

# Brain Stroke Analysis Using Python

I'm thrilled to share the successful completion of a groundbreaking Brain Stroke Analysis project! Here are the key highlights of my work:

- 🔍 **Null Value Handling:** Identified and meticulously addressed null values within the dataset to ensure impeccable data integrity and accuracy, laying a robust foundation for further analysis.
- ✅ **In-depth Analysis:** Conducted a thorough and insightful analysis to decipher patterns and trends related to brain stroke occurrences, providing valuable insights into this critical healthcare domain.
- 🔧 **Data Manipulation Expertise:** Employed advanced data manipulation techniques to fill missing values and optimize data quality, enhancing the reliability and usefulness of the dataset.
- 🚀 **Python Libraries Mastery:** Leveraged powerful Python libraries including Pandas and NumPy for seamless data preprocessing and cleaning, streamlining the analysis process and maximizing efficiency.
- 📊 **Visualization Proficiency:** Utilized state-of-the-art visualization tools such as Matplotlib and Seaborn to craft visually engaging charts and graphs, effectively communicating complex insights to stakeholders.



# Import Library

```
In [1]: import pandas as pd
```

```
In [2]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
```

C:\Users\Syed Arif\anaconda3\lib\site-packages\scipy\\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.25.1

warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}")

## Uploading Csv file

```
In [3]: df = pd.read_csv(r"C:\Users\Syed Arif\Desktop\healthcare-dataset-stroke-data.csv")
```

## Data Preprocessing

### .head()

head is used to show the first 5 rows in the dataset

```
In [4]: df.head()
```

Out[4]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_
0	9046	Male	67.0	0	1	Yes	Private	Urban	
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	
2	31112	Male	80.0	0	1	Yes	Private	Rural	
3	60182	Female	49.0	0	0	Yes	Private	Urban	
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	

### .tail()

tail is used to show rows by Descending order

```
In [5]: df.tail()
```

```
Out[5]:
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	a
5105	18234	Female	80.0	1	0	Yes	Private	Urban	
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	
5108	37544	Male	51.0	0	0	Yes	Private	Rural	
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	

## .shape

It show the total no of rows & Column in the dataset

```
In [6]: df.shape
```

```
Out[6]: (5110, 12)
```

## .Columns

It show the no of each Column

```
In [7]: df.columns
```

```
Out[7]: Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',  
              'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',  
              'smoking_status', 'stroke'],  
             dtype='object')
```

## .dtypes

This Attribute show the data type of each column

```
In [8]: df.dtypes
```

```
Out[8]: id                int64
gender                object
age                  float64
hypertension          int64
heart_disease          int64
ever_married          object
work_type              object
Residence_type        object
avg_glucose_level     float64
bmi                   float64
smoking_status        object
stroke                int64
dtype: object
```

## **.unique()**

In a column, It show the unique value of specific column.

```
In [9]: df["work_type"].unique()
```

```
Out[9]: array(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_worked'],
              dtype=object)
```

## **.nunique()**

It will show the total no of unique value from whole data frame

```
In [10]: df.nunique()
```

```
Out[10]: id                5110
gender                 3
age                   104
hypertension           2
heart_disease           2
ever_married           2
work_type              5
Residence_type         2
avg_glucose_level     3979
bmi                    418
smoking_status         4
stroke                 2
dtype: int64
```

## **.describe()**

It show the Count, mean , median etc

```
In [11]: df.describe()
```

```
Out[11]:
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	
<b>count</b>	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110
<b>mean</b>	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0
<b>std</b>	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0
<b>min</b>	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0
<b>25%</b>	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0
<b>50%</b>	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0
<b>75%</b>	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0
<b>max</b>	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1

## .value\_counts

It Shows all the unique values with their count

```
In [12]: df["work_type"].value_counts()
```

```
Out[12]: Private          2925
Self-employed    819
children         687
Govt_job         657
Never_worked     22
Name: work_type, dtype: int64
```

## .isnull()

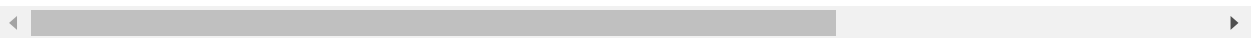
It shows the how many null values

```
In [13]: df.isnull()
```

