Lab 9 Notes

Student ID: 22994257
Name: Gaoyuan Zhang

[Step 1] Detecting Languages from text

In [19]:
```
language_name={
    'en': 'English',
    'es': 'Spanish',
    'fr': 'French',
    'it': 'Italian'
}
text1 = "The French Revolution was a period of social and political upl
text2 = "El Quijote es la obra más conocida de Miguel de Cervantes Saa
text3 = "Moi je n'étais rien Et voilà qu'aujourd'hui Je suis le gardie
text4 = "L'amor che move il sole e l'altre stelle."
```

In [22]:
```
def detect_language(text):
    response = client.detect_dominant_language(Text=text)
    name = language_name[response['Languages'][0]['LanguageCode']]
    confidence = math.floor(response['Languages'][0]['Score']*100)
    print(name,'detected with', confidence, '% confidence')
```

In [23]:
```
detect_language(text1)
```

```
English detected with 99 % confidence
```

In [24]:
```
detect_language(text2)
```

```
Spanish detected with 99 % confidence
```

In [25]:
```
detect_language(text3)
```

```
French detected with 99 % confidence
```

In [26]:
```
detect_language(text4)
```

```
Italian detected with 99 % confidence
```

[Step 2] Sentiment Analysis

```
In [36]: def sentiment_analysis(text, code):
             response = client.batch_detect_sentiment(TextList = [text], LanguageCode = code)
             print(response['ResultList'][0]['SentimentScore'])

In [37]: sentiment_analysis(text2, 'es')

{'Positive': 0.1598433405160904, 'Negative': 0.0008151692454703152, 'Neutral': 0.8380755186080933, 'Mixed': 0.00126604095567017
8}

In [38]: sentiment_analysis(text3, 'fr')

{'Positive': 0.3935803174972534, 'Negative': 0.5921157598495483, 'Neutral': 0.013823595829308033, 'Mixed': 0.000480222224723547
7}

In [39]: sentiment_analysis(text4, 'it')

{'Positive': 0.985332190990448, 'Negative': 0.004254049155861139, 'Neutral': 0.009053979068994522, 'Mixed': 0.00135976471938192
84}
```

[Step 3] Repeat steps from [Step 2] for detecting entities.

```
In [40]: def detect_entities(text, code):
             response = client.batch_detect_entities(TextList = [text], LanguageCode = code)
             print(response['ResultList'][0]['Entities'])

In [41]: detect_entities(text2, 'es')

[{'Score': 0.9886308908462524, 'Type': 'TITLE', 'Text': 'El Quijote', 'BeginOffset': 0, 'EndOffset': 10}, {'Score': 0.999227941
0362244, 'Type': 'PERSON', 'Text': 'Miguel de Cervantes Saavedra', 'BeginOffset': 38, 'EndOffset': 66}, {'Score': 0.90985006093
97888, 'Type': 'QUANTITY', 'Text': 'primera parte', 'BeginOffset': 81, 'EndOffset': 94}, {'Score': 0.9835476279258728, 'Type':
'TITLE', 'Text': 'El ingenioso hidalgo don Quijote de la Mancha', 'BeginOffset': 112, 'EndOffset': 157}, {'Score': 0.8697961568
832397, 'Type': 'DATE', 'Text': '1605', 'BeginOffset': 173, 'EndOffset': 177}, {'Score': 0.6641538143157959, 'Type': 'QUANTIT
Y', 'Text': 'una', 'BeginOffset': 182, 'EndOffset': 185}, {'Score': 0.9596462845802307, 'Type': 'OTHER', 'Text': 'española', 'B
eginOffset': 231, 'EndOffset': 239}, {'Score': 0.719598114490509, 'Type': 'QUANTITY', 'Text': 'una de las más traducidas', 'Beg
inOffset': 269, 'EndOffset': 294}, {'Score': 0.9852679967880249, 'Type': 'DATE', 'Text': '1615', 'BeginOffset': 299, 'EndOffse
t': 303}, {'Score': 0.94227135181427, 'Type': 'QUANTITY', 'Text': 'segunda parte', 'BeginOffset': 318, 'EndOffset': 331}, {'Sco
re': 0.9497652649879456, 'Type': 'TITLE', 'Text': 'Quijote de Cervantes', 'BeginOffset': 336, 'EndOffset': 356}, {'Score': 0.98
99224638938904, 'Type': 'TITLE', 'Text': 'El ingenioso caballero don Quijote de la Mancha', 'BeginOffset': 374, 'EndOffset': 42
1}]

In [43]: detect_entities(text3, 'fr')

[{'Score': 0.9872375726699829, 'Type': 'DATE', 'Text': "aujourd'hui", 'BeginOffset': 32, 'EndOffset': 43}, {'Score': 0.69591051
34010315, 'Type': 'QUANTITY', 'Text': "Tout ce qu'il", 'BeginOffset': 127, 'EndOffset': 140}, {'Score': 0.6048811078071594, 'Ty
pe': 'QUANTITY', 'Text': 'tout', 'BeginOffset': 200, 'EndOffset': 204}, {'Score': 0.5311335325241089, 'Type': 'QUANTITY', 'Tex
t': 'tout', 'BeginOffset': 223, 'EndOffset': 227}]

In [45]: detect_entities(text4, 'it')

[]
```

