Question A. FindBusinessBasedOnCity Function: this function uses the parameters (cityToSearch, saveLocation1, collection) to find all the business from collection which are present in 'cityToSearch' and writes the resulting list to the file specified by 'saveLocation1'

1. Reflection

```
import math
# Graded Cell, PartID: o1flK
def FindBusinessBasedOnCity(cityToSearch,saveLocation1,collection):
    d = collection.filter(lambda obj: obj['city'] == cityToSearch)
    f = open(saveLocation1, 'w')
    for bus in d:
        txt = bus['name']+ "$" + bus['full_address']+"$" + bus['city'] + "$" + bus ['state']+ "\n"
        f.write(txt)
    f.close()
```

NOTE: The 'import math' line is not relevant to this function but keep it in mind for the next function as it frequently makes use of the math library.

After doing some troubleshooting on the collection object to familiarize myself with it, I identified the columns of the dataset and the rest of the program developed from there. The clear route forward was to use a lambda function to filter which objects in the dataset were in the city. With that achieved, 'd' contained a list of the relevant businesses and the only thing left to do was parse and write the relevant data into the file in the format shown by the 'true_results' list.

```
true_results = ["VinciTorio's Restaurant$1835 E Elliot Rd, Ste C109, Tempe, AZ 85284$Tempe$AZ", "P.croissants$7520 S Rural Rd, Te
```

2. Output

Using the 'true_results' list as a guide I formatted the output to use '$' symbols as delimiters and placed each query result on a new line so that they would get read properly by the 'readlines()' function call within the test case.

My output wrote the results to the 'output_city.txt' file in the following format.

```
1 VinciTorio's Restaurant$1835 E Elliot Rd, Ste C109, Tempe, AZ 85284$Tempe$AZ
2 Salt Creek Home$1725 W Ruby Dr, Tempe, AZ 85284$Tempe$AZ
3 P.croissants$7520 S Rural Rd, Tempe, AZ 85283$Tempe$AZ
4
```

3. Result

```
Correct! You FindBusinessByCity function passes these test cases. This does not cover all possible test edge cases, however, so
make sure that your function covers them before submitting!
```

The function passed the given test cases but I did some testing of my own to make sure that it was correctly writing to the file. It worked given a variety of different city queries so I left it as it was.

4. Lessons Learned

For this function I one of the most important things I learned was how to use a collection object to parse data entries. I have used other object types before like Dataframes but collection objects are fairly new to me. I am also not the best just yet with lambda functions but obviously they are hugely powerful and relevant for this kind of parsing.

Question B. FindBusinessBasedOnLocation Function: This function uses the parameters (categoriesToSearch, myLocation, maxDistance,saveLocation2, collection) to find businesses in the 'collection' that are within the 'maxDistance' of 'myLocation'. The function filters by 'categoriesToSearch' then checks whether matching businesses are close enough to the user to display, the output is then written into the file specified by 'saveLocation2'.

1. Reflection

This function took a little more effort to program correctly than the previous one… Turns out calculating distance given a pair of latitude/longitude coordinates is somewhat harder than calculating distance in your typical 2D reference. Reviewing the project Overview document was HUGELY helpful here as it provided a detailed description of what the distanceFunction should look like. Here is the distance function I made using the description(Note the need for the math library):

```
def dist(lat2, lon2, lat1, lon1):
    R = 3959
    o1 = math.radians(lat1)
    o2 = math.radians(lat2)
    do = math.radians(lat2 - lat1)
    dl = math.radians(lon2 - lon1)
    h = math.sin(do/2)*math.sin(do/2) + math.cos(o1) * math.cos(o2) * math.sin(dl/2) * math.sin(dl/2)
    c = 2* math.atan2(math.sqrt(h), math.sqrt(1-h))
    distance = R * c
    return distance
```

And here is the actual function which utilizes it.

```
def FindBusinessBasedOnLocation(categoriesToSearch, myLocation, maxDistance, saveLocation2, collection):
    f = open(saveLocation2, 'w')
    for destination in collection:
        if(dist(myLocation[0], myLocation[1], destination['latitude'],destination['longitude']) < maxDistance):
            l = []
            for de in destination['categories']:
                l.append(de)
            if categoriesToSearch[0] in l:
                result = destination['name']
                f.write(result)
    f.close()
```

I began by writing the general outline for what the function should do. I knew any result needed to be within the maxDistance of myLocation so I iterated over collections and grabbed any destinations within that radius. I then used a list to store the categories of each destination to see whether any of them matched the categoriesToSearch parameter. If they did I wrote the name of the destination to the file.

2. Output

```
1 VinciTorio's Restaurant
```

In the test case there was only a single business in the 'Buffets' category that was within 10 miles of [33.3482589, -111.9088346]. The function therefore wrote 'VinciTorio's Restaurant' into the 'output_loc.txt' file. Other searches for different categories or coordinates also yielded different results.

3. Result

```
Correct! Your FindBusinessBasedOnLocation function passes these test cases. This does not cover all possible edge cases, so mak
e sure your function does before submitting.
```

The function passed the test cases provided. As mentioned before I did try some other coordinates to ensure proper functioning of the distance and category filters. At first I tested New York [40.7123, -74.26], but quickly realized the db was fairly small and mostly contained businesses in the tempe and phoenix area… moving the latitude and longitude back in that direction allowed me to successfully query for other businesses.

4. Lessons Learned

With this program I learned how to find distance given latitudinal and longitudinal coordinates. This involved calculating the angular positions of the two locations, the angular distance between them and using the sin and cos functions to estimate the circumference of slice given the radius of the earth (3959 miles). I also learned to write query results to disk for long term storage and match my query formats to the desired test case results.