Out[13]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	a
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...
5105	False	False	False	False	False	False	False	False	
5106	False	False	False	False	False	False	False	False	
5107	False	False	False	False	False	False	False	False	
5108	False	False	False	False	False	False	False	False	
5109	False	False	False	False	False	False	False	False	

5110 rows × 12 columns



## .info()

To Show Data type of each column

```
In [14]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   id                    5110 non-null  int64  
 1   gender                5110 non-null  object  
 2   age                  5110 non-null  float64 
 3   hypertension          5110 non-null  int64  
 4   heart_disease         5110 non-null  int64  
 5   ever_married          5110 non-null  object  
 6   work_type             5110 non-null  object  
 7   Residence_type        5110 non-null  object  
 8   avg_glucose_level     5110 non-null  float64 
 9   bmi                   4909 non-null  float64 
10   smoking_status        5110 non-null  object  
11   stroke                5110 non-null  int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

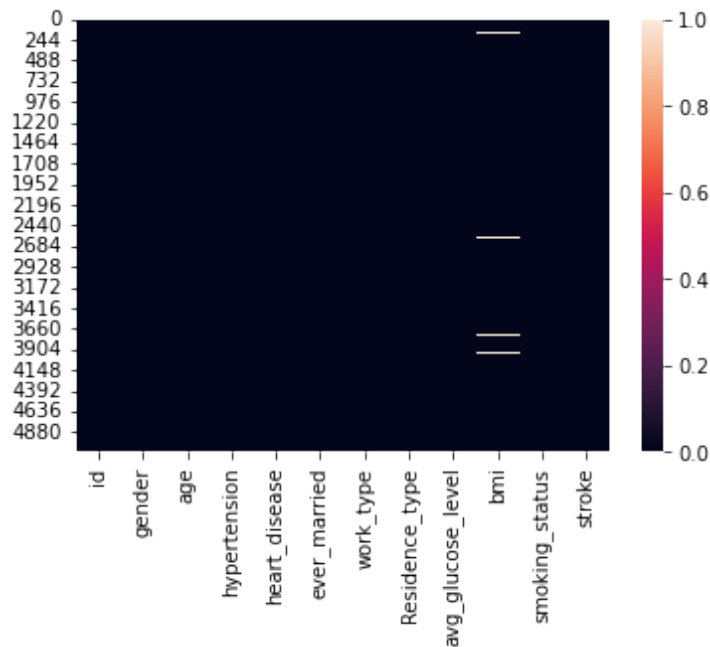
## Is there any Null value present in any Column ? Show with heatmap

```
In [15]: df.isnull().sum()
```

```
Out[15]: id                0
gender                0
age                  0
hypertension          0
heart_disease         0
ever_married          0
work_type             0
Residence_type        0
avg_glucose_level     0
bmi                  201
smoking_status        0
stroke                0
dtype: int64
```

```
In [16]: sns.heatmap(df.isnull())
```

```
Out[16]: <AxesSubplot:>
```



## Change the Numeric values to categorical

```
In [25]: # Replace 'yes' with 1 and 'no' with 0 in the specified column
df['hypertension'] = df['hypertension'].replace({1: 'yes', 0: 'no'})
```

```
In [26]: df.dtypes
```

```
Out[26]: id                int64
gender              object
age                float64
hypertension        object
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level   float64
bmi                float64
smoking_status      object
stroke              int64
dtype: object
```

```
In [27]: # Replace 'yes' with 1 and 'no' with 0 in the specified column
df['heart_disease'] = df['heart_disease'].replace({1: 'yes', 0: 'no'})
```

```
In [28]: # Replace 'yes' with 1 and 'no' with 0 in the specified column
df['stroke'] = df['stroke'].replace({1: 'yes', 0: 'no'})
```

Bar Graph

```
In [29]: df
```

```
Out[29]:
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	a
0	9046	Male	67.0	no	yes	Yes	Private	Urban	
1	51676	Female	61.0	no	no	Yes	Self-employed	Rural	
2	31112	Male	80.0	no	yes	Yes	Private	Rural	
3	60182	Female	49.0	no	no	Yes	Private	Urban	
4	1665	Female	79.0	yes	no	Yes	Self-employed	Rural	
...	...	...	...	...	...	...	...	...	
5105	18234	Female	80.0	yes	no	Yes	Private	Urban	
5106	44873	Female	81.0	no	no	Yes	Self-employed	Urban	
5107	19723	Female	35.0	no	no	Yes	Self-employed	Rural	
5108	37544	Male	51.0	no	no	Yes	Private	Rural	
5109	44679	Female	44.0	no	no	Yes	Govt_job	Urban	