Answer: In my words, entities are the real-world objects in a text, such as a person, location, organization, date and so on, which can be denoted with a proper name. The English word like ‘a’ ‘the’ can’t be an entity.

[Step 4] Repeat steps from [Step 2] for detecting key phrases.

```
[46]: def detect_key_phrases(text, code):
          response = client.batch_detect_key_phrases(TextList = [text], LanguageCode = code)
          print(response['ResultList'][0]['KeyPhrases'])
```

```
[47]: detect_key_phrases(text2, 'es')
```

```
[{'Score': 0.9994539022445679, 'Text': 'El Quijote', 'BeginOffset': 0, 'EndOffset': 10}, {'Score': 0.9999617338180542, 'Text':
'la obra', 'BeginOffset': 14, 'EndOffset': 21}, {'Score': 0.9988439679145813, 'Text': 'más conocida', 'BeginOffset': 22, 'EndOf
fset': 34}, {'Score': 0.9999772906303406, 'Text': 'Miguel de Cervantes Saavedra', 'BeginOffset': 38, 'EndOffset': 66}, {'Scor
e': 0.9999061226844788, 'Text': 'su primera parte', 'BeginOffset': 78, 'EndOffset': 94}, {'Score': 0.9999792575836182, 'Text':
'el título', 'BeginOffset': 99, 'EndOffset': 108}, {'Score': 0.9560838341712952, 'Text': 'El ingenioso hidalgo don Quijote de l
a Mancha', 'BeginOffset': 112, 'EndOffset': 157}, {'Score': 0.9998094439506531, 'Text': 'comienzos', 'BeginOffset': 160, 'EndOf
fset': 169}, {'Score': 0.9962199330329895, 'Text': '1605', 'BeginOffset': 173, 'EndOffset': 177}, {'Score': 0.9999752044677734,
'Text': 'las obras', 'BeginOffset': 189, 'EndOffset': 198}, {'Score': 0.9998710751533508, 'Text': 'más destacadas', 'BeginOffse
t': 199, 'EndOffset': 213}, {'Score': 0.9999703168869019, 'Text': 'la literatura española', 'BeginOffset': 217, 'EndOffset': 23
9}, {'Score': 0.9999440312385559, 'Text': 'la literatura universal', 'BeginOffset': 242, 'EndOffset': 265}, {'Score': 0.9995015
263557434, 'Text': 'las más traducidas', 'BeginOffset': 276, 'EndOffset': 294}, {'Score': 0.9999786019325256, 'Text': 'la segun
da parte', 'BeginOffset': 315, 'EndOffset': 331}, {'Score': 0.9999293088912964, 'Text': 'Quijote de Cervantes', 'BeginOffset':
336, 'EndOffset': 356}, {'Score': 0.9999479055404663, 'Text': 'el título', 'BeginOffset': 361, 'EndOffset': 370}, {'Score': 0.9
364849328994751, 'Text': 'ingenioso caballero don Quijote de la Mancha', 'BeginOffset': 377, 'EndOffset': 421}]
```

```
[48]: detect_key_phrases(text3, 'fr')
```

```
[{'Score': 0.9999090433120728, 'Text': 'Moi', 'BeginOffset': 0, 'EndOffset': 3}, {'Score': 0.9550336599349976, 'Text': 'je', 'B
eginOffset': 4, 'EndOffset': 6}, {'Score': 0.9563182592391968, 'Text': "n'étais rien", 'BeginOffset': 7, 'EndOffset': 19}, {'Sc
ore': 0.9490335583686829, 'Text': "aujourd'hui", 'BeginOffset': 32, 'EndOffset': 43}, {'Score': 0.941249430179596, 'Text': 'Je
suis le gardien Du sommeil de ses nuits', 'BeginOffset': 44, 'EndOffset': 86}, {'Score': 0.9999150037765503, 'Text': 'Je', 'Beg
inOffset': 87, 'EndOffset': 89}, {'Score': 0.9998370409011841, 'Text': "l'", 'BeginOffset': 90, 'EndOffset': 92}, {'Score': 0.9
990161657333374, 'Text': 'Vous', 'BeginOffset': 106, 'EndOffset': 110}, {'Score': 0.982813835144043, 'Text': 'Tout ce', 'BeginO
ffset': 127, 'EndOffset': 134}, {'Score': 0.9646840691566467, 'Text': "qu'", 'BeginOffset': 135, 'EndOffset': 138}, {'Score':
0.9985837936401367, 'Text': 'il', 'BeginOffset': 138, 'EndOffset': 140}, {'Score': 0.9997034668922424, 'Text': 'vous', 'BeginOf
fset': 141, 'EndOffset': 145}, {'Score': 0.9997631907463074, 'Text': 'Elle', 'BeginOffset': 153, 'EndOffset': 157}, {'Score':
0.9946832656860352, 'Text': "L'espace de ses bras", 'BeginOffset': 174, 'EndOffset': 194}, {'Score': 0.8932915329933167, 'Tex
t': 'tout', 'BeginOffset': 200, 'EndOffset': 204}, {'Score': 0.9624954462051392, 'Text': 'tout', 'BeginOffset': 223, 'EndOffse
t': 227}, {'Score': 0.9998551607131958, 'Text': 'Je', 'BeginOffset': 241, 'EndOffset': 243}, {'Score': 0.9999161958694458, 'Tex
t': "l'", 'BeginOffset': 244, 'EndOffset': 246}]
```

```
[49]: detect_key_phrases(text4, 'it')
```

```
[{'Score': 0.9999063611030579, 'Text': "L'amor", 'BeginOffset': 0, 'EndOffset': 6}, {'Score': 0.9997649788856506, 'Text': 'ch
e', 'BeginOffset': 7, 'EndOffset': 10}, {'Score': 0.999977171421051, 'Text': 'il sole', 'BeginOffset': 16, 'EndOffset': 23},
{'Score': 0.9999290108680725, 'Text': "l'altre stelle", 'BeginOffset': 26, 'EndOffset': 40}]
```

Answer: In my words, key phrases are some key noun phrases in one sentence, such as ‘The French Revolution’, ‘a period’ ,‘social and political upheaval’ and 'France'.

