BIS 634 01 / CB&B 634 01 Assignment 3 ¶

Exercise 1

Use the requests module (or urllib) to use the Entrez API (see slides 5) to identify the PubMed IDs for 1000 Alzheimers papers from 2022 and for 1000 cancer papers from 2022. (9 points)

Note: To search for a disease and a publication year, structure the term like: Alzheimers+AND+2022[pdat] (Here [pdat] indicates that this is a publication year, and the AND (has to be all caps) means both conditions should apply.)

Use the Entrez API via requests/urllib to pull the metadata for each such paper found above (both cancer and Alzheimers) (and save a JSON file storing each paper's title, abstract, and the query that found it that is of the general form: (12 points)

```
{
    "32008517": {
        "ArticleTitle": "Deep Learning for Alzheimer's Disease Classification using Texture Features",
        "AbstractText": "We propose a classification method for Alzheimer's disease (AD)...",
        "query": "Alzheimer"
    }, ...
}
```

Here 32008517 would be the PubMed ID of one of the 2000 papers, specifically one that came from searching for Alzheimer's papers (it won't be in your data set because it was published in 2019). You should include the full AbstractText; I'm abridging here for clarity.

There are of course many more papers of each category, but is there any overlap in the two sets of papers that you identified? (3 points)

Hint: To do this, you'll probably want to look at one of the XML responses with a text editor so that you understand how it is structured.

Hint: Some papers like 32008517 have multiple AbstractText fields (e.g. when the abstract is structured). Be sure to store all parts. You could do this in many ways, from using a dictionary or a list or simply concatenating with a space in between. Discuss any pros or cons of your choice in your readme (1 point).

Caution: the PubMed API allows a rate of at most one query at a time and no more than 3 per second unless you have an API key. To be safe, use

```
time.sleep(1)
```

after each query to the PubMed API.

Note: This doesn't require 2002 separate queries. You can get the metadata for many articles at a time by using a comma separated list of ids. While GET queries have a total line length limit, you could use a POST query instead and get the information for all the papers in one pass. (We can use POST instead of GET here in part because this is not a RESTful API.)

Note: BioPython provides functions for accessing the PubMed API. Do not use them; use the requests module to do an HTTP or HTTPS request directly on a URL that you specify with the parameters that you specify. Why? Because this approach is general and will work in many contexts whereas BioPython only works for PubMed and only from Python.

NOTE: sometimes papers (e.g. 31842501) have italics in their title or abstract. If so, using .text won't work well; use ET.tostring(item, method="text").decode() instead.

Response

In [84]:
```
# import modules
import requests
import xml.dom.minidom as m
import xml.etree.ElementTree as et
import json
import time
```

Creating a function to return the list of a specified diseases using the requests module

In [85]:
```
):
cer/alzheimers

papers per diseases
ml
et(f"https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=pubmed&term={disease}+AND+2022[pdat]&retmax=1000&retmode=xml")

tring(r.text)
.getElementsByTagName('Id')

(len(PubmedId)):
end(PubmedId[i].firstChild.data)
```

In [86]:
```
## Checking for the total length of the paper : as specified should be 1000

len(get_id("Alzheimers") + get_id("Cancer"))
```

Out[86]: 2000

In [95]:
```
## PubMedId for Disease - Alzheimers

get_id("Alzheimers")
```

Out[95]:
```
['36323061',
 '36322888',
 '36322800',
 '36322495',
 '36322470',
 '36321981',
 '36321927',
 '36321882',
 '36321654',
 '36321615',
 '36321363',
 '36321205',
 '36321194',
 '36320609',
 '36320346',
 '36319270',
 '36319045',
 '36318754',
 '36318594',
```

In [96]:
```
## PubMedId for Disease - Cancer

get_id("Cancer")
```

Out[96]:
```
['36323507',
 '36323504',
 '36323475',
 '36323457',
 '36323452',
 '36323443',
 '36323436',
 '36323435',
 '36323434',
 '36323433',
 '36323432',
 '36323431',
 '36323430',
 '36323420',
 '36323419',
 '36323417',
 '36323370',
 '36323360',
 '36323356',
```

Finding overlap between two sets of papers

In [91]:
```
def overlap_PubmedId(disease1,disease2):
    return set(get_id(disease1))&set(get_id (disease2))
```

In [92]:
```
overlap_PubmedId("Alzheimers", "Cancer")
```

Out[92]:
```
{'36314209'}
```

Finding the MetaData of the papers in Alzheimers and Cancer Sets

In [93]:
```
def metadata(disease):
    PubmedIdList = get_id(disease)
    disease_paper_dict = {}
    for PubmedId in PubmedIdList:
        time.sleep(1)
        r = requests.post(f"https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmed&retmode=xml&id={int(PubmedId)}")
        doc = m.parseString(r.text)

        ArticleTitle = doc.getElementsByTagName('ArticleTitle')
        Title = ""
        if len(ArticleTitle) > 0:
            for elm in ArticleTitle:
                for textmessage in elm.childNodes:
                    try:
                        Title += textmessage._get_wholeText()
                        Title = et.tostring(Title, method = "text").decode()

                    except AttributeError:
                        for subnode in textmessage.childNodes:
                            if subnode.nodeType == m.Node.TEXT_NODE:
                                Title += subnode.data

        AbstractText = doc.getElementsByTagName('AbstractText')
        Abstract = ""
        if len(AbstractText) > 0:
            for elm in AbstractText:
                for textmessage in elm.childNodes:
                    try:
                        Abstract += textmessage._get_wholeText()
                        Abstract = et.tostring(Abstract, method = "text").decode()
                    except AttributeError:
                        for subnode in textmessage.childNodes:
                            if subnode.nodeType == m.Node.TEXT_NODE:
                                Abstract += subnode.data


        MeshHeading = doc.getElementsByTagName('MeshHeading')
        ArticleMeshTerms = []
        if len(MeshHeading) > 0:
            try:
                for i in MeshHeading:
                    ArticleMeshTerms.append(i.firstChild.childNodes[0].nodeValue)
            except AttributeError: pass

        disease_paper_dict[PubmedId] = {
            'ArticleTitle': Title,
            'ArticleAbstract': Abstract,
            'Query': disease,
            'Mesh': ArticleMeshTerms
        }

    return  disease_paper_dict
```

I looped over the abstract section to get all the elements. I think it is an efficient and less time consuming way to find the and store the data. The data was then was store in a form of string and then to a common dictionary including the other data keys. But while going through the data file, the structured parameters names such as objective, results, method were not stored. It would be difficult to extract information of objective/method/results of all papers separately.

Pulling Metadata for Alzheimers and saving the JSON file

In [36]:
```
alzheimers_metadata = metadata('Alzheimers')
```

In [37]:
```
with open("Alzheimer.json", "w") as f:
    json.dump(alzheimers_metadata, f, indent=4)
```

Pulling Metadata for Cancer and saving the JSON file

In [38]:
```
cancer_metadata = metadata('Cancer')
```

In [39]:
```
with open("Cancer.json", "w") as f:
    json.dump(cancer_metadata, f, indent=4)
```

Combing the data in a single json file

In [94]:
```
combined_data = metadata('Alzheimers')
cancer_data = metadata('Cancer')
combined_data.update(cancer_data)

with open('papers.json','w') as f:
    json.dump(combined_data,f)
```

Exercise 2

Machine learning and data visualization strategies generally work best on data that is numeric, but exercise 1 gave us text data, and indeed text is common. Fortunately, modern NLP algorithms powered by machine learning trained on massive datasets exist that can take words (e.g. word2vec) or titles and abstracts (e.g. SPECTER) and return a vector of numbers in a way that similar items are given similar vectors. Since we have titles and abstracts, let's use SPECTER.

In particular, for each paper identified from exercise 1, compute the SPECTER embedding (a 768-dimensional vector). Keep track of which papers came from searching for Alzheimers, which came from searching for cancer. (5 points) If you are familiar with SPECTER and wish to do it another way, that's great, if not here's one approach based on https://github.com/allenai/specter (https://github.com/allenai/specter):

Install pytorch (a deep learning library) by following the instructions here: https://pytorch.org/get-started/locally/ (https://pytorch.org/get-started/locally/)

Install the huggingface transformers module: pip install transformers

(🤗 huggingface provides access to a number of pre-trained NLP language models.)

Have your code load the SPECTER model (the first time you do this, it will take a bit to download the model; it will be stored locally for fast reuse later):

```
from transformers import AutoTokenizer, AutoModel

# load model and tokenizer
tokenizer = AutoTokenizer.from_pretrained('allenai/specter')
model = AutoModel.from_pretrained('allenai/specter')
```

Load the papers dictionary (3 points) and then process your dictionary of papers to find the SPECTER embeddings (2 points). One (somewhat slow) way to do this is as follows:

```
import tqdm

# we can use a persistent dictionary (via shelve) so we can stop and restart if needed
# alternatively, do the same but with embeddings starting as an empty dictionary
embeddings = {}
for pmid, paper in tqdm.tqdm(papers.items()):
    data = [paper["ArticleTitle"] + tokenizer.sep_token + get_abstract(paper)]
    inputs = tokenizer(
        data, padding=True, truncation=True, return_tensors="pt", max_length=512
    )
    result = model(**inputs)
    # take the first token in the batch as the embedding
    embeddings[pmid] = result.last_hidden_state[:, 0, :].detach().numpy()[0]

# turn our dictionary into a list
embeddings = [embeddings[pmid] for pmid in papers.keys()]
```

At this point, embeddings[i] is the 768-dim vector for the ith paper.