5110 rows × 12 columns



Count of Hear Disease

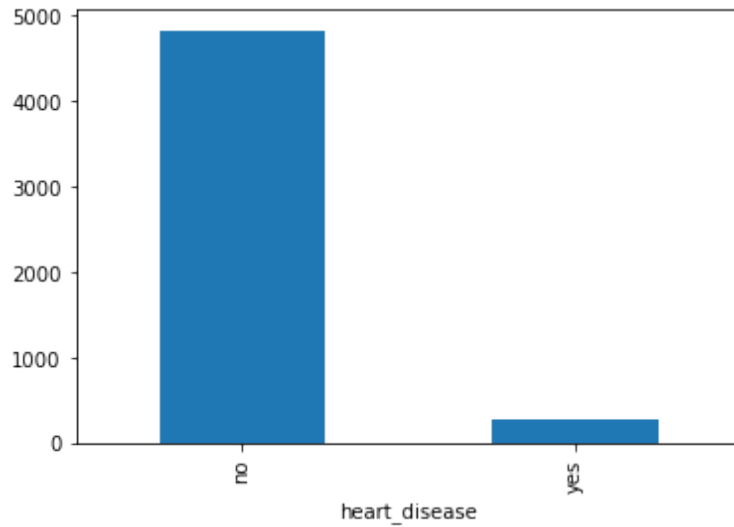


```
In [30]: df.groupby("heart_disease").heart_disease.count()
```

```
Out[30]: heart_disease  
no      4834  
yes      276  
Name: heart_disease, dtype: int64
```

```
In [31]: df.groupby("heart_disease").heart_disease.count().plot(kind = "bar")
```

```
Out[31]: <AxesSubplot:xlabel='heart_disease'>
```



Convert the age column into categorical

```
In [35]: # Apply conditions and replace values in the column  
df['age'] = df['age'].apply(lambda x: 'Younger' if x < 18 else ('Young Adult' if x < 30 else 'Older'))
```

In [36]: df

Out[36]:

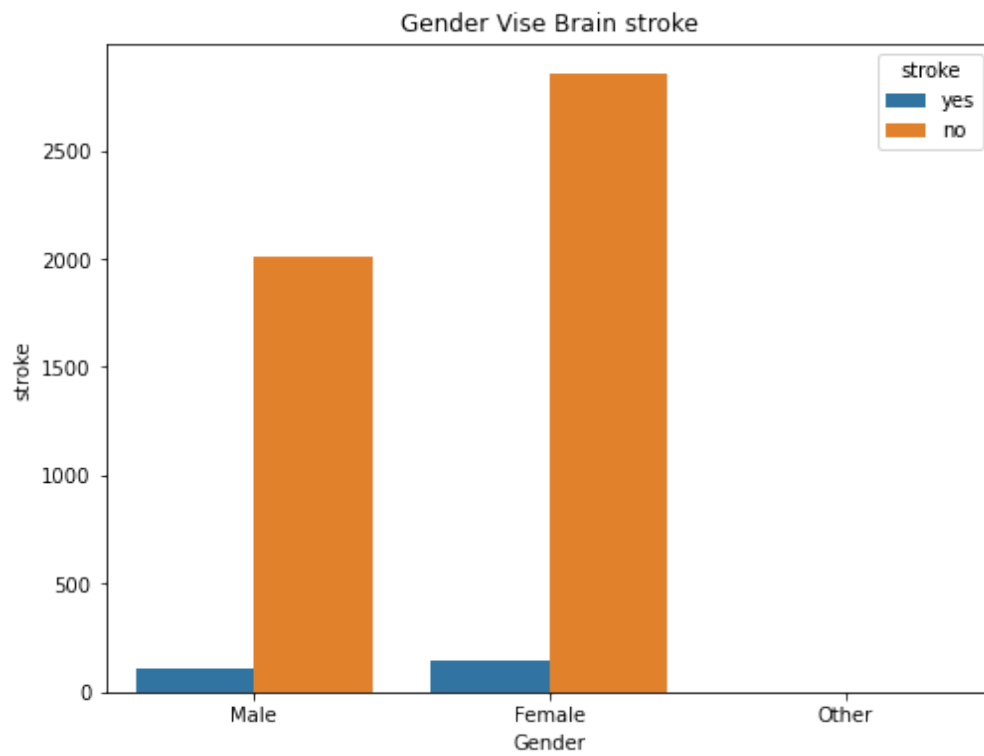
	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
0	9046	Male	Senior Adult	no	yes	Yes	Private	Urban
1	51676	Female	Senior Adult	no	no	Yes	Self-employed	Rural
2	31112	Male	Senior Adult	no	yes	Yes	Private	Rural
3	60182	Female	Adult	no	no	Yes	Private	Urban
4	1665	Female	Senior Adult	yes	no	Yes	Self-employed	Rural
...	...	...	...	...	...	...	...	...
5105	18234	Female	Senior Adult	yes	no	Yes	Private	Urban
5106	44873	Female	Senior Adult	no	no	Yes	Self-employed	Urban
5107	19723	Female	Senior Adult	no	no	Yes	Self-employed	Rural
5108	37544	Male	Senior Adult	no	no	Yes	Private	Rural
5109	44679	Female	Adult	no	no	Yes	Govt_job	Urban