[Step 5] Repeat steps from [Step 2] for detecting syntax.

```
def detect_Syntax(text, code):
    response = client.detect_syntax(Text = text, LanguageCode = code)
    print(response['SyntaxTokens'])
```

```
detect_Syntax(text1, 'en')
```

```
[{'TokenId': 1, 'Text': 'The', 'BeginOffset': 0, 'EndOffset': 3, 'PartOfSpeech': {'Tag': 'DET', 'Score': 0
{'TokenId': 2, 'Text': 'French', 'BeginOffset': 4, 'EndOffset': 10, 'PartOfSpeech': {'Tag': 'PROPN', 'Scor
7}}, {'TokenId': 3, 'Text': 'Revolution', 'BeginOffset': 11, 'EndOffset': 21, 'PartOfSpeech': {'Tag': 'PRO
15483093262}}, {'TokenId': 4, 'Text': 'was', 'BeginOffset': 22, 'EndOffset': 25, 'PartOfSpeech': {'Tag': '
6681213378906}}, {'TokenId': 5, 'Text': 'a', 'BeginOffset': 26, 'EndOffset': 27, 'PartOfSpeech': {'Tag': '
966025352478}}, {'TokenId': 6, 'Text': 'period', 'BeginOffset': 28, 'EndOffset': 34, 'PartOfSpeech': {'Tag
0.9999529719352722}}, {'TokenId': 7, 'Text': 'of', 'BeginOffset': 35, 'EndOffset': 37, 'PartOfSpeech': {'T
0.9987958073616028}}, {'TokenId': 8, 'Text': 'social', 'BeginOffset': 38, 'EndOffset': 44, 'PartOfSpeech':
e': 0.9973918199539185}}, {'TokenId': 9, 'Text': 'and', 'BeginOffset': 45, 'EndOffset': 48, 'PartOfSpeech'
ore': 0.9999731183052063}}, {'TokenId': 10, 'Text': 'political', 'BeginOffset': 49, 'EndOffset': 58, 'Part
DJ', 'Score': 0.9997422695159912}}, {'TokenId': 11, 'Text': 'upheaval', 'BeginOffset': 59, 'EndOffset': 67
ag': 'NOUN', 'Score': 0.9997760653495789}}, {'TokenId': 12, 'Text': 'in', 'BeginOffset': 68, 'EndOffset':
{'Tag': 'ADP', 'Score': 0.999699592590332}}, {'TokenId': 13, 'Text': 'France', 'BeginOffset': 71, 'EndOffs
h': {'Tag': 'PROPN', 'Score': 0.9975756406784058}}, {'TokenId': 14, 'Text': 'and', 'BeginOffset': 78, 'End
peech': {'Tag': 'CONJ', 'Score': 0.9999986886978149}}, {'TokenId': 15, 'Text': 'its', 'BeginOffset': 82, '
```

```
detect_Syntax(text2, 'es')
```

```
[{'TokenId': 1, 'Text': 'El', 'BeginOffset': 0, 'EndOffset': 2, 'PartOfSpeech': {'Tag': 'DET', 'Score': 0.9999085664749146}},
{'TokenId': 2, 'Text': 'Quijote', 'BeginOffset': 3, 'EndOffset': 10, 'PartOfSpeech': {'Tag': 'PROPN', 'Score': 0.52565830945968
63}}, {'TokenId': 3, 'Text': 'es', 'BeginOffset': 11, 'EndOffset': 13, 'PartOfSpeech': {'Tag': 'VERB', 'Score': 0.9999171495437
622}}, {'TokenId': 4, 'Text': 'la', 'BeginOffset': 14, 'EndOffset': 16, 'PartOfSpeech': {'Tag': 'DET', 'Score': 0.9999368190765
381}}, {'TokenId': 5, 'Text': 'obra', 'BeginOffset': 17, 'EndOffset': 21, 'PartOfSpeech': {'Tag': 'NOUN', 'Score': 0.9993823766
708374}}, {'TokenId': 6, 'Text': 'más', 'BeginOffset': 22, 'EndOffset': 25, 'PartOfSpeech': {'Tag': 'ADV', 'Score': 0.999944686
8896484}}, {'TokenId': 7, 'Text': 'conocida', 'BeginOffset': 26, 'EndOffset': 34, 'PartOfSpeech': {'Tag': 'ADJ', 'Score': 0.826
4637589454651}}, {'TokenId': 8, 'Text': 'de', 'BeginOffset': 35, 'EndOffset': 37, 'PartOfSpeech': {'Tag': 'ADP', 'Score': 0.999
9833106994629}}, {'TokenId': 9, 'Text': 'Miguel', 'BeginOffset': 38, 'EndOffset': 44, 'PartOfSpeech': {'Tag': 'PROPN', 'Score':
0.9809792637825012}}, {'TokenId': 10, 'Text': 'de', 'BeginOffset': 45, 'EndOffset': 47, 'PartOfSpeech': {'Tag': 'ADP', 'Score':
```

```
detect_Syntax(text3, 'fr')
```

