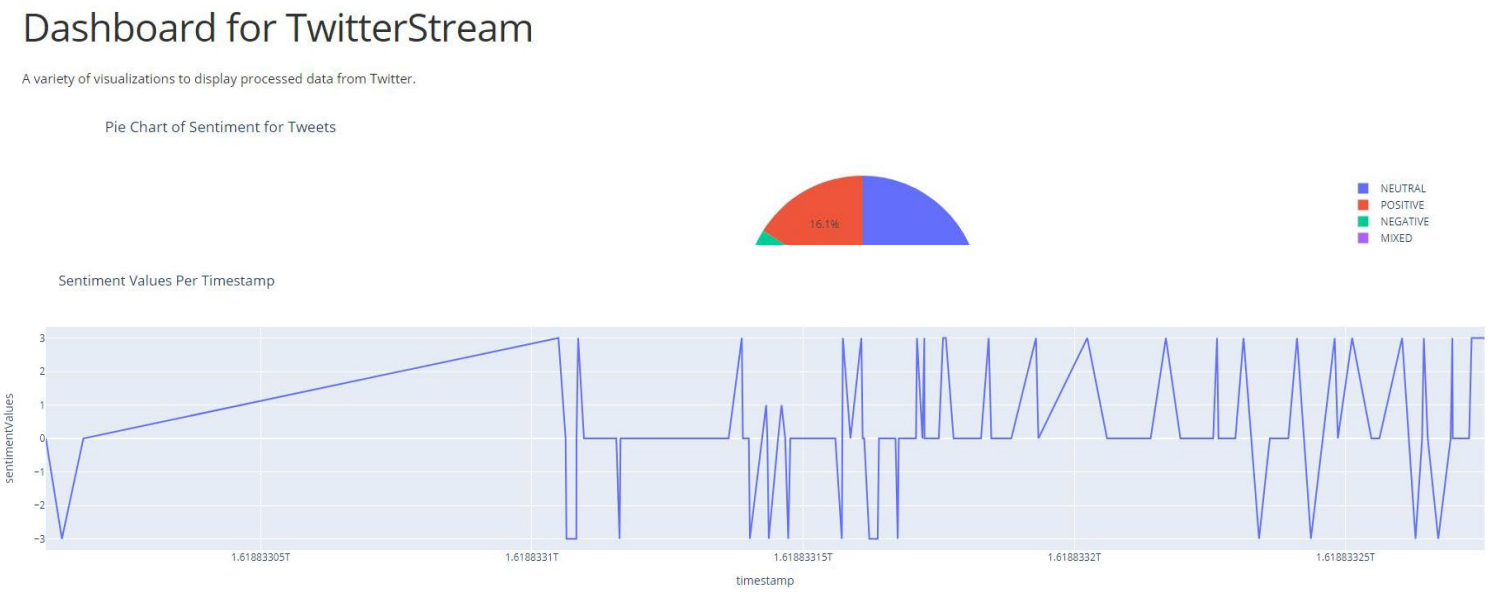
v. Total cost
    1. Instance per hour cost + EBS storage cost
    2. $25.92 + $4.05 = **$29.97**

g. Total cost for application for a month:
    i. $175.47 + $39.85 + $8.35 + $1.79 + $10.98 + $29.97 =
    ii. **$266.41**

The total cost per month for the whole application is about $268.74, making certain assumptions about the total requests/items/tweets and data size per tweets. For each service, Lambda costs $177.80 per month, DynamoDB costs $39.85 per month, EC2 costs $8.35 per month, Kinesis Data Stream costs $10.98 per month, Kinesis Firehose costs $1.79 per month, and Elasticsearch costs $29.97 per month. Overall, the Lambda service costs the most each month.