5110 rows × 12 columns



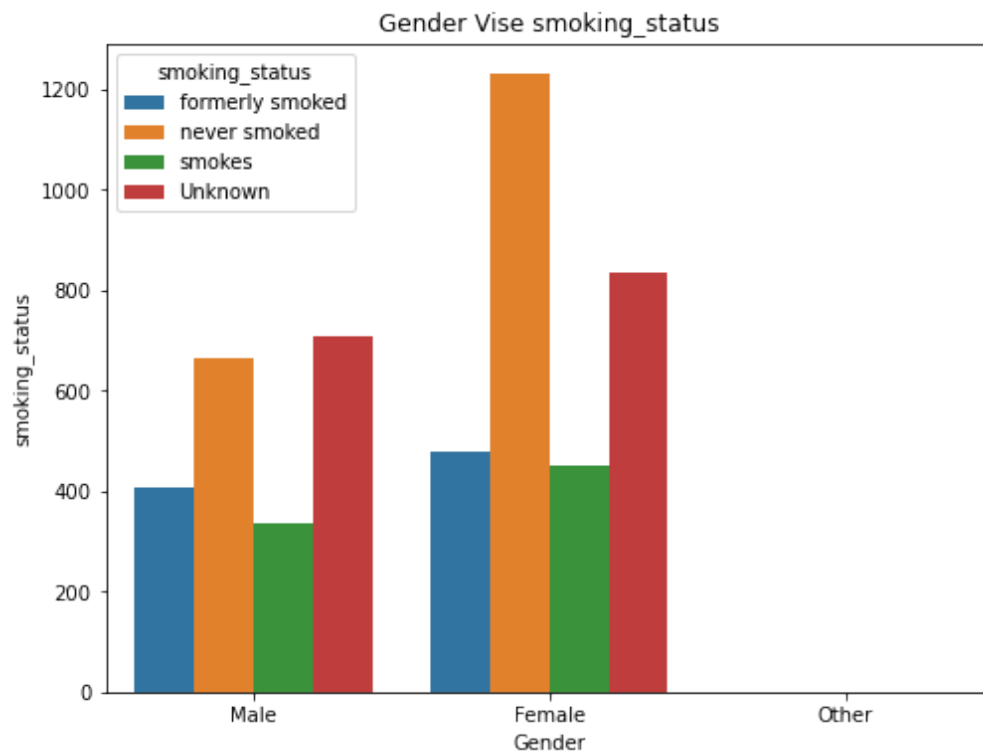
Gender Vise Count of Brain Stroke

```
In [38]: plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='gender', hue ="stroke")
plt.xlabel('Gender')
plt.ylabel('stroke')
plt.title('Gender Vise Brain stroke')
plt.show()
```