Apply principal component analysis (PCA) to identify the first three principal components. (5 points) I suggest using the sklearn module, e.g.

```
from sklearn import decomposition
pca = decomposition.PCA(n_components=3)
embeddings_pca = pd.DataFrame(
    pca.fit_transform(embeddings),
    columns=['PC0', 'PC1', 'PC2']
)
embeddings_pca["query"] = [paper["query"] for paper in papers.values()]
```

Plot 2D scatter plots for PC0 vs PC1, PC0 vs PC2, and PC1 vs PC2; color code these by the search query used (Alzheimers vs cancer). (5 points) Comment on the separation or lack thereof, and any take-aways from that. (5 points)

Response

In [181]:
```
#download transformers using conda in terminal
!conda install -c huggingface transformers -y
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to
avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3

  added / updated specs:
    - transformers


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    huggingface_hub-0.10.1     |             py_0         160 KB  huggingface
    sacremoses-master          |             py_0         404 KB  huggingface
    tokenizers-0.13.0.dev0     |           py39_0         3.3 MB  huggingface
    transformers-4.24.0        |             py_0         2.6 MB  huggingface
    ------------------------------------------------------------
                                           Total:         6.5 MB

The following NEW packages will be INSTALLED:

  huggingface_hub    huggingface/noarch::huggingface_hub-0.10.1-py_0 None
  sacremoses         huggingface/noarch::sacremoses-master-py_0 None
  tokenizers         huggingface/osx-64::tokenizers-0.13.0.dev0-py39_0 None
  transformers       huggingface/noarch::transformers-4.24.0-py_0 None



Downloading and Extracting Packages
tokenizers-0.13.0.de | 3.3 MB    | ##################################### | 100%
transformers-4.24.0  | 2.6 MB    | ##################################### | 100%
sacremoses-master    | 404 KB    | ##################################### | 100%
huggingface_hub-0.10 | 160 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Retrieving notices: ...working... done
```

In [115]:
```
pip install transformers
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to
avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
Requirement already satisfied: transformers in /opt/anaconda3/lib/python3.9/site-packages (4.23.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.10.0 in /opt/anaconda3/lib/python3.9/site-packages (from transformers)
(0.10.1)
Requirement already satisfied: regex!=2019.12.17 in /opt/anaconda3/lib/python3.9/site-packages (from transformers) (2022.7.9)
Requirement already satisfied: requests in /opt/anaconda3/lib/python3.9/site-packages (from transformers) (2.28.1)
Requirement already satisfied: filelock in /opt/anaconda3/lib/python3.9/site-packages (from transformers) (3.6.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /opt/anaconda3/lib/python3.9/site-packages (from transforme
rs) (0.13.1)
Requirement already satisfied: pyyaml>=5.1 in /opt/anaconda3/lib/python3.9/site-packages (from transformers) (6.0)
Requirement already satisfied: numpy>=1.17 in /opt/anaconda3/lib/python3.9/site-packages (from transformers) (1.21.5)
Requirement already satisfied: tqdm>=4.27 in /opt/anaconda3/lib/python3.9/site-packages (from transformers) (4.64.1)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/python3.9/site-packages (from transformers) (21.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /opt/anaconda3/lib/python3.9/site-packages (from huggingface-hub<
1.0,>=0.10.0->transformers) (4.3.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/anaconda3/lib/python3.9/site-packages (from packaging>=20.0->tr
ansformers) (3.0.9)
Requirement already satisfied: charset-normalizer<3,>=2 in /opt/anaconda3/lib/python3.9/site-packages (from requests->transform
ers) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/lib/python3.9/site-packages (from requests->transformers) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/anaconda3/lib/python3.9/site-packages (from requests->transformer
s) (1.26.11)
Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/lib/python3.9/site-packages (from requests->transformers)
(2022.9.24)
Note: you may need to restart the kernel to use updated packages.
```

In [116]:
```
# import AutoTokenizer, AutoModel.
from transformers import AutoTokenizer, AutoModel

# load model and tokenizer
tokenizer = AutoTokenizer.from_pretrained('allenai/specter')
model = AutoModel.from_pretrained('allenai/specter')
```

In [117]:
```
# importing tools

import json
import numpy as np
import pandas as pd
from sklearn import decomposition
import matplotlib as plt
import seaborn as sns
```

In [118]:
```
# Loading the Alzheimers paper set file

with open("Alzheimer.json") as f:
    alzheimers_metadata = json.load(f)

alzheimer_df = pd.DataFrame.from_dict(alzheimers_metadata, orient = 'index')
alzheimer_df.head()
```

Out[118]:

	ArticleTitle	ArticleAbstract	Query	Mesh
36315115	Association of Stroke and Cerebrovascular Path...	Scam susceptibility is associated with adverse...	Alzheimers	[]
36314730	Cerebrospinal fluid neurofilament light and ce...	Neurodegeneration underpins the pathological p...	Alzheimers	[]
36314503	Improving community health-care systems' early...	Preliminary estimates suggest that current glo...	Alzheimers	[]
36314232	Erratum to: Predictors of Life Expectancy in A...		Alzheimers	[]
36314212	The Ethics of Using Caregivers as Cognitive Te...	O'Caoimh et al. demonstrated that caregivers o...	Alzheimers	[]

In [119]:
```
# Loading the Cancer paper set file

with open("Cancer.json") as f:
    cancer_metadata = json.load(f)

cancer_df = pd.DataFrame.from_dict(cancer_metadata, orient = 'index')
cancer_df.head()
```

Out[119]:

	ArticleTitle	ArticleAbstract	Query	Mesh
36316114	Apatinib as a Third-Line Treatment for HER2-Po...	Treatment options are limited after the failur...	Cancer	[]
36316113	Safe Discharge Criteria After Curative Gastrec...	This study aimed to investigate the relationsh...	Cancer	[]
36316112	Endoscopic Findings and Treatment of Gastric N...	Gastric neoplasia is a common manifestation of...	Cancer	[]
36316111	Tumor Location Causes Different Recurrence Pat...	Tumor recurrence is the principal cause of poo...	Cancer	[]
36316110	PLK2 Single Nucleotide Variant in Gastric Canc...	Chromosomal instability is a hallmark of gastr...	Cancer	[]

In [142]:
```
combined_papers = pd.concat([alzheimer_df,cancer_df])
```

In [143]:
```
combined_papers
```

Out[143]:

	ArticleTitle	ArticleAbstract	Query	Mesh
36315115	Association of Stroke and Cerebrovascular Path...	Scam susceptibility is associated with adverse...	Alzheimers	[]
36314730	Cerebrospinal fluid neurofilament light and ce...	Neurodegeneration underpins the pathological p...	Alzheimers	[]
36314503	Improving community health-care systems' early...	Preliminary estimates suggest that current glo...	Alzheimers	[]
36314232	Erratum to: Predictors of Life Expectancy in A...		Alzheimers	[]
36314212	The Ethics of Using Caregivers as Cognitive Te...	O'Caoimh et al. demonstrated that caregivers o...	Alzheimers	[]
...	...	...	...	...
36308179	Molecular subtypes of invasive breast carcinom...	Tumor budding (TB), poorly differentiated clus...	Cancer	[Humans, Female, Retrospective Studies, Recept...
36308083	Efficacy and safety of EUS-guided gallbladder ...	ERCP is the first line of treatment for malign...	Cancer	[]
36308082	Forkhead Box S1 inhibits the progression of lu...	Lung squamous cell carcinoma (SCC) is consider...	Cancer	[Mice, Animals, Wnt Signaling Pathway, beta Ca...
36308081	ZNF561 antisense RNA 1 contributes to angiogen...	Hepatocellular carcinoma is a common malignant...	Cancer	[Humans, Carcinoma, Hepatocellular, RNA, Antis...
36308078	Exosomes derived from bone marrow mesenchymal ...	Bone marrow mesenchymal stem cells (BM-MSCs), ...	Cancer	[Humans, Female, Exosomes, YAP-Signaling Prote...

