The folder contains three files:

- script.py: code
- Headers.json: header json
- graphqlQuery.json: graphqlQuery

The scripts queries graphql API endpoint to acquire the product price and other metadata information. To by-pass the CORS mechanism, I injected the http header from chrome browser to the python requests. I also copied the graphql query value from the browser's request payload and removed redundant fields and the size of the response data.

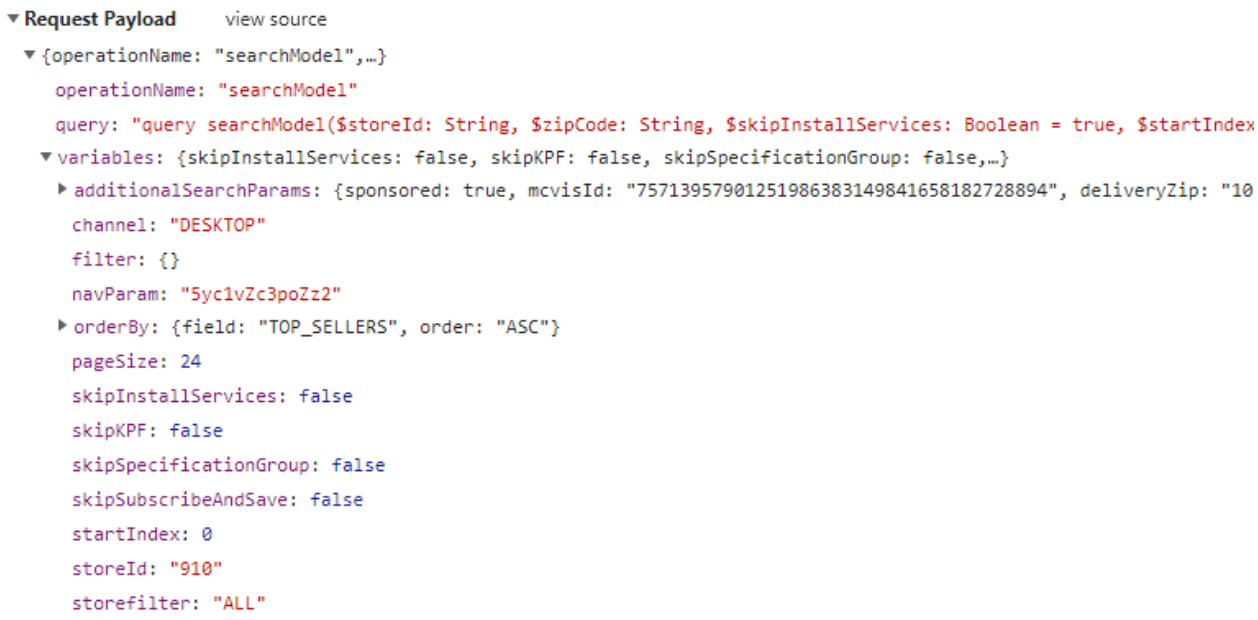


Fig. graphql request payload(query and variables)

The list of departments and subdepartments are acquired from this endpoint which returns a nested json object for each department and subdepartment.