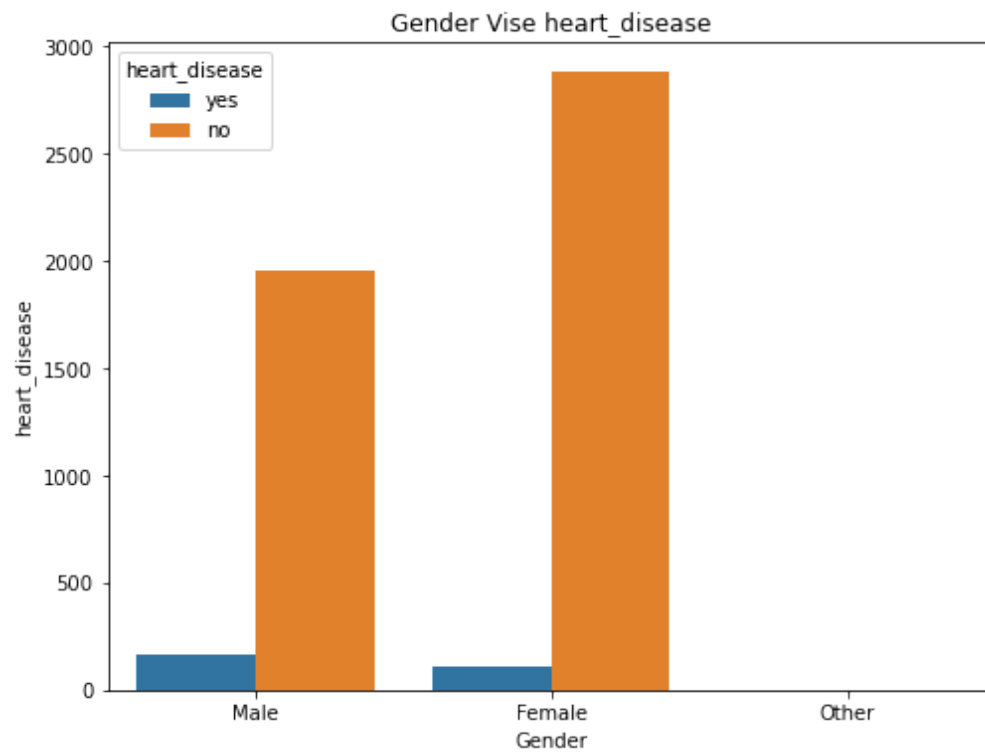
Gender Vise Count of Smoking Status

```
In [39]: plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='gender', hue ="smoking_status")
plt.xlabel('Gender')
plt.ylabel('smoking_status')
plt.title('Gender Vise smoking_status')
plt.show()
```



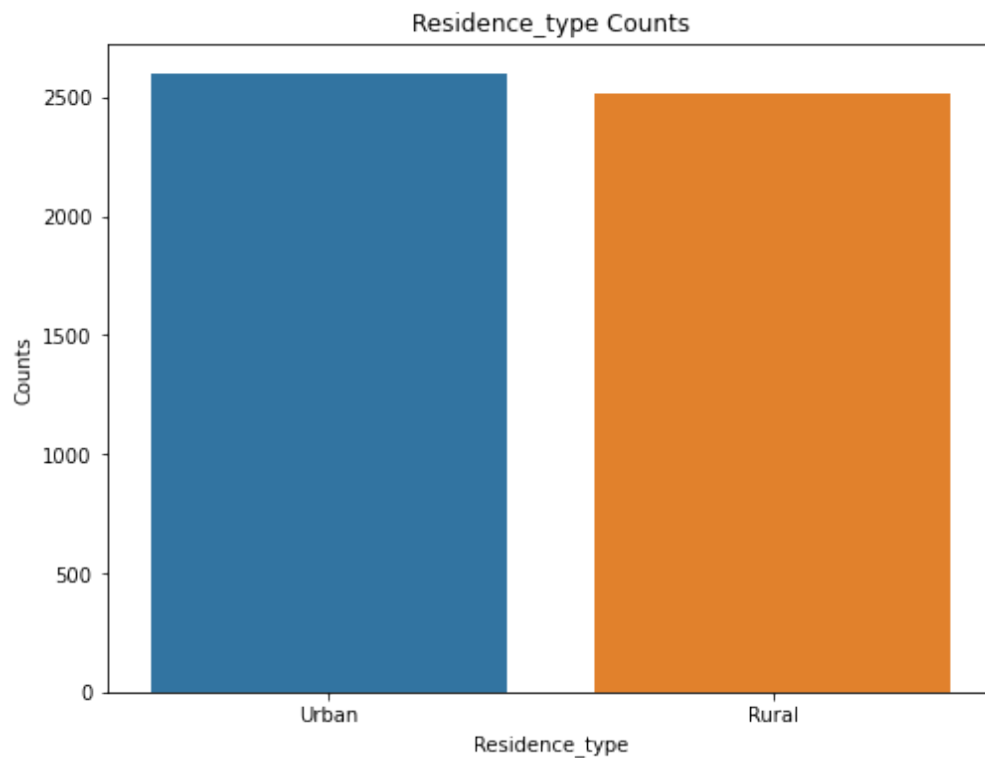
Gender Vise Count of Heart Disease

```
In [40]: plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='gender', hue ="heart_disease")
plt.xlabel('Gender')
plt.ylabel('heart_disease')
plt.title('Gender Vise heart_disease')
plt.show()
```