2000 rows × 4 columns

In [154]:
```
# Loading the Combined paper set file : includes paper from Alzheimers and Cancer disease

with open('papers.json') as f:
    combined_data = json.load(f)
```

In [158]:
```
# Computing the SPECTER embedding

data = [paper["ArticleTitle"] + tokenizer.sep_token + paper["ArticleAbstract"] for paper in combined_data.values()]
inputs = tokenizer([data[0]], padding=True, truncation=True, return_tensors="pt", max_length=512)
result = model(**inputs)
embeddings_total = result.last_hidden_state[:, 0, :].detach().numpy()

for i in range(1,len(data)):
    inputs = tokenizer([data[i]], padding=True, truncation=True, return_tensors="pt", max_length=512)
    result = model(**inputs)
    embeddings = result.last_hidden_state[:, 0, :].detach().numpy()
    embeddings_total = np.concatenate((embeddings_total, embeddings),axis = 0)
```

In [159]:
```
# Applying principal component analysis (PCA) to identify the first three principal components

pca = decomposition.PCA(n_components=3)
embeddings_pca = pd.DataFrame(pca.fit_transform(embeddings_total),columns=['PC0', 'PC1', 'PC2'])
embeddings_pca["Query"] = [paper["Query"] for paper in combined_data.values()]
```

In [160]:
```
embeddings_pca
```

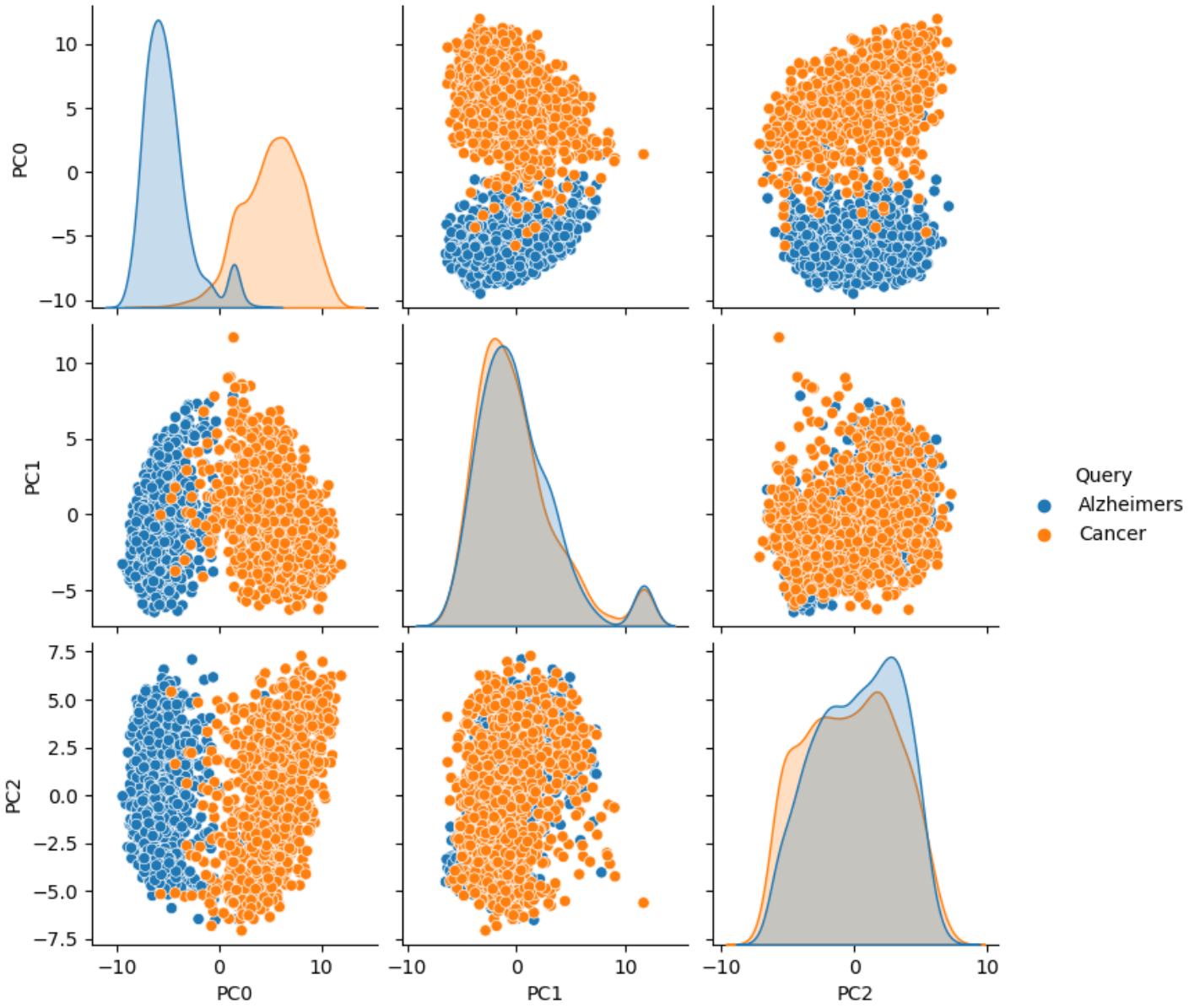
Out[160]:

	PC0	PC1	PC2	Query
0	-4.678195	3.000064	3.983280	Alzheimers
1	-5.506620	1.476830	3.631659	Alzheimers
2	-7.132979	3.150683	3.925214	Alzheimers
3	-5.904363	2.459445	2.852544	Alzheimers
4	-5.359968	-0.412834	-0.422238	Alzheimers
...	...	...	...	...
1994	8.653900	-2.024786	-1.850365	Cancer
1995	5.899750	6.154752	3.499627	Cancer
1996	3.180859	0.754004	-2.725007	Cancer
1997	2.865486	-0.261790	-6.288406	Cancer
1998	3.703853	1.045163	-2.427681	Cancer

1999 rows × 4 columns

Plotting the 2D scatter plots for PC0 vs PC1, PC0 vs PC2, and PC1 vs PC2; color code these by the search query used (Alzheimers vs cancer.

In [180]:
```
sns.pairplot(embeddings_pca, hue = 'Query')
plt.show()
```



Findings :

- Principal Component Analysis (PCA) is a linear methods used for dimensionality reduction.
- PCA seeks to build a small-dimensional coordinate system using whatever small collection of components (i.e. combinations of variables) best explains the variability of the underlying dataset, and then simply throw out the coordinates which do not have a high impact on variation.
- Scatter plot for PC0 vs PC1 using PCA method : the variation (separation) is clearly visible. The first principal component is the linear combination of x-variables that has maximum variance (among all linear combinations). It accounts for as much variation in the data as possible. When the two queries datset is projected to a two-dimensional space, can be linearly separable up to some extent.
- Scatter plot for PC0 vs PC2 using PCA method : Similarly to PC0 vs PC1, PC0 vs PC2 we can notice the separatability from the two diseases datasets.When the two queries datset is projected to a two-dimensional space, can be linearly separable up to some extent.
- Scatter plot for PC1 vs PC2 using PCA method : Datasets are not linearly separable across PC1 and PC2.
- Most of the variability in the data is captured by PC0 (Principal Component 0) and the residual variability is captured by PC1 (Principal Component 1), which is orthogonal (independent) to PC0 and PC2 (Principal Component 2), which is orthogonal (independent) to PC1. PC1 and PC2 tries to capture the abnormal variations in the dataset. PC1 and PC2 have 0 correlation.

Exercise 3

Consider the simple, non-spatial SIR modelLinks to an external site.:

In this epidemiology model, is the number of "susceptible" individuals, is the number of infected individuals, is the removed population that can no longer become sick, is the total population (hence), is a scale factor measuring how likely an infected person is to make a susceptible person sick, measures the rate at which infected individuals are removed from that state.

Write a Python function that uses the Explicit Euler method to plot given and (the last time point to compute). (N follows from the formula.) Do not use an integration library; provide your own implementation (5 points).

The New Haven population is approximately . Suppose that on day 0, there was 1 person infected with a new disease in New Haven and everyone else was susceptible (as the disease is new). Suppose further that and . Plot the time course of the number of infected individuals until that number drops below 1 (at which point, we'll assume the disease has run its course). (5 points)

For those parameter values, when does the number of infected people peak? (2 points) How many people are infected at the peak? (3 points). (Have your code identify these things; don't do it manually.)

Unfortunately, for new diseases, we may not know or with much accuracy. Vary these two variables over "nearby" values, and plot on a heat map how the time of the peak of the infection depends on these two variables. (5 points). Do the same for the number of individuals infected at peak. (5 points)

Hint: if you're using plotnine, consider using geom_raster or geom_tileLinks to an external site. to make the heat map. Other approaches are fine too.

