# EDA and Preprocessing : Brain MRI Segmentation

## About Dataset

we are using **"LGG Segmentation Dataset"**

This dataset contains brain MR images together with manual FLAIR abnormality segmentation masks.
The images were obtained from The Cancer Imaging Archive (TCIA).

They correspond to 110 patients

we also have patient detail for example tumor_tissue_site, tumor location, RPPACluster and gender, age, age_at_initial_pathologic and ethnicity etc.

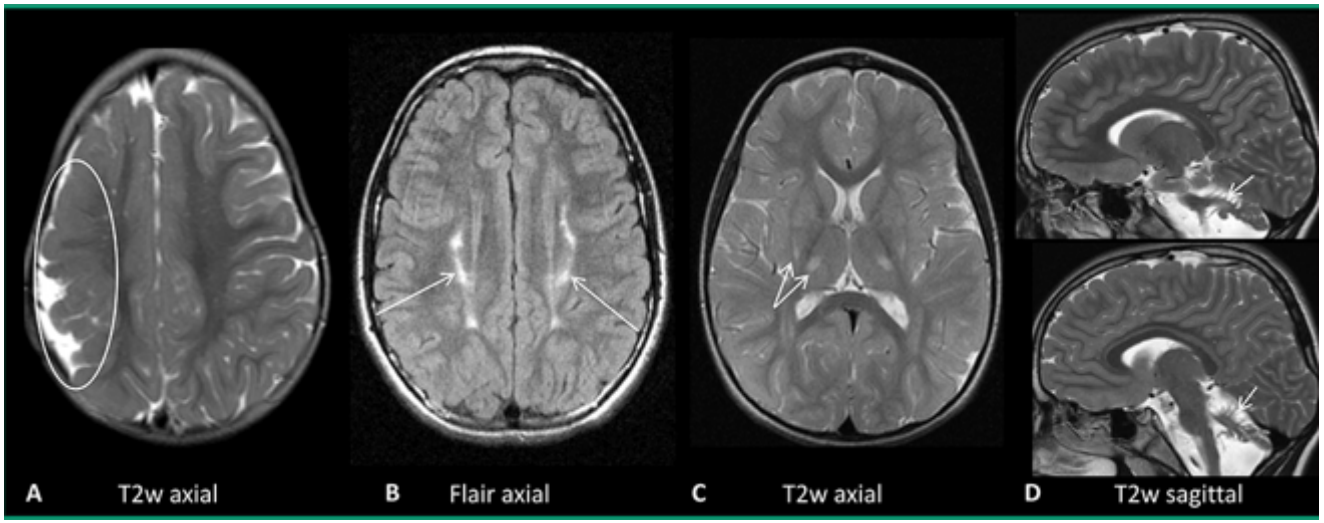
so as task of EDA we will mainly focus on image data for the sementic segmentation task,
and do understand what type of information these images contain.

## Domain knowledge : Brain

In simple terminology, there are multiple layers in brain and these tumors could be in any layer of brain, based on layer we known this tumor with different name.

For capturing these tumor info in MRI scan there are different-2 technique uses for different different types of layer.

for example, FLAIR is an MRI sequence(/technique), used in brain imaging to suppress cerebrospinal fluid (CSF) effects on the image, so as to bring out the periventricular hyperintense lesions,
for more info : - [click me (https://case.edu/med/neurology/NR/MRI%20Basics.htm#:~:text=The%20most%20common%20MRI%20sequences,longer%20TE%20and%20TR%20times. )](https://case.edu/med/neurology/NR/MRI%20Basics.htm#:~:text=The%20most%20common%20MRI%20sequences,longer%20TE%20and%20TR%20times.)




in above it showing 4 technique (indication through circle and arrows).

in out dataset, through FLAIR AXIAL data has been collect. which collect tumor information of middle part of brain

**Note:** These information is based on my understanding to make an easy to understand this project, please understand i am not any type of medical expert