Count of Residency Type

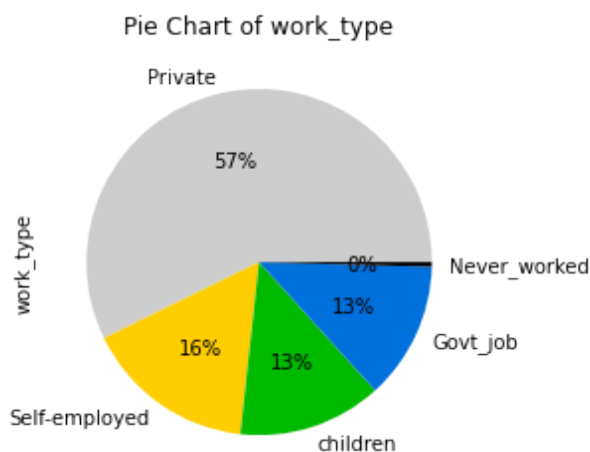
```
In [41]: plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Residence_type',)
plt.xlabel('Residence_type')
plt.ylabel('Counts')
plt.title('Residence_type Counts')
plt.show()
```



Pie chart for Work Type

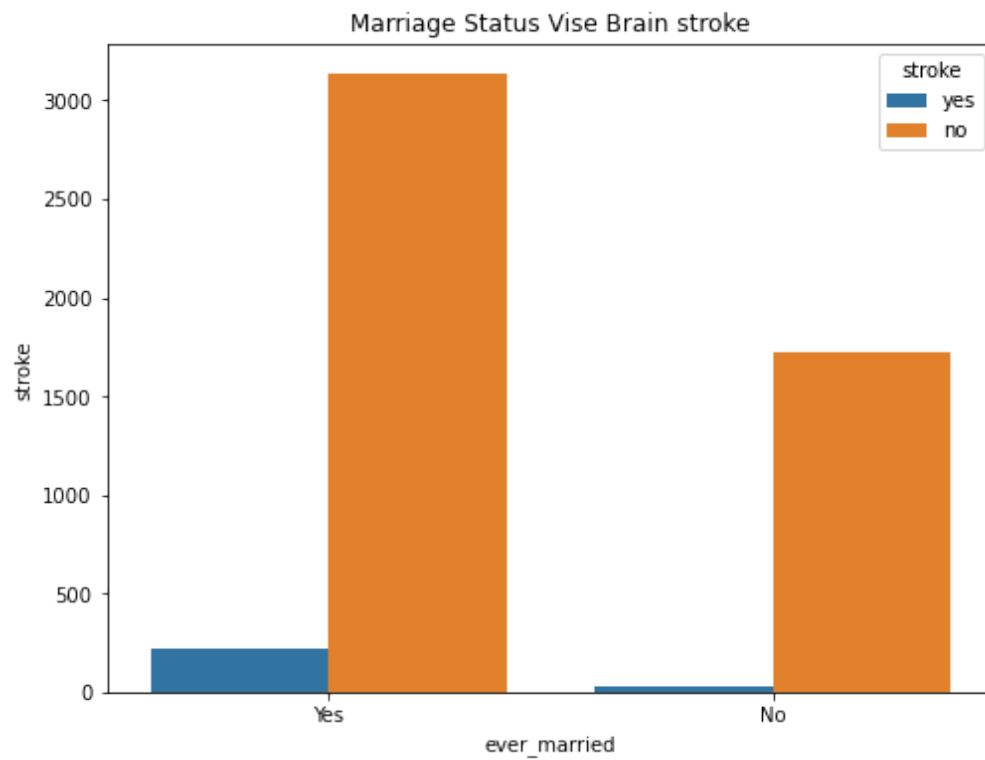
```
In [42]: df['work_type'].value_counts().plot(kind = 'pie' , title = 'Pie Chart of work_type',
autopct="%.0f%%", colormap='nipy_spectral_r')
```

```
Out[42]: <AxesSubplot:title={'center':'Pie Chart of work_type'}, ylabel='work_type'>
```