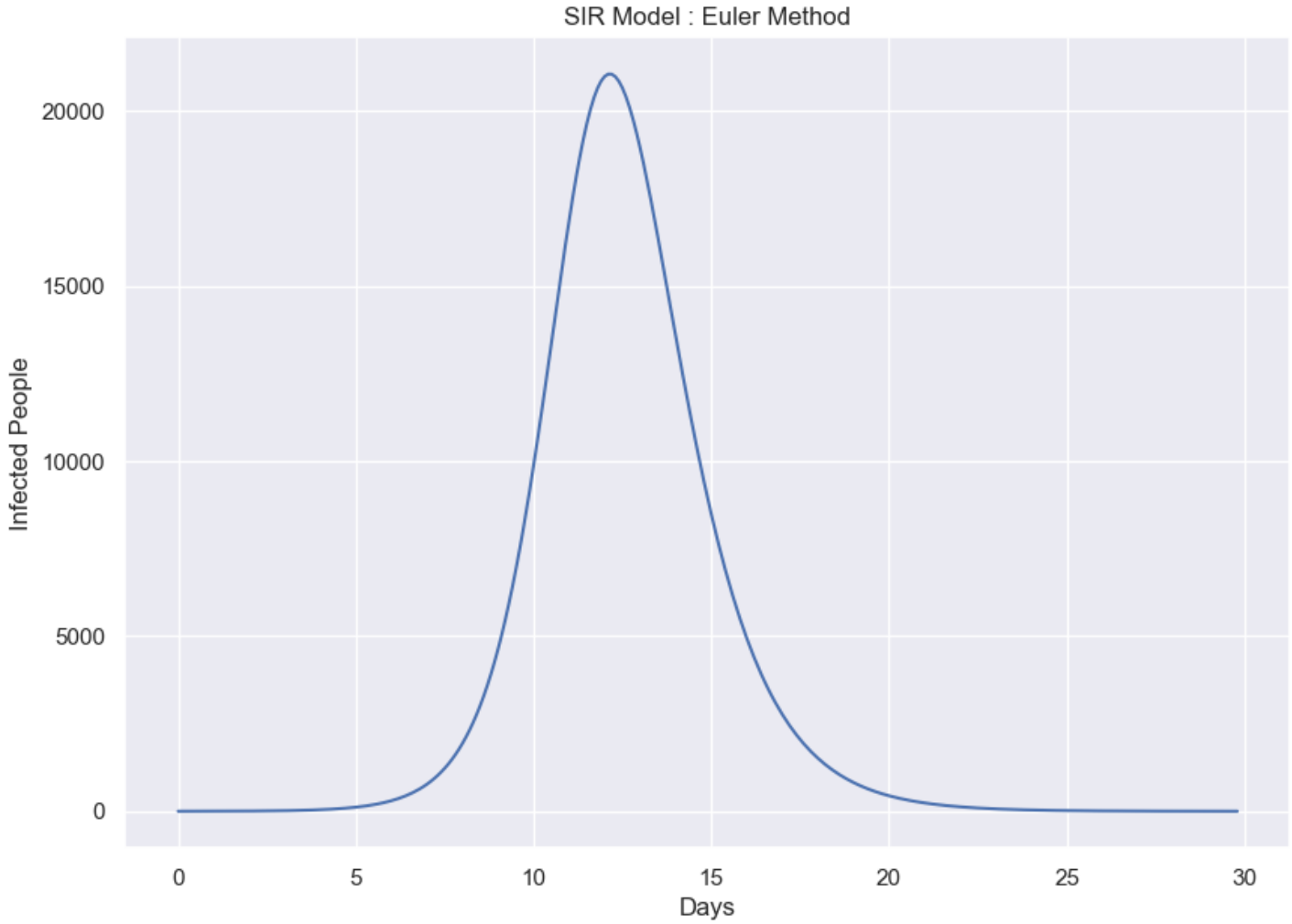
Response

In [228]:
```
# importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [229]:
```
delta_time = 0.1

def ExplicitEuler(s0, i0, r0, beta, gamma, tmax, plot):

    t = 0
    s = s0
    i = i0
    r = r0
    peak_reached = False
    peak_day= 0
    peak_infection=i0

    #x and y variables for the plot
    days = [0]
    infected = [i0]

    while t < tmax:
        susceptible_next = s + (-1*(beta/(s+i+r))*s*i)*delta_time
        infected_next = i + ((beta/(s+i+r))*s*i - (gamma*i))*delta_time
        removed_next = r + gamma*i*delta_time

        infected.append(infected_next)
        t = t + delta_time
        days.append(t)

        s, i, r = susceptible_next, infected_next, removed_next

        if not peak_reached and ((beta/(s+i+r))*s*i - (gamma*i))<0:
            peak_reached = True
            peak_day = round(t)
            peak_infection = round(i)


        if i<1:
            break
    if plot:
        plt.figure(figsize=(10,7))
        plt.plot(days, infected)
        plt.xlabel("Days")
        plt.ylabel("Infected People")
        plt.title("SIR Model : Euler Method")

    return peak_day, peak_infection
```

In [230]:
```
peak_day, peak_infection = ExplicitEuler((134000), 1,0, 2, 1, 1000, True)
print(f'The number of infected people peaked on {peak_day}th day.')
print(f'The number of people infected at the peak were {peak_infection}.')
```

```
The number of infected people peaked on 12th day.
The number of people infected at the peak were 21051.
```



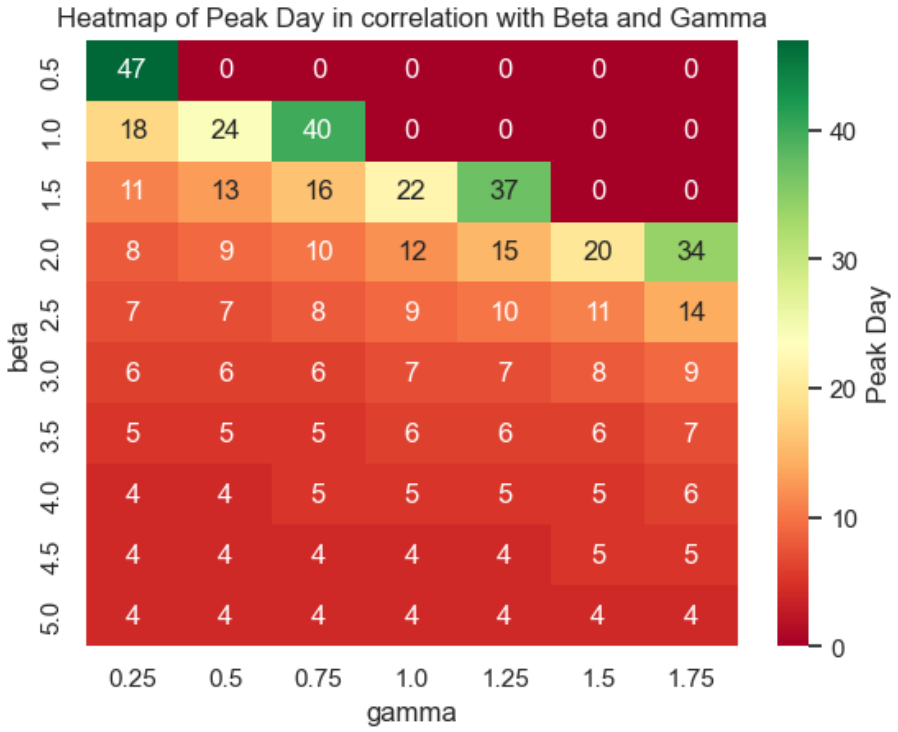
Creating a list to Vary beta and gamma over "nearby" values.

In [231]:
```
Gamma = []
Beta = []
peak_days = []
peakinfection = []
for gamma in np.arange(0.25, 2.0, 0.25):
    for beta in np.arange(0.5, 5.5, 0.5):
        peak_day, peak_infection = ExplicitEuler((134000), 1, 0, beta, gamma, 100, False)
        Beta.append(beta)
        Gamma.append(gamma)
        peak_days.append(peak_day)
        peakinfection.append(peak_infection)
```

In [232]:
```
# Creating a dataframe of the above parameters