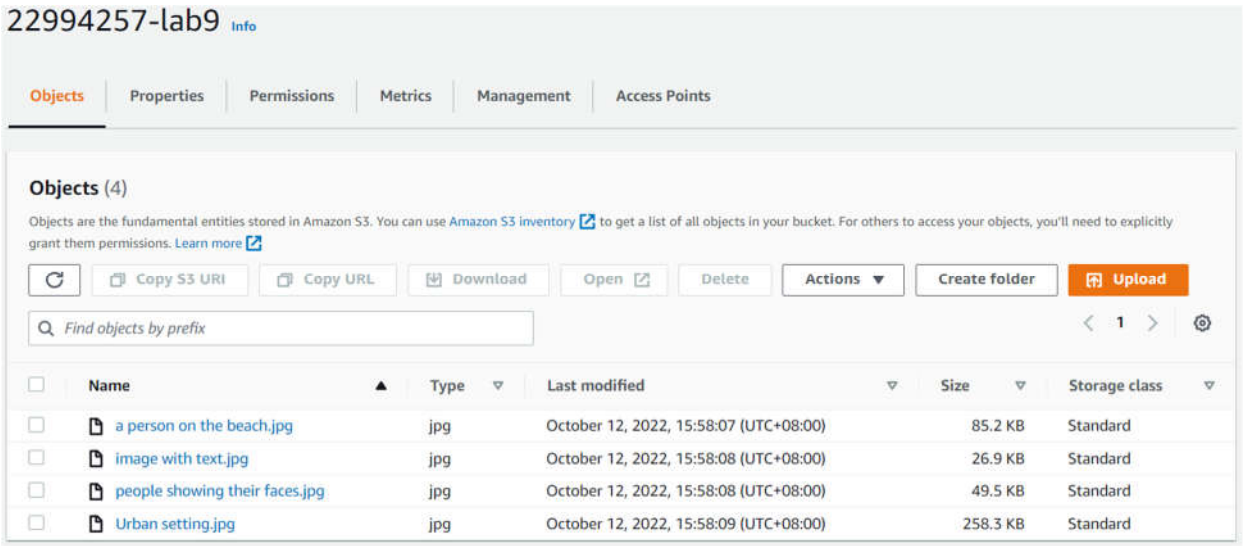
```
[{'TokenId': 1, 'Text': 'Moi', 'BeginOffset': 0, 'EndOffset': 3, 'PartOfSpeech': {'Tag': 'PRON', 'Score': 0.989230908052063}},
{'TokenId': 2, 'Text': 'je', 'BeginOffset': 4, 'EndOffset': 6, 'PartOfSpeech': {'Tag': 'PRON', 'Score': 0.9999996423721313}},
{'TokenId': 3, 'Text': "n'", 'BeginOffset': 7, 'EndOffset': 9, 'PartOfSpeech': {'Tag': 'ADV', 'Score': 0.9420575499534607}},
{'TokenId': 4, 'Text': 'étais', 'BeginOffset': 9, 'EndOffset': 14, 'PartOfSpeech': {'Tag': 'AUX', 'Score': 0.953959465026855
5}}, {'TokenId': 5, 'Text': 'rien', 'BeginOffset': 15, 'EndOffset': 19, 'PartOfSpeech': {'Tag': 'PRON', 'Score': 0.952463924884
7961}}, {'TokenId': 6, 'Text': 'Et', 'BeginOffset': 20, 'EndOffset': 22, 'PartOfSpeech': {'Tag': 'CCONJ', 'Score': 0.9998279809
951782}}, {'TokenId': 7, 'Text': 'voilà', 'BeginOffset': 23, 'EndOffset': 28, 'PartOfSpeech': {'Tag': 'VERB', 'Score': 0.999816
5965080261}}, {'TokenId': 8, 'Text': "qu'", 'BeginOffset': 29, 'EndOffset': 32, 'PartOfSpeech': {'Tag': 'SCONJ', 'Score': 0.640
893280506134}}, {'TokenId': 9, 'Text': "aujourd'hui", 'BeginOffset': 32, 'EndOffset': 43, 'PartOfSpeech': {'Tag': 'ADV', 'Scor
e': 0.9999371767044067}}, {'TokenId': 10, 'Text': 'Je', 'BeginOffset': 44, 'EndOffset': 46, 'PartOfSpeech': {'Tag': 'PRON', 'Sc
ore': 0.9998699426651001}}, {'TokenId': 11, 'Text': 'suis', 'BeginOffset': 47, 'EndOffset': 51, 'PartOfSpeech': {'Tag': 'AUX',
```

```