3) Implement a web application using Flask or Django to display processed data retrieved from DynamoDB and develop your own visualizations
    a. First of all, I chose to use Dash, a Python framework for building web analytic applications, to implement my web application for my own visualizations. Dash is written on top of Flask, Plotly.js, and React.js, allowing for a similar way to implement a data visualization app using only python script.
    b. The first thing I did was install dash and pandas through the script below in order have access to the necessary libraries for dash and the visualizations I am trying to implement:
        i. **pip install dash**
        ii. **pip install pandas**
    c. I then used boto3 to access my DynamoDB table for Tweet, creating a IAM User that has the policy AmazonDynamoDBFullAccess attached. From this IAM User, I obtained a AWS Access Key and Secret Access Key that I used to gain access to my DynamoDB from the script as shown below:
        i. **dynamodb = boto3.resource('dynamodb',**
        aws_access_key_id=AWS_ACCESS_KEY,
        aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
        region_name=AWS_REGION)
    d. After gaining access to my dynamodb table, I obtained each column of values by reading each line and appending the values under the correct column array.
    e. Afterwards, I went through each visualization starting from the pie chart, line graph, map, and lastly the table.

f. For the pie chart, I had 4 different categories: POSITIVE, NEGATIVE, NEUTRAL, MIXED. I created an array that held each category and another array that held the total number of tweets that were a part of each category. So, I created a for loop that compared the category name with all sentiment values from the Tweet table and incremented the sum value for that category if it matched.



g. For the line graph, I set the timestamp as the x-axis and the sentiment values as the y-axis. As labeled from challenge 1, I had made the string value sentiments equal numerical values. POSITIVE = 3, NEGATIVE = -3, NEUTRAL = 0, MIXED = 1 were the values I set them as. Rather than using the string values, I used the numerical values for the sentiment/y-axis. I had gotten the sentiment values already from the Tweet table since it was originally in a numerical value from when I did challenge 1.

h. For the map, I used the location column from the dynamodb and split the string into the latitude and longitude values. I also created another array to hold the count for each location. Afterwards, I created the map with the longitude and latitude values and made

the markers for each location change sizes depending on the count or how many tweets came from that location.

i. For the table, I created a Data Table using values directly from the Tweet table. I used the timestamp, sentiment, tweet_text, and tweet_user_id columns for the data table since these values best represent a tweet