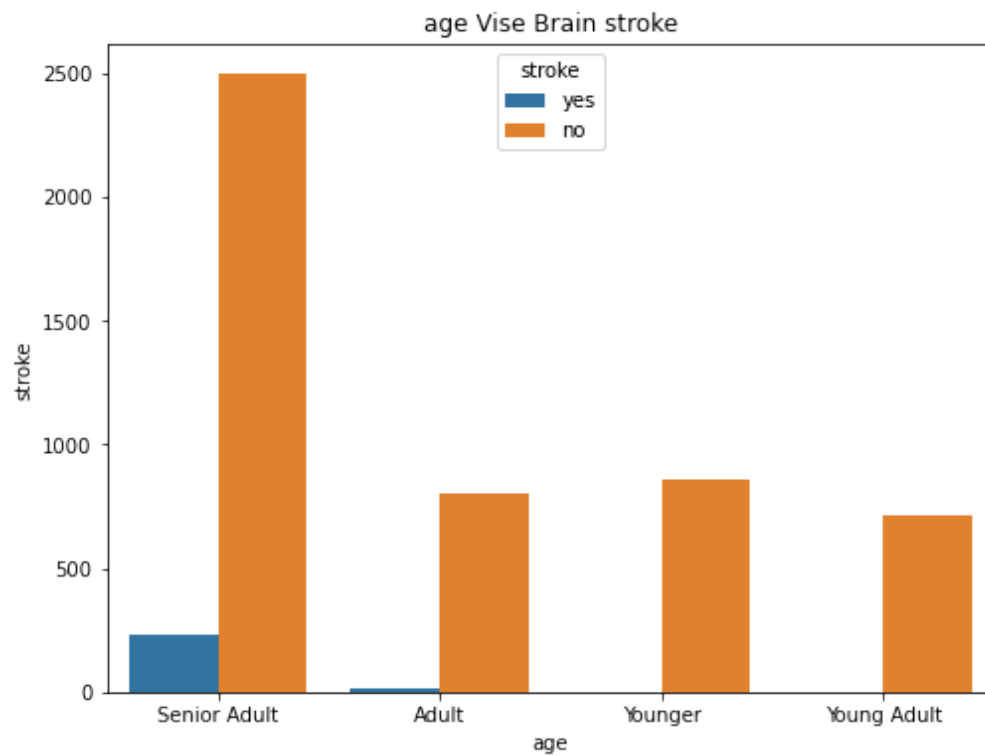
Count of Married Wise Brain Stroke

```
In [43]: plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='ever_married', hue ="stroke")
plt.xlabel('ever_married')
plt.ylabel('stroke')
plt.title('Marriage Status Vise Brain stroke')
plt.show()
```



Count of Age Wise Brain Stroke

```
In [44]: plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='age', hue ="stroke")
plt.xlabel('age')
plt.ylabel('stroke')
plt.title('age Vise Brain stroke')
plt.show()
```



Count of Age Wise Smoking Status



```
In [46]: plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='age', hue ="smoking_status")
plt.xlabel('age')
plt.ylabel('smoking_status')
plt.title('Age Vise Smoking Status')
plt.show()
```