detect_Syntax(text4, 'it')
```

```
[{'TokenId': 1, 'Text': "L'", 'BeginOffset': 0, 'EndOffset': 2, 'PartOfSpeech': {'Tag': 'DET', 'Score': 0.9990321397781372}},
{'TokenId': 2, 'Text': 'amor', 'BeginOffset': 2, 'EndOffset': 6, 'PartOfSpeech': {'Tag': 'NOUN', 'Score': 0.9907339811325073}},
{'TokenId': 3, 'Text': 'che', 'BeginOffset': 7, 'EndOffset': 10, 'PartOfSpeech': {'Tag': 'PRON', 'Score': 0.9950056672096252}},
{'TokenId': 4, 'Text': 'move', 'BeginOffset': 11, 'EndOffset': 15, 'PartOfSpeech': {'Tag': 'VERB', 'Score': 0.997265696525573
7}}, {'TokenId': 5, 'Text': 'il', 'BeginOffset': 16, 'EndOffset': 18, 'PartOfSpeech': {'Tag': 'DET', 'Score': 0.999986290931701
7}}, {'TokenId': 6, 'Text': 'sole', 'BeginOffset': 19, 'EndOffset': 23, 'PartOfSpeech': {'Tag': 'NOUN', 'Score': 0.996540427207
9468}}, {'TokenId': 7, 'Text': 'e', 'BeginOffset': 24, 'EndOffset': 25, 'PartOfSpeech': {'Tag': 'CCONJ', 'Score': 0.99983000755
31006}}, {'TokenId': 8, 'Text': "l'", 'BeginOffset': 26, 'EndOffset': 28, 'PartOfSpeech': {'Tag': 'DET', 'Score': 0.99996674060
82153}}, {'TokenId': 9, 'Text': 'altre', 'BeginOffset': 28, 'EndOffset': 33, 'PartOfSpeech': {'Tag': 'ADJ', 'Score': 0.68281966
4478302}}, {'TokenId': 10, 'Text': 'stelle', 'BeginOffset': 34, 'EndOffset': 40, 'PartOfSpeech': {'Tag': 'NOUN', 'Score': 0.994
3856000900269}}, {'TokenId': 11, 'Text': '.', 'BeginOffset': 40, 'EndOffset': 41, 'PartOfSpeech': {'Tag': 'PUNCT', 'Score': 0.9
999809265136719}}]
```