timestamp	sentiment	tweet_text	tweet_user_id
1618833120617	NEUTRAL	MI vs DC Preview, Indian Premier League 2021: Battle Of Best As Delhi Capitals Face Defending Champions Mumbai Indi… https://t.co/AmLdW1no7N	LatestNdtv
1618833115261	NEUTRAL	@RevintoPvt Shahi Rajasthan - Rajasthan Royals #REVINTOPREMIERLEAGUE #RPL #BigWin #Win #Contest #IPL #Cricket… https://t.co/JFWPcnNStU	Chhbi777
1618833104907	POSITIVE	RT @mir_sana05: It is an absolute pleasure and honour to be included in @ICC's pilot female future leaders program with the greats like @be…	AhmedRa69736257
1618833169932	NEUTRAL	RT @TinusvS4: "This has left the Minister with no further option but to exercise his rights in terms of section 13(5) of the Sports Act.…	logicaldebate
1618833160742	POSITIVE	RT @farghie: Y a des mecs autour de nous qui sont excités par le projet SuperLeague quand meme mdrrr, faut rappeler que le projet est const…	MisterBenko
1618833148137	NEUTRAL	RT @LetsWatcho: Apna game khud banaiye. Watcho Premiere League Quiz Contest par hain dheron sawal, aur unpe dheron inaam. #cricket #crick…	MeenaMi58737458
1618833262899	NEGATIVE	RT @drlangtry_girl: Oh dear GOD, are you all still on about the football. It's not like it's cricket.	TheFactCompiler
1618833134878	NEUTRAL	Cricket here we come	Kijana_Wa_Chuka
1618833133219	NEUTRAL	RT @Itxjunu18: Type of Accounts on Cricket Twitter- A thread https://t.co/z7Rwu2lx2N	SaiKrishna_45
1618833188388	NEUTRAL	RT @ZeeNewsSports: जानिए Head To Head में कौन हैं आगे? #CSKvsRR #IPL2021 https://t.co/rw0cwQRATB	ZeeNews
1618833129835	NEUTRAL	@cricketpun_duh Cricket is already closed though..	deepeshsalhotra
1618833184695	NEUTRAL	RT @KKRiders: 158* (73) 😎😎😎 T20 Cricket was never the same again 💜 @Bazmccullum #OnThisDay #KKRHaiTaiyaar #IPL2021 https://t.co/hvV1yjhmKe	i_am_srk_jawwad
1618833184215	POSITIVE	@SDhawan25 का #IPL2021 में कमाल, #OrangeCap पर किया कब्जा https://t.co/KhUYZ9PCeC via @fitsportsindia… https://t.co/YWZS0Hc9gW	fitsportsindia
1618833147267	NEGATIVE	@AllenFCB12 I blame modi, also cricket, because finished sport	DG_Migueroz
1618833109609	NEUTRAL	RT @Amit_kr_Roy01: Cricket is not just a game is an emotion actually ☝️💪 #IPL2021 #SundayThoughts #StaySafeStayHealthy https://t.co/1tZu81…	jeetu_pbtech
1618833151140	NEUTRAL	RT @exharrie: so u agree? that the rapid progress in men's cricket is fueled by street cricket? and that development in women's cricket in…	terhideewani
1618833238996	NEUTRAL	RT @UshaPlay: 📢 Paltan, Pick the Wrong'un ❌ Follow @mipaltan and stand a chance to win @UshaIntl Mixer Grinder Follow at @UshaPlay for w…	Chandra56039776
1618833255996	NEUTRAL	@KP24 New cricket helmets looking a bit guff	jamesradhi
1618833157143	NEGATIVE	RT @ESPNcricinfo: Dilhara Lokuhettige, the former Sri Lanka allrounder, has been banned from all cricket for eight years by the ICC Anti-Co…	Murubele
1618833206028	NEUTRAL	@Paytm I'm gifting you Yorker Card for Paytm Cricket League! Get started & Win up to ₹10,000! #PaytmCricketLeague https://t.co/Bi2SzRMGrG	MindFreak0813
1618833118181	NEUTRAL	RT @RevintoPvt: #REVINTOPREMIERLEAGUE Unishade SHARE & PREDICT THE WINNER OF THE MATCH. Winner wins Rs 500 voucher(NO PURCHASE LIMIT) Ret…	Chhbi777
1618833275603	POSITIVE	RT @BarrieEditor1: It's the cricket season! Here's Billy Dane having a smashing time on one of our Tiger covers. Gran not amused! https:/…	chiddickstree
1618833273240	POSITIVE	RT @myohmizmai: Yuzu's eye smiles! So glad there is ONE cricket bros photo. :) https://t.co/aZ2dSLvmQ8	sakura2016_aa
1618833226312	POSITIVE	The show must go on! Bangladesh Cricket Team is all ready to start its Test tour of Sri Lanka this week despite co… https://t.co/brVRRDzOSs	TenPakistan

j. Lastly, I ran this app by running the line:
   i. **python app.py**

k. If successful, the command prompt should say:
   i. **Dash is running on http://127.0.0.1:8050/**
   *** Serving Flask app "app" (lazy loading)**
   *** Environment: production**
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
   *** Debug mode: on**

l. In order to view the web app, open **http://127.0.0.1:8050/** on your browser.

4) Create CloudFormation stack for the architecture implemented

a. When filling out the missing pieces needed in the template to create the CloudFormation stack for the minimal architecture, I first used the Lab 5 instruction set to understand what the section missing on the template is trying to implement. The CloudFormation template went along step by step with the instruction set for Lab 5. From there, there was sections from different AWS services that were not already in the template. So, I went to the AWS CloudFormation online User Guide to understand how to implement and adjust the configurations for the other services. I used this website to find the documentation for each service and what I needed to fill in to match with the architecture we have built:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html
I was able to completely fill out the CloudFormation stack template using the website above and the code already in the template. In order to check that my CloudFormation stack implements the architecture correctly, I used AWS CloudFormation and created a stack by uploading the template and including stack details such as keys and names. I checked to see if everything was implemented just like the original architecture given to us and the AWS comprehend update as well.