For each of the following use the SWAPI docs, go figure out the complete URL(s) (including params or queries) that you need to go to in order to reach the following data:

1. the height of Darth Vader

   https://swapi.dev/api/people/4/

   https://swapi.dev/api/people/?search=darth vader

   202 cm

2. the population of the planet Alderaan

   https://swapi.dev/api/planets?search=alderaan

   https://swapi.dev/api/planets/2/

   2000000000

3. the name of the manufacturer of the Millennium Falcon

   https://swapi.dev/api/starships?search=falcon

   https://swapi.dev/api/starships/10/

   Corellian Engineering Corporation

4. the name of the species that C-3PO belongs to (multiple URLs)

   https://swapi.dev/api/people?search=c-3po

   https://swapi.dev/api/people/2/

   https://swapi.dev/api/species/2/

   Droid

5. the title of each film that Obi-Wan Kenobi is in (multiple URLs)

   https://swapi.dev/api/people/?search=obi-wan

   https://swapi.dev/api/films/1/

   A New Hope

   https://swapi.dev/api/films/2/

   The Empire Strikes Back

   https://swapi.dev/api/films/3/

   Return of the Jedi

   https://swapi.dev/api/films/4/

   The Phantom Menace

   https://swapi.dev/api/films/5/

   Attack of the Clones

https://swapi.dev/api/films/6/

Revenge of the Sith

6. use the search query (the how to on the search query is at the bottom of the Getting Started section of the documentation) to get the information about the Millennium Falcon, it's a starship

   https://swapi.dev/api/starships?search=millenium falcon

In this section, you'll be looking through the documentation for the Social Mountain API and answering questions. You'll also be making requests and recording the URLs and some information about the responses. Run the requests in Postman. **Note: this API is live and viewable by your classmates and staff. Keep things appropriate for class.**

1. Check if the POST request accepts params, queries, and/or a body. Which one(s) and what information is it expecting to be sent?
   a. The POST request's body is an object, and that object contains a single property ("text"), which is a string.

2. What data type does the GET request return?
   a. It returns an array of all the posts

3. What would the URL look like for deleting the post with the id 555? (This post does not exist anymore, but the syntax is the same for existing posts)
   a. https://practiceapi.devmountain.com/api/posts?id=555

4. List the possible response codes from the GET request at '/posts/filter'
   a. 200 & 409

5. Create a post whose text is your name, record the URL and body here:
   a. https://practiceapi.devmountain.com/api/posts
   b. { "id": 6845, "text": "clint", "date": "18 Aug 2021" }
   c. { "id": 6850, "text": "Carston Work", "date": "18 Aug 2021" }

6. What would the URL and body object be to update the post you just made to contain your favorite color instead of your name?
   a. { "id": 6850, "text": "Royal Blue", "date": "18 Aug 2021" }
      https://practiceapi.devmountain.com/api/posts?id=6850
   b. { "id": 6845, "text": "Blue", "date": "18 Aug 2021" }
      https://practiceapi.devmountain.com/api/posts?id=6845

7. What is the URL to get posts that contain the text “blue”?
    a. https://practiceapi.devmountain.com/api/posts/filter?text=blue

8. Make a request to GET all the posts. What are the content type and charset of the response? (Hint: look on the Headers)
    a. application/json; charset=utf-8

9. What would cause a PUT request to return a 409 status?
    a. Request was missing req.query.id or req.body.text

10. What happens if you try to send a query in the GET request URL? Why do you get that response?
    a. We get back all of the posts. This happens because the /posts endpoint is not set up to receive queries. It is programmed to simply send back all of the posts.