Answer: In my words, the syntax tokens are all the tokens in the sentences and it will be classified with a tag, such as ‘ADJ’, ’NOUN’, ’ADV’, which means ‘adjective’ ‘noun’ and ‘adverb’.

[Step 6] AWS Rekognition



1. Label Recognition

```
def detect_labels(photo, bucket):
    client=boto3.client('rekognition')
    response = client.detect_labels(Image={'S3Object':{'Bucket':bucket,'Name':photo}}, MaxLabels=10)
    print('Detected', len(response['Labels']), 'labels for ',photo)
    print()
    for label in response['Labels']:
        print ("Label: " + label['Name'])
        print ("Confidence: " + str(label['Confidence']))
        print ("----------")
```

```
detect_labels('Urban setting.jpg','22994257-lab9')
```

```
Detected 10 labels for  Urban setting.jpg

Label: Grass
Confidence: 99.95657348632812
----------
Label: Plant
Confidence: 99.95657348632812
----------
Label: City
Confidence: 98.78321838378906
----------
Label: Urban
Confidence: 98.78321838378906
----------
Label: Building
Confidence: 98.78321838378906
----------
Label: High Rise
Confidence: 97.90977478027344
----------
Label: Lawn
Confidence: 94.08061218261719
----------
Label: Downtown
Confidence: 93.8046875
----------
Label: Park
Confidence: 88.2553482055664
----------
Label: Metropolis
Confidence: 81.7234115600586
----------
```