## 1. Dependencies

In [48]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from os import path
import cv2
import os
import re
import random
import pdb
import seaborn as sns
```

In [49]:
```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleW

```

```
'wget' is not recognized as an internal or external command,
operable program or batch file.
```

In [50]:
```
!unzip "/content/archive.zip"
```

```
'unzip' is not recognized as an internal or external command,
operable program or batch file.
```

## 2. data loading

In [51]:
```
df = pd.read_csv("kaggle_3m/data.csv")
```

In [52]:
```
1 df.head()
```

Out[52]:

| | Patient | RNASeqCluster | MethylationCluster | miRNACluster | CNCluster | RPPACluster | OncosignCluster | COCCluster | histological_type | neo |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | TCGA_CS_4941 | 2.0 | 4.0 | 2 | 2.0 | NaN | 3.0 | 2 | 1.0 | |
| **1** | TCGA_CS_4942 | 1.0 | 5.0 | 2 | 1.0 | 1.0 | 2.0 | 1 | 1.0 | |
| **2** | TCGA_CS_4943 | 1.0 | 5.0 | 2 | 1.0 | 2.0 | 2.0 | 1 | 1.0 | |
| **3** | TCGA_CS_4944 | NaN | 5.0 | 2 | 1.0 | 2.0 | 1.0 | 1 | 1.0 | |
| **4** | TCGA_CS_5393 | 4.0 | 5.0 | 2 | 1.0 | 2.0 | 3.0 | 1 | 1.0 | |

In [53]:
```
1 df.shape
```

Out[53]:
```
(110, 18)
```

In [54]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110 entries, 0 to 109
Data columns (total 18 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Patient                    110 non-null    object
 1   RNASeqCluster              92 non-null     float64
 2   MethylationCluster         109 non-null    float64
 3   miRNACluster               110 non-null    int64
 4   CNCluster                  108 non-null    float64
 5   RPPACluster                98 non-null     float64
 6   OncosignCluster            105 non-null    float64
 7   COCCluster                 110 non-null    int64
 8   histological_type          109 non-null    float64
 9   neoplasm_histologic_grade  109 non-null    float64
 10  tumor_tissue_site          109 non-null    float64
 11  laterality                 109 non-null    float64
 12  tumor_location             109 non-null    float64
 13  gender                     109 non-null    float64
 14  age_at_initial_pathologic  109 non-null    float64
 15  race                       108 non-null    float64
 16  ethnicity                  102 non-null    float64
 17  death01                    109 non-null    float64
dtypes: float64(15), int64(2), object(1)
memory usage: 15.6+ KB
```

In [56]:
```
1 # main_dir = "/content/kaggle_3m/"
2 # for i in os.listdir(main_dir)[:-1]:
3 #     sub_dir = main_dir+i+"/"
4 #     if path.isdir(i+"/"+sub_dir):
5 #         for j in os.listdir(sub_dir):
6 #             print("\'{}\'".format(sub_dir+j))
7
8
```

In [57]:
```
1 # path.isfile('kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_19960909_1.tif')
```

In [58]:

```
def return_file_name(root_dir):
    img_url = []
    mask_img_url = []
    df = pd.DataFrame([])
    for i in os.listdir(root_dir)[2:-3]:
        sub_dir = root_dir+i+"/"
        # pdb.set_trace()
        if path.isdir(sub_dir):
            for j in os.listdir(sub_dir):
    #             img_dir.append(str(sub_dir+j))

                if "mask" in j :
                    mask_img_url.append(str(sub_dir+j))

    # pdb.set_trace()
    img_url = [re.sub("_mask","", i) for i in mask_img_url]

    df["image"] = img_url
    df["mask"] = mask_img_url

    return  df

main_dir = "kaggle_3m/"
data = return_file_name(main_dir)

```

In [59]:
```
"""
    checking all file path
"""
b = True
for i in range(data.shape[0]):
    if not path.isfile(data["image"][i]):
        b = False
    elif not path.isfile(data["mask"][i]):
        b = False

print(b)
```

```
True
```