df = pd.DataFrame({'beta': Beta,'gamma' :Gamma, 'peak_day': peak_days,'peak_infections': peakinfection})
```

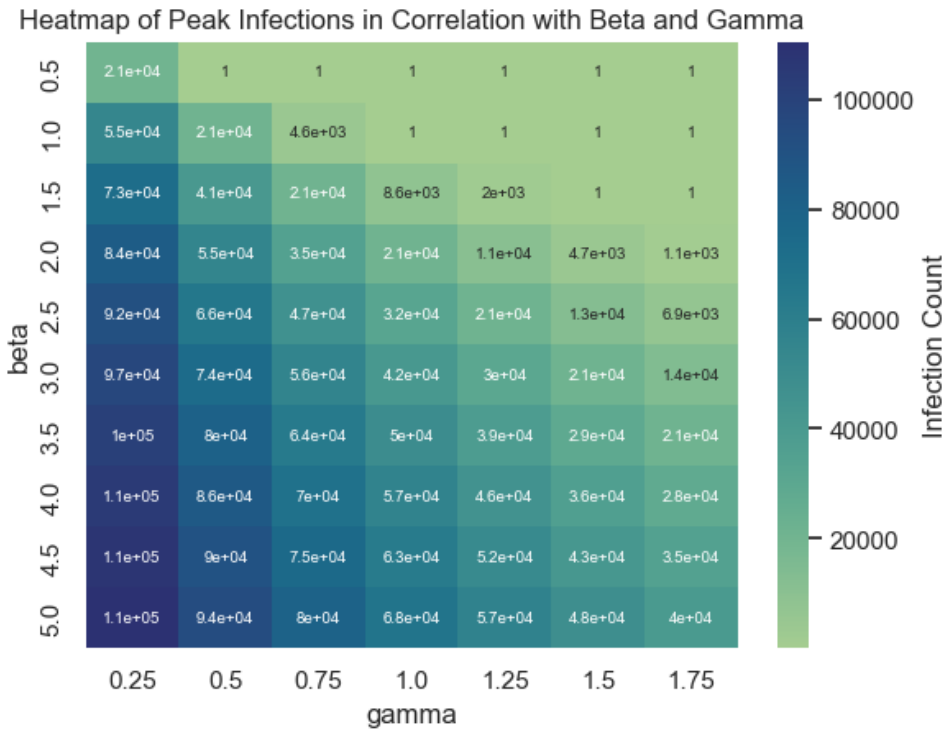
Plotting HeatMap to show how the time of the peak of the infection depends on variables beta and gamma.

In [233]:
```
Parameters = df.pivot("beta", "gamma", "peak_day")
ax = plt.axes()
sns.heatmap(Parameters,fmt="",cmap='RdYlGn', annot=True, cbar_kws={'label': 'Peak Day'})
ax.set_title("Heatmap of Peak Day in correlation with Beta and Gamma")
plt.show()
```



Plotting HeatMap to show how the number of individuals infected at peak. depends on variables beta and gamma

In [234]:
```
Parameters = df.pivot("beta", "gamma", "peak_infections")
ax = plt.axes()
sns.heatmap(Parameters, cmap="crest", annot=True, annot_kws={"size": 7}, cbar_kws={'label': 'Infection Count'})
ax.set_title("Heatmap of Peak Infections in Correlation with Beta and Gamma")
plt.show()
```



Exercise 4

Identify a data set online (5 points) that you find interesting that could potentially be used for the final project; the main requirements is that there needs to be many (hundreds or more) data items with several identifiable variables, at least one of which could be viewed as an output variable that you could predict from the others.

Describe the dataset (5 points) Your answer should address (but not be limited to): how many variables? Are the key variables explicitly specified or are they things you would have to derive (e.g. by inferring from text)? Are any of the variables exactly derivable from other variables? (i.e. are any of them redundant?) Are there any variables that could in principle be statistically predicted from other variables? How many rows/data points are there? Is the data in a standard format? If not, how could you convert it to a standard format?

Describe the terms of use and identify any key restrictions (e.g. do you have to officially apply to get access to the data? Are there certain types of analyses you can't do?) (5 points)

Remember: if you can't find explicit permission to use a given dataset, assume that you cannot do so.

Do data exploration on the dataset, and present a representative set of figures that gives insight into the data. Comment on the insights gained. (5 points)

Identify any data cleaning needs (this includes checking for missing data) and write code to perform them. If the data does not need to be cleaned, explain how you reached this conclusion. (5 points)

Response

Dataset Identified

The Health Information National Trends Survey (HINTS). I chose the HINTS 5, Cycle 3 (2019) dataset, updated April 2021 version. It has Total respondents: 5,438, Complete responses: 5,247, Partial responses: 191. The survey collects nationally representative data routinely about the American public's use of cancer-related information. It is a large dataset with several identifible variables. As per the research question, several variables can be viewed as an output variable that could be predicted from other variable. Dataset Link : https://hints.cancer.gov/data/download-data.aspx (https://hints.cancer.gov/data/download-data.aspx) (can't be uploaded to GitHub due to large volume of data).

Dataset Description

In [140]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from plotnine import *
import warnings
warnings.filterwarnings("ignore")
```