2. Image Moderation

```
def moderate_image(photo, bucket):
    client=boto3.client('rekognition')
    response = client.detect_moderation_labels(Image={'S3Object':{'Bucket':bucket,'Name':photo}})
    for label in response['ModerationLabels']:
        print (label['Name'] + ' : ' + str(label['Confidence']))
        print (label['ParentName'])
```

```
moderate_image('a person on the beach.jpg','22994257-lab9')
```

```
Suggestive : 86.2708969116211

Barechested Male : 86.2708969116211
Suggestive
```

3. Facial Analysis

```
def detect_faces(photo, bucket):
    client=boto3.client('rekognition')
    response = client.detect_faces(Image={'S3Object':{'Bucket':bucket,'Name':photo}},Attributes=['ALL']
    for faceDetail in response['FaceDetails']:
        print('The detected face is between ' + str(faceDetail['AgeRange']['Low'])
              + ' and ' + str(faceDetail['AgeRange']['High']) + ' years old')
        print('Here are the other attributes:')
        # print(json.dumps(faceDetail, indent=4, sort_keys=True))
        # Access predictions for individual face details and print them
        print("Gender: " + str(faceDetail['Gender']))
        print("Smile: " + str(faceDetail['Smile']))
        print("Eyeglasses: " + str(faceDetail['Eyeglasses']))
        print("Emotions: " + str(faceDetail['Emotions'][0]))
        print ("----------")
```

```
detect_faces('people showing their faces.jpg','22994257-lab9')
```

```
The detected face is between 36 and 44 years old
Here are the other attributes:
Gender: {'Value': 'Male', 'Confidence': 98.40275573730469}
Smile: {'Value': True, 'Confidence': 95.01976013183594}
Eyeglasses: {'Value': False, 'Confidence': 95.87667846679688}
Emotions: {'Type': 'HAPPY', 'Confidence': 96.12042236328125}
----------
The detected face is between 25 and 35 years old
Here are the other attributes:
Gender: {'Value': 'Female', 'Confidence': 99.99845123291016}
Smile: {'Value': True, 'Confidence': 96.15525817871094}
Eyeglasses: {'Value': False, 'Confidence': 97.4869613647461}
Emotions: {'Type': 'HAPPY', 'Confidence': 98.3642349243164}
----------
```

4. Detect Text from an image

```
def detect_text(photo, bucket):
    client=boto3.client('rekognition')
    response=client.detect_text(Image={'S3Object':{'Bucket':bucket,'Name':photo}})
    textDetections=response['TextDetections']
    print ('Detected text\n----------')
    for text in textDetections:
        print ('Detected text:' + text['DetectedText'])
        print ('Confidence: ' + "{:.2f}".format(text['Confidence']) + "%")
        print ('Id: {}'.format(text['Id']))
        if 'ParentId' in text:
            print ('Parent Id: {}'.format(text['ParentId']))
        print ('Type:' + text['Type'])
        print ("----------")
```

```
detect_text('image with text.jpg', '22994257-lab9')
```

```
Detected text
----------
Detected text:COME SAIL AWAY
Confidence: 99.83%
Id: 0
Type:LINE
----------
Detected text:COME
Confidence: 99.78%
Id: 1
Parent Id: 0
Type:WORD
----------
Detected text:SAIL
Confidence: 100.00%
Id: 2
Parent Id: 0
Type:WORD
----------
Detected text:AWAY
Confidence: 99.69%
Id: 3
Parent Id: 0
Type:WORD
----------
```