In [60]:
```
data.head()
```

Out[60]:

| | image | mask |
|---|---|---|
| **0** | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... |
| **1** | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... |
| **2** | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... |
| **3** | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... |
| **4** | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... | kaggle_3m/TCGA_CS_4941_19960909/TCGA_CS_4941_1... |

In [61]:
```
data.shape
```

Out[61]:
```
(3725, 2)
```

In [ ]:
```

```

# 3. EDA

in this stage, our approach would be to analyze the data using visual techniques and discover trends, patterns, or to check assumptions with the

help of statistical summary and graphical representations

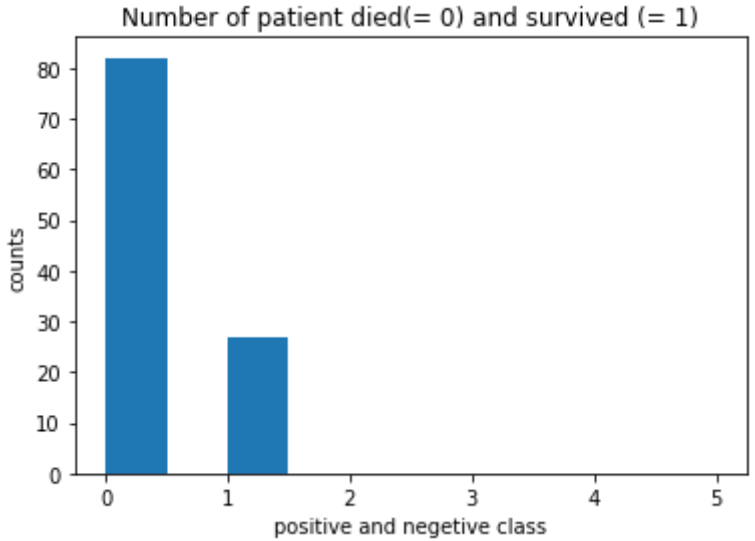
***exploring patient medical related information***

In [62]:
```
1 df.sample()
```

Out[62]:

| | Patient | RNASeqCluster | MethylationCluster | miRNACluster | CNCluster | RPPACluster | OncosignCluster | COCCluster | histological_type | ne |
|---|---|---|---|---|---|---|---|---|---|---|
| **33** | TCGA_DU_7010 | 2.0 | 5.0 | 2 | 2.0 | 1.0 | 2.0 | 2 | 1.0 | |

In [63]:
```
1 plt.hist(df["death01"], range= (0,5))
2 plt.title("Number of patient died(= 0) and survived (= 1)")
3 plt.xlabel("positive and negetive class")
4 plt.ylabel("counts")
5 plt.show()
```