In [141]:
```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', -1)
```

In [142]:
```
pip install pyreadstat
```

```
Requirement already satisfied: pyreadstat in /opt/anaconda3/lib/python3.9/site-packages (1.2.0)
Requirement already satisfied: pandas>=1.2.0 in /opt/anaconda3/lib/python3.9/site-packages (from pyreadstat) (1.4.4)
Requirement already satisfied: python-dateutil>=2.8.1 in /opt/anaconda3/lib/python3.9/site-packages (from pandas>=1.2.0->pyread
stat) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/anaconda3/lib/python3.9/site-packages (from pandas>=1.2.0->pyreadstat) (202
2.1)
Requirement already satisfied: numpy>=1.18.5 in /opt/anaconda3/lib/python3.9/site-packages (from pandas>=1.2.0->pyreadstat) (1.
21.5)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.9/site-packages (from python-dateutil>=2.8.1->pandas>=1.
2.0->pyreadstat) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [143]:
```
HINTS_data = pd.read_spss("/Users/mahimakaur/Desktop/hints5_cycle3_public.sav")
```

In [144]:
```
HINTS_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5438 entries, 0 to 5437
Columns: 730 entries, HHID to IncomeRanges_IMP
dtypes: category(368), float64(360), object(2)
memory usage: 17.1+ MB
```

In [145]:
```
HINTS_data.head()
```

Out[145]:

	HHID	PersonID	Stratum	APP_REGION	HIGHSPANLI	HISPSURNAME	HISP_HH	RUC2003	RUC2013	PR_RUCA_2010	SEC_RUCA_2010	DRA	Treatment_H5(
0	91000002.0	91000002-02	High Minority Areas		No	No	No	Nonmetro county with urban population of 2,500-19,999, adjac	Nonmetro - Urban population of 2,500 to 19,999, adjacent to	Small town core: primary flow within an Urban Cluster of 2,5	Small town: prim flow w/in Urban Cluster 2,500-9,999(small U	In the Mississippi Delta region	Mail or
1	91000006.0	91000006-02	High Minority Areas		No	No	No	Nonmetro county with urban population of 2,500-19,999, adjac	Nonmetro - Urban population of 2,500 to 19,999, adjacent to	Metropolitan area high commuting: primary flow 30% or more t	Metro area high commuting: primary flow 30%+ to a UA, No add	In the Mississippi Delta region	Mail or
2	91000007.0	91000007-03	High Minority Areas		No	No	No	Nonmetro county with urban population of 2,500-19,999, adjac	Nonmetro - Urban population of 2,500 to 19,999, adjacent to	Small town high commuting: primary flow 30% or more to a sma	Small town/high commuting: prim flow 30%+ to small UC, Secon	In the Mississippi Delta region	Mail or
3	91000008.0	91000008-01	High Minority Areas	Southern Appalachia	No	No	No	County in metro area of fewer than 250,000 population	Metro - Counties in metro areas of fewer than 250,000 popula	Metropolitan area core: primary flow within an urbanized are	Metro area core: primary flow w/in urbanized area (UA), No a	Not in the Mississippi Delta region	Mail or
4	91000012.0	91000012-01	High Minority Areas	Southern Appalachia	No	No	No	Nonmetro county with urban population of 2,500-19,999, adjac	Nonmetro - Urban population of 2,500 to 19,999, adjacent to	Micropolitan area core: primary flow within an Urban Cluster	Micropol area: prim flow w/in Urban Cluster of 10,000-49,999	Not in the Mississippi Delta region	Mail or

In [146]:
```
HINTS_data.shape
```

Out[146]:
```
(5438, 730)
```

In [147]:
```
HINTS_data.dtypes
```

Out[147]:
```
HHID                    float64
PersonID                object
Stratum                 category
APP_REGION              category
HIGHSPANLI              category
HISPSURNAME             category
HISP_HH                 category
RUC2003                 category
RUC2013                 category
PR_RUCA_2010            category
SEC_RUCA_2010           category
DRA                     category
Treatment_H5C3          category
NCHSURCODE2013          category
CENSDIV                 category
CENSREG                 category
VAR_STRATUM             category
VAR_CLUSTER             float64
FormType                category
```

In [148]:
```
HINTS_data.count()
```

Out[148]:
```
HHID                    5438
PersonID                5438
Stratum                 5438
APP_REGION              5438
HIGHSPANLI              5438
HISPSURNAME             5438
HISP_HH                 5438
RUC2003                 5438
RUC2013                 5438
PR_RUCA_2010            5438
SEC_RUCA_2010           5438
DRA                     5438
Treatment_H5C3          5438
NCHSURCODE2013          5438
CENSDIV                 5438
CENSREG                 5438
VAR_STRATUM             5438
VAR_CLUSTER             5438
FormType                5438
```

In [149]:
```
HINTS_data.describe(include = 'all')
```

Out[149]:

	HHID	PersonID	Stratum	APP_REGION	HIGHSPANLI	HISPSURNAME	HISP_HH	RUC2003	RUC2013	PR_RUCA_2010	SEC_RUCA_2010	DRA	Treatm
count	5.438000e+03	5438	5438	5438	5438	5438	5438	5438	5438	5438	5438	5438	
unique	NaN	5438	2	4	2	2	2	9	8	11	16	2	
top	NaN	91000002-02	High Minority Areas		No	No	No	County in metro area with 1 million population or more	Metro - Counties in metro areas of 1 million population or m	Metropolitan area core: primary flow within an urbanized are	Metro area core: primary flow w/in urbanized area (UA), No a	Not in the Mississippi Delta region	
freq	NaN	1	3360	5116	4943	4844	4512	3195	3254	4364	3802	5234	
mean	9.159088e+07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	8.063104e+05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	9.100000e+07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	9.100681e+07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	9.101265e+07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	9.201817e+07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	9.302343e+07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Converting the dataset into a standard format (SPSS to CSV)

In [150]:
```
import pyreadstat
```

In [151]:
```
df, metadata = pyreadstat.read_sav("/Users/mahimakaur/Desktop/hints5_cycle3_public.sav")
csv_file = df.to_csv("/Users/mahimakaur/Desktop/hints5_cycle3_public.csv")
```

Dataset Description

- HINTS measures how people access and use health information; how people use information technology to manage health and health information; and the degree to which people are engaged in healthy behaviors. Finally, several items in HINTS have a specific focus on cancer prevention and control.
- The dataset has 5438 rows and 730 columns(variables).
- The data types include : category(368), float64(360), object(2).
- The key variables are key variables explicitly specified. I did not find any varibale can be eactly derivable from other varibale. Each question in the survey interview is a unique question.
- There are no redundant data variables. Nationally representative HINTS data are free to download and analyze.
- The dataset contains variables that could in principle be statistically predicted from other variables.It depends on the research question. For example : a) multivariable logistic regression models can be fitted to estimate predictors of achieving health goals with the help of smartphone or tablet and sending or receiving an SMS text

message to or from a health care provider. b) ordinal logistic regression models can be used identify predictors of high trust in doctors and the internet separately for cancer-seeking information. c) Additionally, subgroup-specific analyses of HINTS data can be used to inform tailored cancer control planning efforts (e.g., by gender, race/ethnicity, socioeconomic status, cancer history).The dataset can be used to determine which individual health behaviors (e.g., fruit consumption, physical activity, and sleep) are associated with psychological health among cancer survivors and 2) determine if the number of health behaviors engaged in is associated with better psychological health among cancer survivors.

- HINTS data are available in SAS, SPSS, and STATA formats. It is not available in standard formats like csv, json, xml. I converted the spss file to csv using the above mentioned code.

Terms of Use and Identification of Key Restrictions

The dataset is freely available to use. If one requires access to detailed geographic information or variables that have been suppressed in the public use datasets, then one needs to visit the Request Restricted HINTS Datasets page to request restricted-use HINTS data. The individual can either fill out and submit the interactive form below or fill out the Restricted Data Request Form and email it to NCIHINTS@mail.nih.gov (mailto:NCIHINTS@mail.nih.gov) for review. The survey has comprehensive documentation(survey instrument,data files, codebooks, design documents, methodology, and more) for public use.In order to help cultivate a community of HINTS users, the organization has asked to let them know about published HINTS-related articles so that these can be posted on the HINTS Web site.

In my opinion geospatial analyses cannot be done on publically available dataset.Because HINTS is a cross-sectional survey, it is not possible to infer causal relationships between constructs or items in the survey. Additionally, while researchers can examine trends over time at the national level for outcomes included in multiple iterations of the survey, one cannot assess change over time at the individual level.

Since, we can perform logistic and linear regressions, k-means, and examine trends over time it can be a potentiate dataset for the final project. Additionally, Because of common sampling procedures used across the repeated, cross-sectional HINTS surveys, each HINTS iteration can be considered as an independent sample of the non-institutionalized US adult population 18 and older. Thus, iterations can be merged together to look at patterns over time and yield larger sample size.

Exploratory Data Analysis

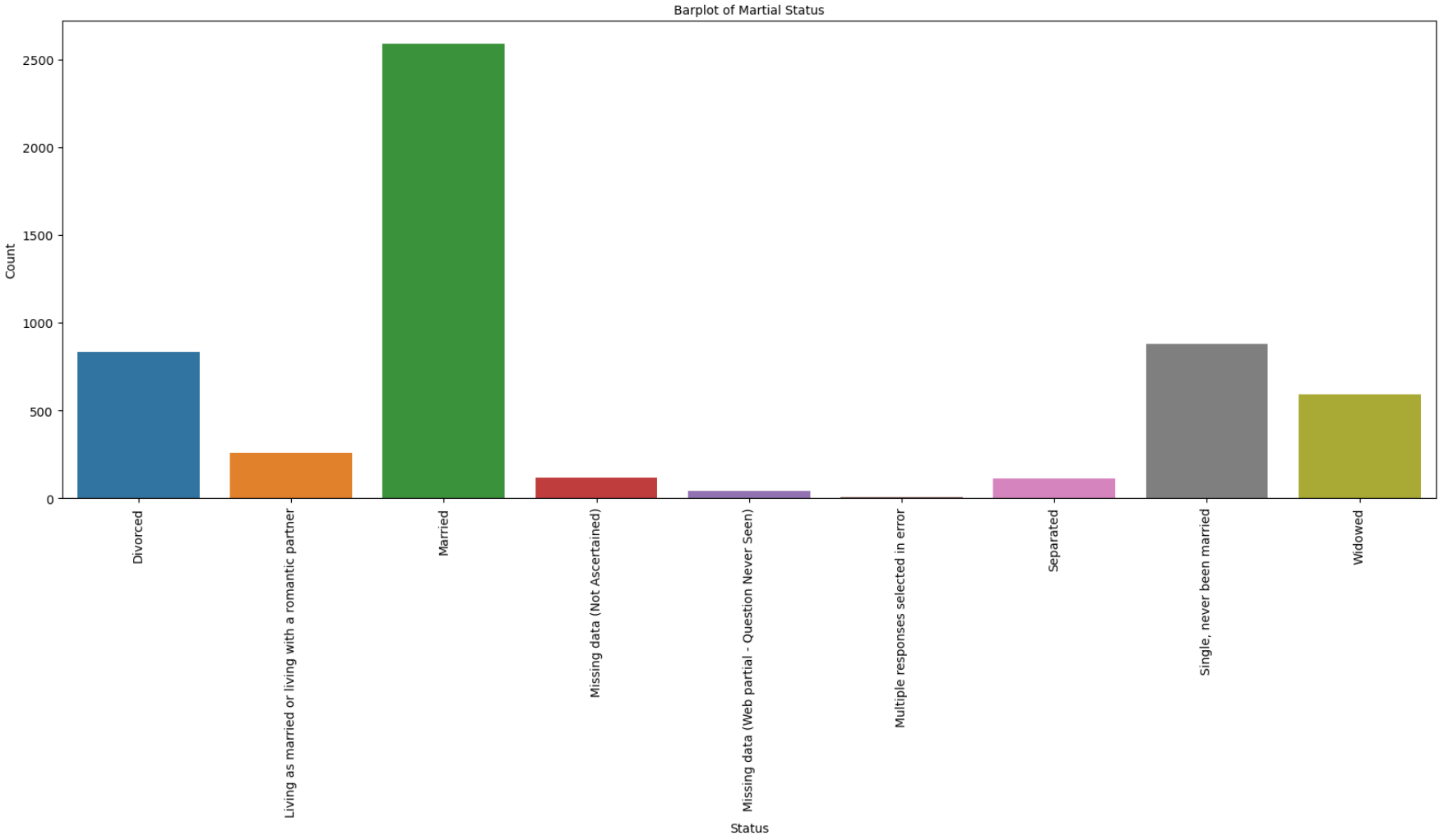
Exploring information about the marital status

In [152]:
```
#MaritalStatus: O2. What is your marital status?
#Variable Name: MaritalStatus

HINTS_MaritalStatus = HINTS_data['MaritalStatus'].value_counts(normalize=True) * 100
print(HINTS_MaritalStatus)
```