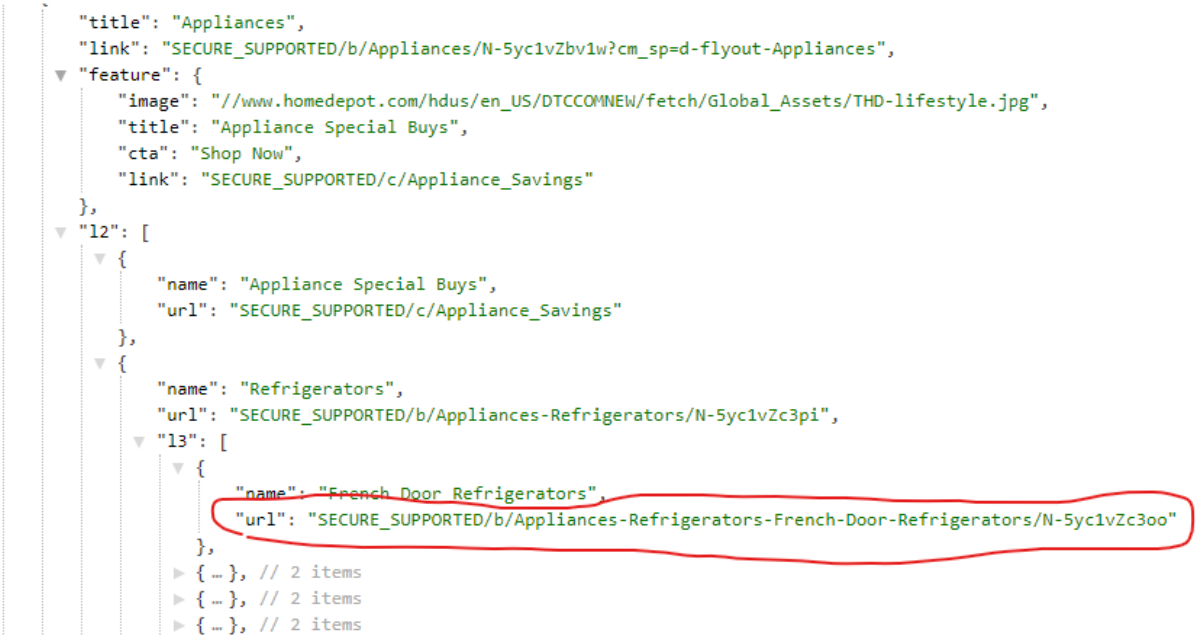


Fig. nested json object that contains department and subdepartment information

After searching for the required department and subdepartments in the nested json object, I took the sub department url to navigate to the sub department products page and filter by brands.

Currently, the script selects multiple brands by subsequently navigating to the link(href) for each brand starting from the department url.

```
def getBrandURL(departmentURL, brand):
    headers = getJson(HEADER)
    html = requests.get(departmentURL, headers=headers)
    print(departmentURL)
    soup = BeautifulSoup(html.text, 'html.parser')
    url = soup.find_all(text=brand)[0].parent.parent.get('href')
    return HOMEDEPOT_URL + url
```

Fig. function for getting brand url starting from department url

This implementation can be improved by including brand name and their id value(each brand has a data-id value) inside the department URL path instead of making multiple requests.

Store filtering is done by updating the storeId field in the graphql query json. The navParam field in graphql query specifies the navigation parameter associated with the sub/department and list of brands.

```
"variables":{
  "storefilter": "ALL",
  "channel": "DESKTOP",
  "additionalSearchParams": {
    "sponsored": True,
    "mcvisId": "75713957901251986383149841658182728894",
    "deliveryZip": "10010"
  },
  "filter": {},
  "navParam": "N-5yc1vZc3piZ",
  "orderBy": {
    "field": "TOP_SELLERS",
    "order": "ASC"
  },
  "pageSize": 48,
  "startIndex": 0,
  "storeId": "6845"
}
```

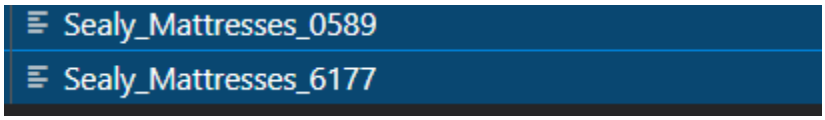
Fig. fields for the graphql variables

How to run the script for the given categories:

```
#run category 1

 python script.py -s_ids '6177' '0589' -d1 'Appliances' -d2
'Dishwashers' -b 'Samsung' 'LG Electronics' 'LG STUDIO' 'LG
SIGNATURE' -o 'LG_SamSung_Dishwasher'

# run category 2
 python script.py -s_ids '6177' '0589' -d1 'Appliances' -d2
'Refrigerators' -b 'Whirlpool' 'GE' -o
'Whirlpool_GE_Refrigerators'

# run category 3
python script.py -s_ids '6177' '0589' -d1 'Decor & Furniture'
-d2 'Bedroom Furniture' 'Mattresses' -b 'Sealy' -o
'Sealy_Mattresses'
```

Note:

- The script produces separate output files for each store.



- LG brand has three variations, LG Electronics, Studio and Signature.

- The script requires for the brand names to be passed explicitly(similar to run category 1).