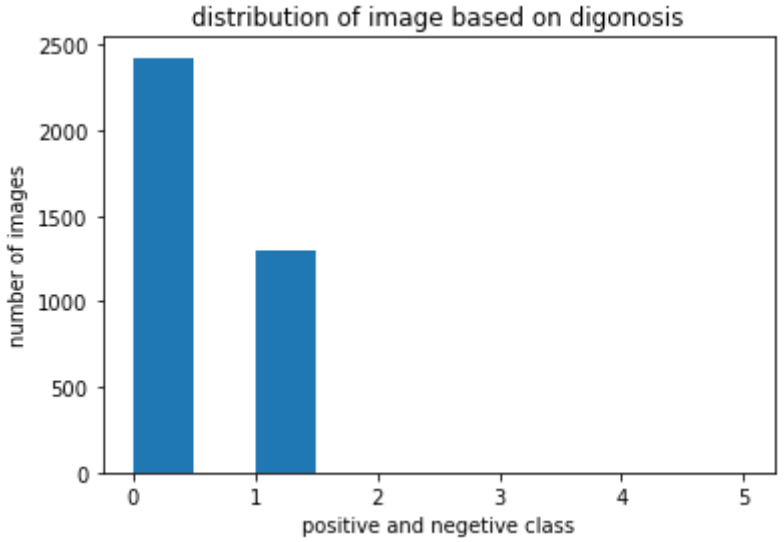

**observation** we have 110 patients in which 80 patient survived succussfuly and 30 patient died due to tumor.

In [64]:

```
#https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.apply.html

def check_tumor(mask_img):
    """
        here we checking in each mask image, does any value greater than 0
        here 0 indicate black color, if image contain only 0 than it means
        there is no tumor region in actual image.
    """

    val = np.max(cv2.imread(mask_img))
    return 1 if val>0 else 0


data["digonosis"] = data["mask"].apply(lambda m : check_tumor(m))

```

In [65]:

```
plt.title("distribution of image based on digonosis")
plt.hist(data["digonosis"], range = (0,5))
plt.xlabel("positive and negetive class")
plt.ylabel("number of images")

plt.show()
```



**observation**

there are 2400 image which belongs to negetive class(does not contains any tumor information) and 1300 image belongs to positive class

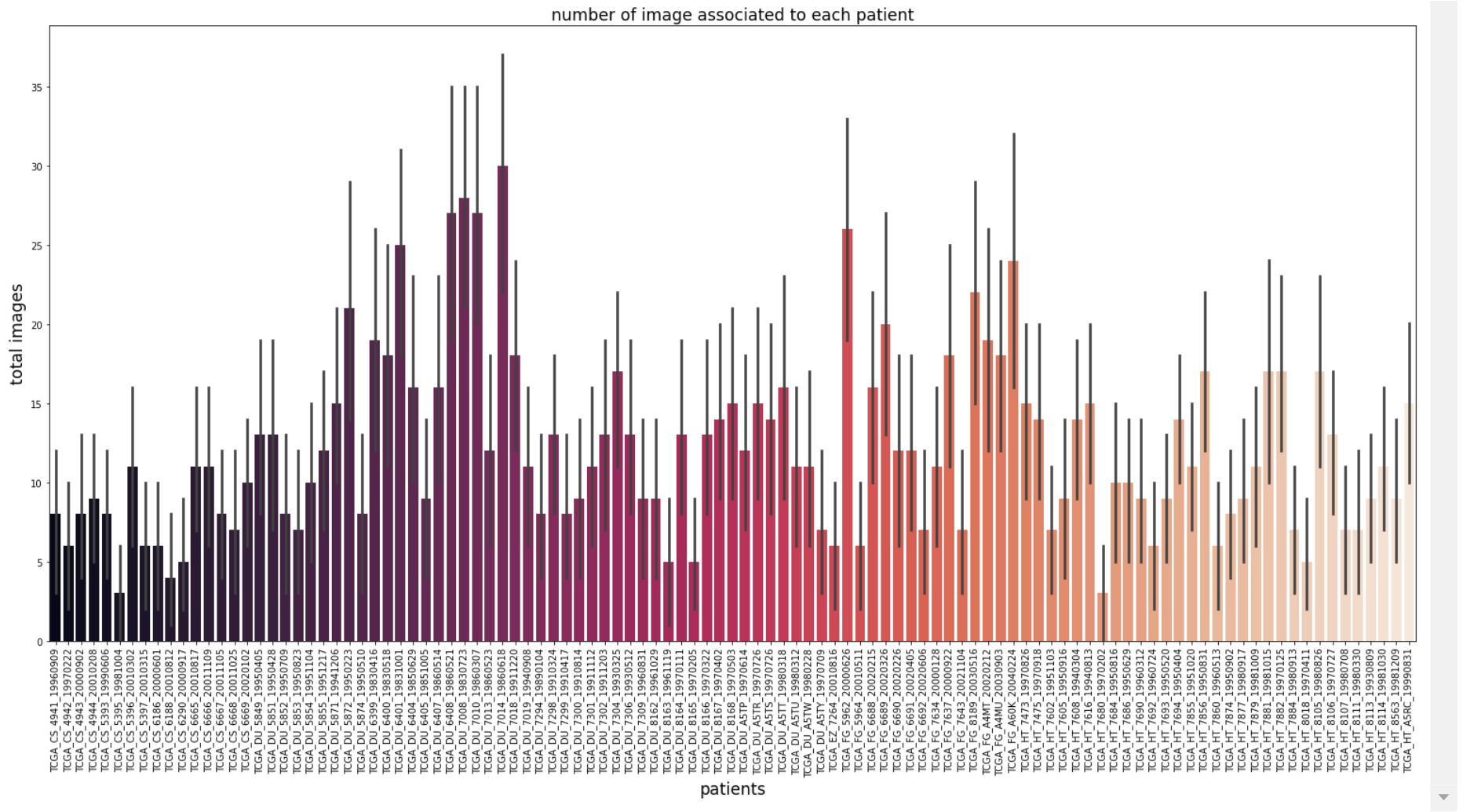
**ques - how many number of images associated to each patient**