```
Married                                                 47.591026
Single, never been married                              16.219198
Divorced                                                15.318132
Widowed                                                 10.904744
Living as married or living with a romantic partner     4.762780
Missing data (Not Ascertained)                          2.206694
Separated                                               2.059581
Missing data (Web partial - Question Never Seen)        0.790732
Multiple responses selected in error                    0.147113
Name: MaritalStatus, dtype: float64
```

In [153]:
```
plt.figure(figsize = (20,7))
sns.countplot(HINTS_data['MaritalStatus'])
plt.title("Barplot of Martial Status", fontsize = 10)
plt.xlabel("Status", fontsize = 10)
plt.xticks(rotation=90)
plt.ylabel("Count", fontsize = 10)
```

Out[153]:
```
Text(0, 0.5, 'Count')
```



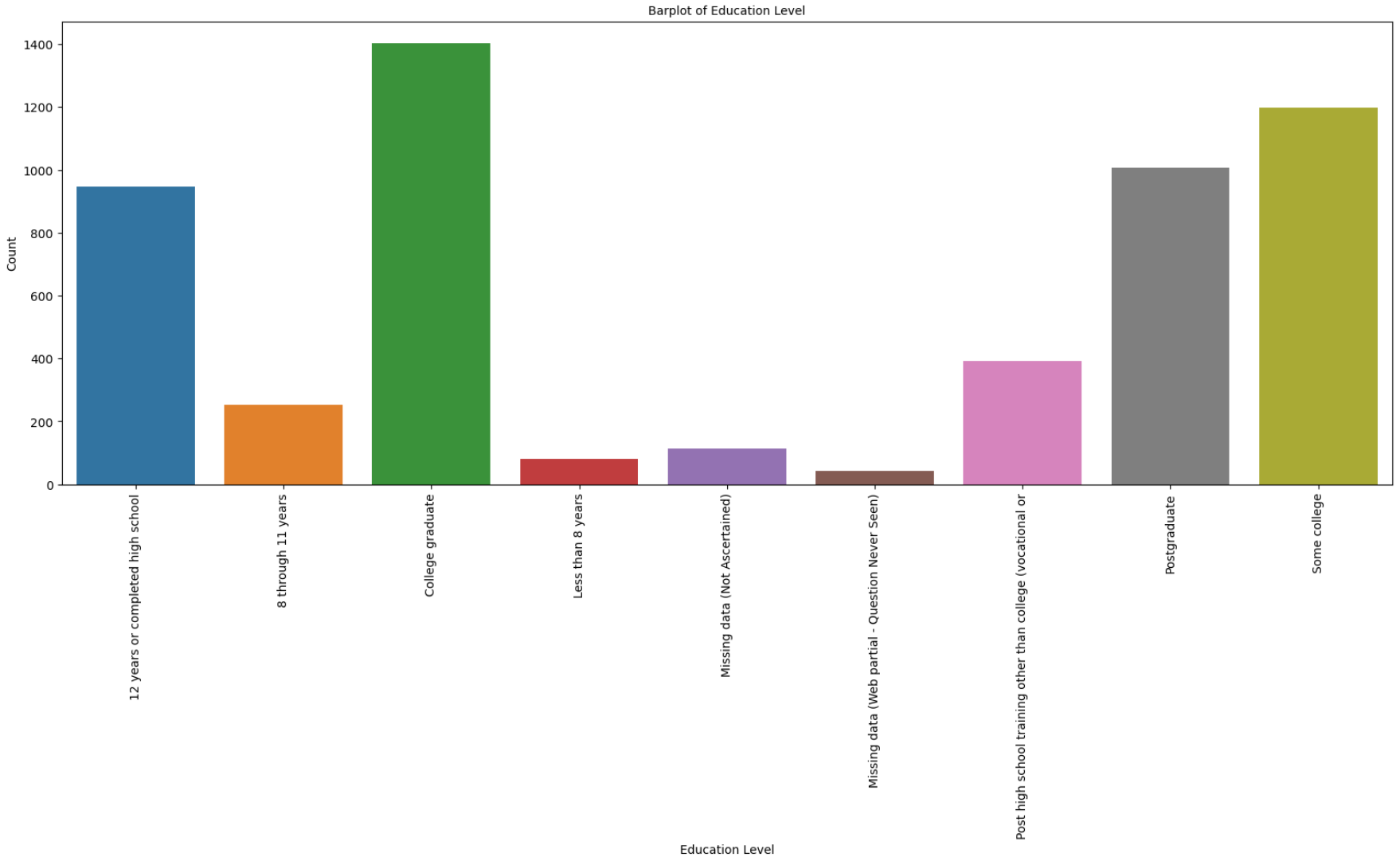
Exploring Information about the education status

In [154]:
```
#Education: O3. What is the highest grade or level of schooling you completed?
#Variable Name: Education

HINTS_Education = HINTS_data['Education'].value_counts(normalize=True) * 100
print(HINTS_Education)
```

```
College graduate                                              25.781537
Some college                                                  22.048547
Postgraduate                                                  18.536227
12 years or completed high school                             17.396102
Post high school training other than college (vocational or    7.208533
8 through 11 years                                            4.670835
Missing data (Not Ascertained)                                2.096359
Less than 8 years                                             1.471129
Missing data (Web partial - Question Never Seen)              0.790732
Name: Education, dtype: float64
```

In [155]:
```
plt.figure(figsize = (20,7))
sns.countplot(HINTS_data['Education'])
plt.title("Barplot of Education Level", fontsize = 10)
plt.xlabel("Education Level", fontsize = 10)
plt.xticks(rotation=90)
plt.ylabel("Count", fontsize = 10)
```

Out[155]:
```
Text(0, 0.5, 'Count')
```



Exploring the dataset to get insights about the diagnosis of cancer

In [156]:
```
#EverHadCancer: M1. Have you ever been diagnosed as having cancer?
##Variable Name: EverHadCancer

HINTS_Cancerdiagnosis = HINTS_data['EverHadCancer'].value_counts(normalize=True) * 100
print(HINTS_Cancerdiagnosis)
```

```
No                                                 82.199338
Yes                                                15.741081
Missing data (Not Ascertained)                      1.324016
Missing data (Web partial - Question Never Seen)    0.735565
Name: EverHadCancer, dtype: float64
```

Exploring the dataset to get insights about the diagnosis of hypertension

In [157]:
```
#MedConditions_HighBP:
##F6b. Has a doctor or other health professional ever told you that you had high blood pressure or hypertension?
#Variable Name: MedConditions_HighBP

HINTS_HighBP = HINTS_data['MedConditions_HighBP'].value_counts(normalize=True) * 100
print(HINTS_HighBP)
```

```
No                                                 53.641045
Yes                                                43.949982
Missing data (Not Ascertained)                      2.225083
Missing data (Web partial - Question Never Seen)    0.183891
Name: MedConditions_HighBP, dtype: float64
```

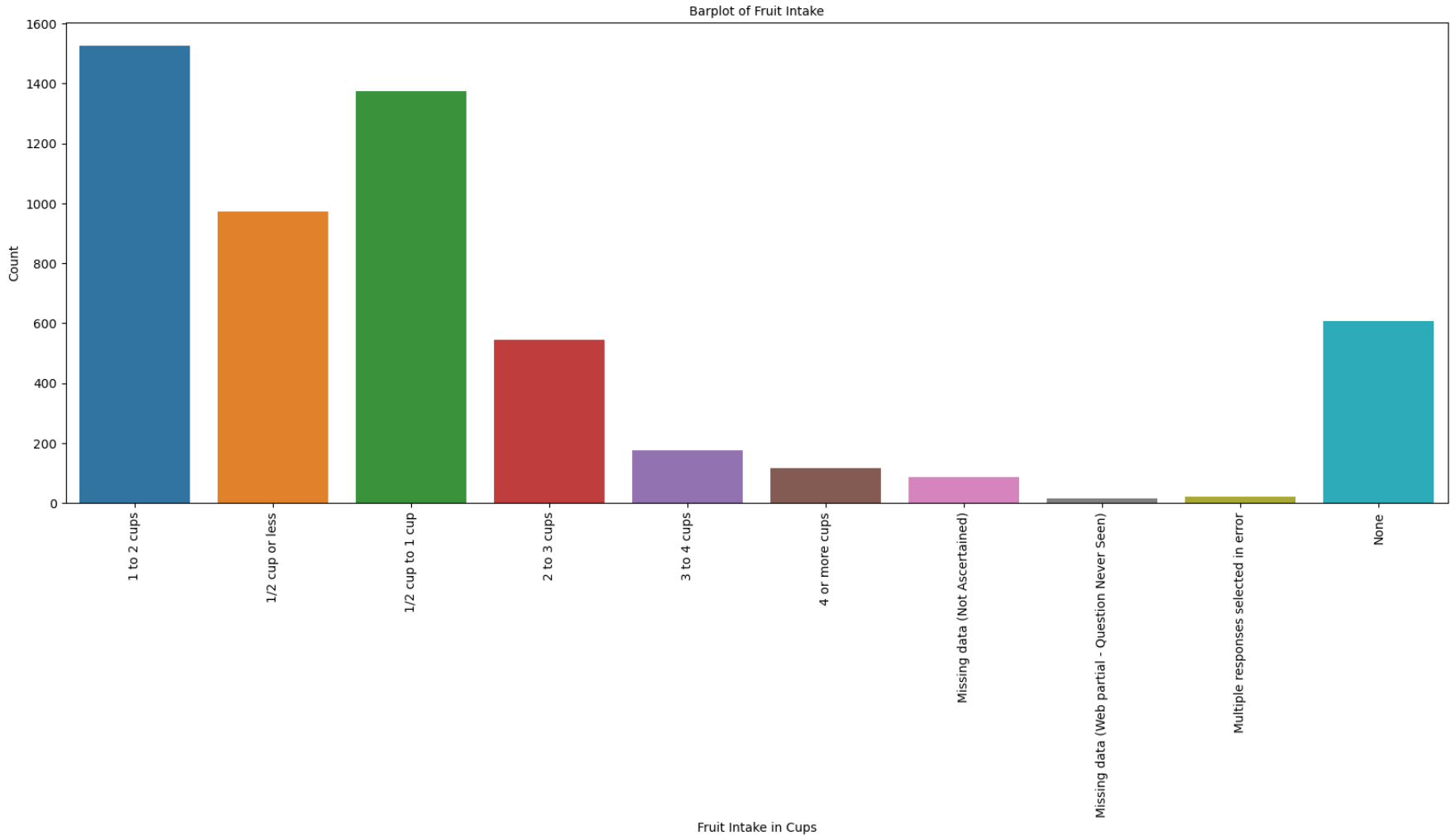
Exploring the dataset to get insights about the fruits intake of the participants

In [158]:
```
## Fruit: G1. About how many cups of fruit (including 100% pure fruit juice) do you eat or drink each day?
##Variable Name: Fruit