In [66]:

```
def extract_name(url):
    """
        returning patient name/id
    """
    return re.findall("TCGA[a-zA-Z0-9_]*", url)[0]

data["patient"] = data["image"].apply(lambda x : extract_name(x))


```

In [67]:

```
# ques - how many number of image associated to each patient

fig, ax = plt.subplots(figsize=(26,12))
sns.barplot(data=data, x="patient", y="digonosis",estimator =np.sum,
            ax=ax, palette="rocket",)#.set_title("total images associate to each indivisual patient")

plt.title("number of image associated to each patient", size = "xx-large")
plt.xlabel("patients", size = "xx-large")
plt.xticks(rotation=90)
plt.ylabel("total images", size = "xx-large")
plt.show()
```




In [68]:
```
1 data.sample()
```

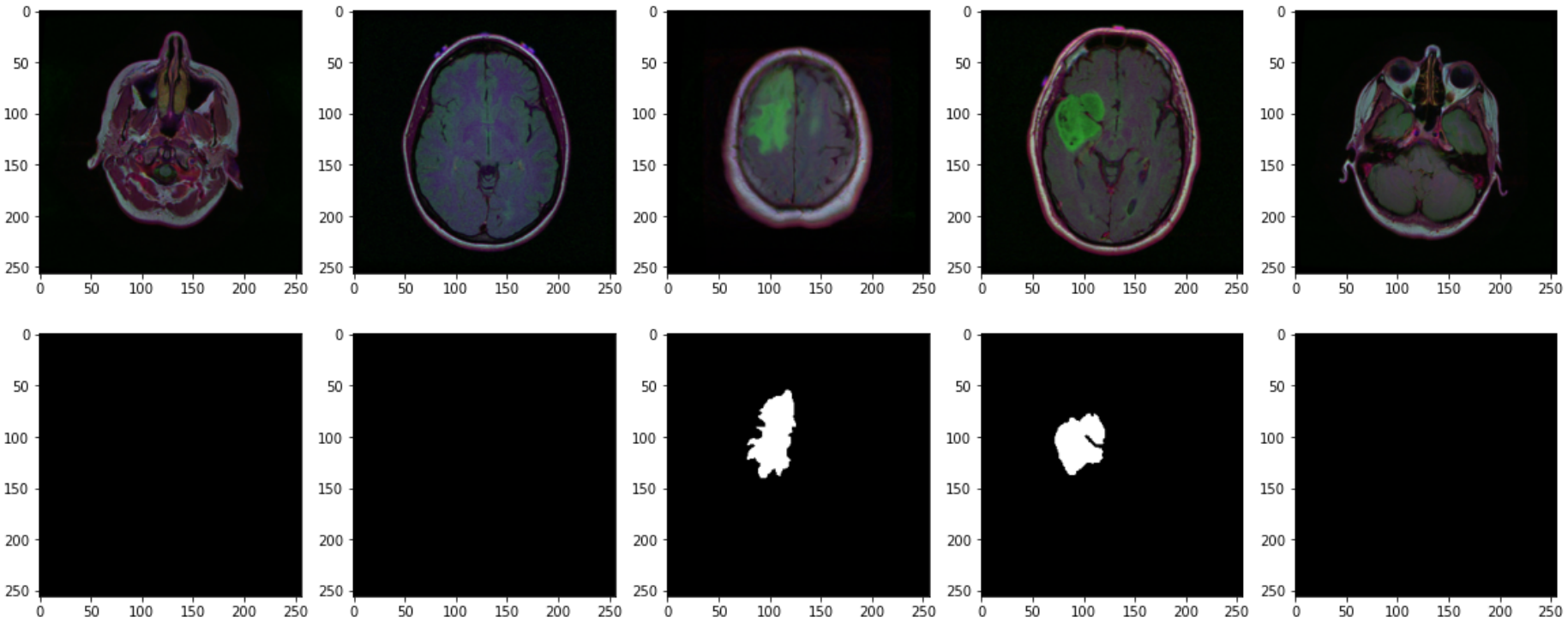
Out[68]:

| | image | mask | digonosis | pati |
|---|---|---|---|---|
| **2932** | kaggle_3m/TCGA_HT_7475_19970918/TCGA_HT_7475_1... | kaggle_3m/TCGA_HT_7475_19970918/TCGA_HT_7475_1... | 0 | TCGA_HT_7475_19970 |

**ques - how actual images and its associated mask image looks like**

In [69]:

```
# plt1 = plt.figure(figsize=(8,8))
r = 2
c = 5
fig,axis = plt.subplots(r, c, figsize=(20,8))


for row in range(r):
    for i in range(c):
        idx = random.randint(0,data.shape[0])
        if row==0:
            axis[row,i].imshow(cv2.imread(data["image"][idx]))
            axis[row+1,i].imshow(cv2.imread(data["mask"][idx]))

```



**here:** In first row, we have MRI images and in second row we have masked images which indicates region of tumor

# 4. Data Preprocessing

In data this stage: we will analyse, filter and transforming data so that
algorithm can understand and work with the preprocessed data.

So, as the popular saying goes, "if garbage goes in, garbage comes out".
model will only work successfully if data going into machine is high quality.

when we get data from real world scenarios, these is high chances that it contains noice
data and missing value.

in this case, we are using the LGG Segmentation Dataset.

## 4.1 Data cleaning and filtering

while doing EDA we observe there are some images which doesn't contain any information(black image) so we will remove those images

In [70]:
```
1 data.shape
```

Out[70]:
```
(3725, 4)
```

In [71]:

```
def garbage_img_prepross(df):
    """
        finding image which doesn't have much infomation
        here we choose 30 as pixel value threshhold  all img which maximum pixel value
        is less that 30 considered to be garbage image
    """
    thres = 30
    temp_img = []

    for i in df["image"]:
        val = np.max(cv2.imread(i))
        if val <thres:
            temp_img.append(i)

    temp_img = np.array(temp_img)
    df = df[~df["image"].isin(temp_img)]

    return df, temp_img

data, temp_img = garbage_img_prepross(data)


print("we got total {} image which doesn't contains information, so in order to prepross, we removed it from data".f
```

```
we got total 73 image which doesn't contains information, so in order to prepross, we removed it from data
```

In [72]:

```
data.shape
```

Out[72]:

```
(3652, 4)
```

In [73]:

```
# visulization garbase images which having less information (max picxel val < 30)

r = 18
c = 4
fig,axis = plt.subplots(r, c, figsize=(100,300))
p = 0
for j in range(r):
    for  i in range(c):
        axis[j,i].imshow(cv2.imread(temp_img[63-p]))
        p +=1


```

these image in our train data doesn't have any information so as next move we will remove it from our train data so that our model learn better

**NOW**, going ahead, we resizes of all images(will done in data pipeline), remove garbage images, now our data is ready for next task - ( data pipeline, augmentation , modeling)

## 5. Modeling

next stage .....