HINTS_Fruits = HINTS_data['Fruit'].value_counts(normalize=True) * 100
print(HINTS_Fruits)
```

```
1 to 2 cups                                        28.080177
1/2 cup to 1 cup                                   25.248253
1/2 cup or less                                    17.910997
None                                               11.162192
2 to 3 cups                                        10.003678
3 to 4 cups                                         3.218095
4 or more cups                                      2.114748
Missing data (Not Ascertained)                      1.581464
Multiple responses selected in error                0.404561
Missing data (Web partial - Question Never Seen)    0.275837
Name: Fruit, dtype: float64
```

In [159]:
```
plt.figure(figsize = (20,7))
sns.countplot(HINTS_data['Fruit'])
plt.title("Barplot of Fruit Intake", fontsize = 10)
plt.xlabel("Fruit Intake in Cups", fontsize = 10)
plt.xticks(rotation=90)
plt.ylabel("Count", fontsize = 10)
```

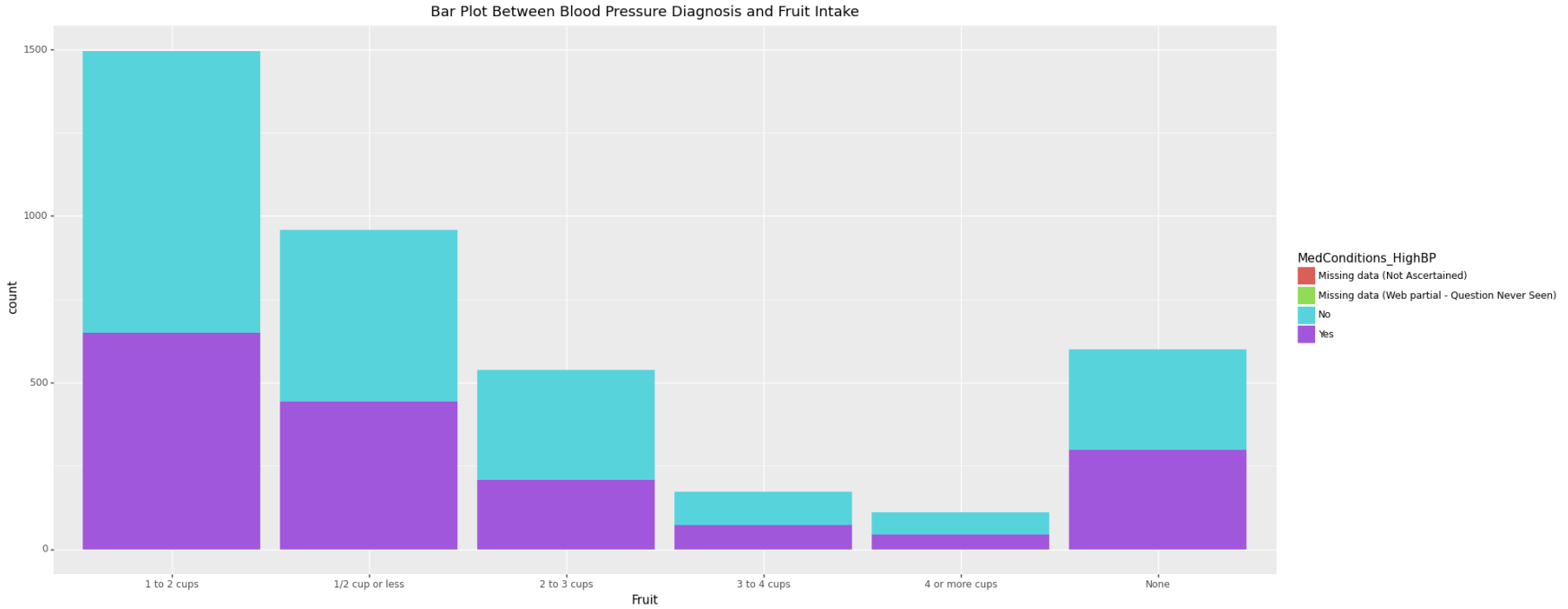
Out[159]:
```
Text(0, 0.5, 'Count')
```



Visualization Between Diagnosis of BP and Fruit Intake

In [160]:
```
## Slicing the Dataset as per the requirement

BP = HINTS_data[['MedConditions_HighBP', 'Fruit']]
BP1 = BP[BP["MedConditions_HighBP"].isin(['Yes', 'No'])]
BP2 = BP1[BP1["Fruit"].isin(['None', '1/2 cup or less',
                             '1/2 cup to 1 cup ', '1 to 2 cups', '2 to 3 cups', '3 to 4 cups', '4 or more cups'])]
```

In [161]:
```
## Creating a ggplot

ggplot(BP2, aes(x='Fruit',fill = 'MedConditions_HighBP')) + \
    geom_bar(stat = 'count') + \
    ggtitle("Bar Plot Between Blood Pressure Diagnosis and Fruit Intake") + \
    theme(figure_size=(20, 9))
```



Out[161]: <ggplot: (8777682958695)>

Exploring the dataset to get insights about the smoking patterns

In [162]:
```
#Smoke100: K1. Have you smoked at least 100 cigarettes in your entire life?
#Variable Name: Smoke100

HINTS_Smoke = HINTS_data['Smoke100'].value_counts(normalize=True) * 100
print(HINTS_Smoke)
```

```
No                                                  60.573740
Yes                                                 37.734461
Missing data (Not Ascertained)                       1.121736
Missing data (Web partial - Question Never Seen)     0.570063
Name: Smoke100, dtype: float64
```

Insights Regarding the Dataset using EDA

- Half of the participants were married.
- Around 66% of the participants had education more than 12 years or completed high school.
- 82% of the participants were never diagnosed with any kind of cancer. The data regarding the diagnosis of cancer is unbalanced.
- About 43% of the participants were told that they have high BP/hypertension. This subset of the population can be considered to have high BP. Since the data isnt skewed for this parameter we can assess predictors or classify the behaviours of those who have high BP v/s who doesnt have high BP using other parameters such as physical activity, diet and online information seeking.
- The federal guidelines recommend that adults eat at least 1½ to 2 cups per day. Interestingly 69% of the sample was meeting this recommendation.
- From the barplot of blood pressure diagnosis and fruit intake we can see that the intake of fruits were more who did not get diagnosed with high BP. Further analysis can be done to find the correlations. The same can be done with those who were diagnosed with cancer or not.
- 37% of the participants have at least smoked 100 cigarettes in your entire life.

Missing Data Findings

- The organization states that the dataset was updated in March 2021 after it was discovered that 35 variables were affected by coding errors associated with missing values for data from participants who completed the survey online as part of the push-to-web pilot study. For some of the open-ended questions in the survey, invalid skips (-9) were coded to 0 instead of -9, and other related, derived variables required minor revisions to missing value assignments.

In [163]:
```
# Looking for null value in the data
HINTS_data.isnull().sum()
```

```
WhereSeekHealthInfo                0
WhoLookingFor                      0
LotOfEffort                        0
Frustrated                         0
ConfidentGetHealthInf              0
TrustDoctor                        0
TrustFamily                        0
TrustGov                           0
TrustCharities                     0
TrustReligiousOrgs                 0
StrongNeedHealthInfo               0
StrongNeedHealthInfo_OS            0
SeekCancerInfo                     0
UseInternet                        0
Internet_DialUp                    0
Internet_BroadBnd                  0
Internet_Cell                      0
Internet_WiFi                      0
InternetCancerInfoSelf             0
WhereUseInternet Home              0
```

Data Cleaning Needs Insights

- The missing data such as TG1_FINWT1 is the sampling weights of only the paper based records. This is a parameter not asked in the questionnaire, so therefore for now I believe it is not necessary to focus on it. the data set has been cleaned and pre-processed by the organization. All other important parameters does not contain any null value. In the dataset the missing values/ uncertainities and errors are coded for example as we saw in the maritalstatus. We can drop the rows of the missing/uncertains values if the missing values are completely at random if we do an advanced data analysis.
- Since most of the variables which have missing values are categorical, we can't use mean or median but we can impute the Most Frequent Value to fill in the missing variables.