Data Visualization for Stroke

March 15, 2025

[ ]:
```
## Loading necessary libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[113]:
```
## Import the dataset
df = pd.read_csv("C:/Users/PC/OneDrive/Desktop/Data Science/Datasets/Datasets/
 ↪Stroke.csv")
```

[114]:
```
## Inspect the first 10 observations of the dataset
df.head(10)
```

[114]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
5	53882	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
6	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
7	60491	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
8	12109	Female	81.0	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
9	12095	Female	61.0	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1

```
0       1
1       1
2       1
3       1
4       1
5       1
6       1
7       1
8       1
9       1
```

[115]:
```
## Inspect the last 10 observations of the dataset
df.tail(10)
```

[115]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
4899	579	Male	9.0	0	0	No	children	Urban	71.88	17.5	Unknown	0
4900	68398	Male	82.0	1	0	Yes	Self-employed	Rural	71.97	28.3	never smoked	0
4901	36901	Female	45.0	0	0	Yes	Private	Urban	97.95	24.5	Unknown	0
4902	45010	Female	57.0	0	0	Yes	Private	Rural	77.93	21.7	never smoked	0
4903	22127	Female	18.0	0	0	No	Private	Urban	82.85	46.9	Unknown	0
4904	14180	Female	13.0	0	0	No	children	Rural	103.08	18.6	Unknown	0
4905	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
4906	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
4907	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
4908	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

```
4907       0
4908       0
```

[116]:
```
## Check the structure of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4909 entries, 0 to 4908
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 4909 non-null   int64
 1   gender             4909 non-null   object
 2   age                4909 non-null   float64
 3   hypertension       4909 non-null   int64
 4   heart_disease      4909 non-null   int64
 5   ever_married       4909 non-null   object
 6   work_type          4909 non-null   object
 7   Residence_type     4909 non-null   object
 8   avg_glucose_level  4909 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     4909 non-null   object
 11  stroke             4909 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 460.3+ KB
```

[117]:
```
## Check data types
df.dtypes
```

[117]:
```
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
```

[118]:
```
## Check for duplicates
df.duplicated().sum()
```

[118]:
```
0
```

[119]:
```
## Check for missing values
df.isnull().sum()
```

[119]:
```
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                  0
smoking_status       0
stroke               0
dtype: int64
```

[120]:
```
## Data preprocessing
df["hypertension"]  = df["hypertension"].astype("object")
df["heart_disease"]  = df["heart_disease"].astype("object")
df["stroke"]  = df["stroke"].astype("object")
```

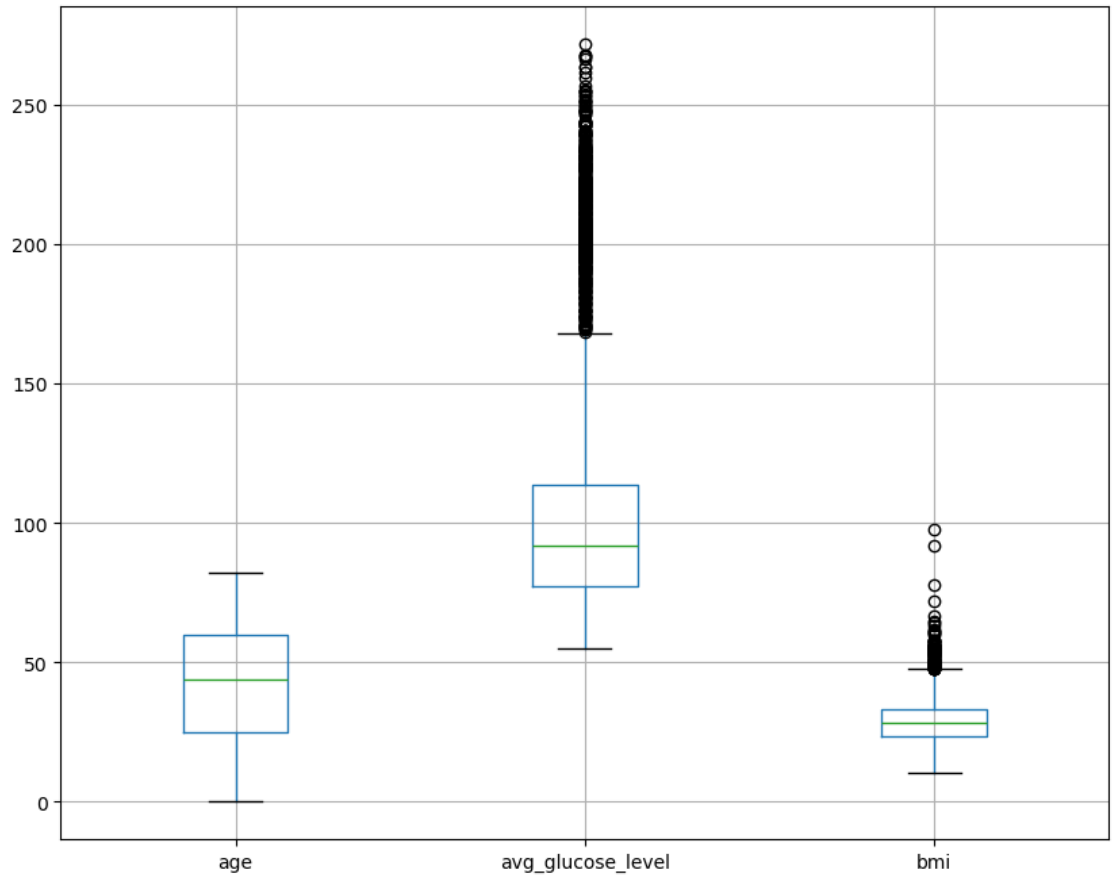
[121]:
```
## Drop Id
df = df.drop("id", axis = 1)
```

[122]:
```
## Summary statistics
df.describe()
```

[122]:

	age	avg_glucose_level	bmi
count	4909.000000	4909.000000	4909.000000
mean	42.865374	105.305150	28.893237
std	22.555115	44.424341	7.854067
min	0.080000	55.120000	10.300000
25%	25.000000	77.070000	23.500000
50%	44.000000	91.680000	28.100000
75%	60.000000	113.570000	33.100000
max	82.000000	271.740000	97.600000

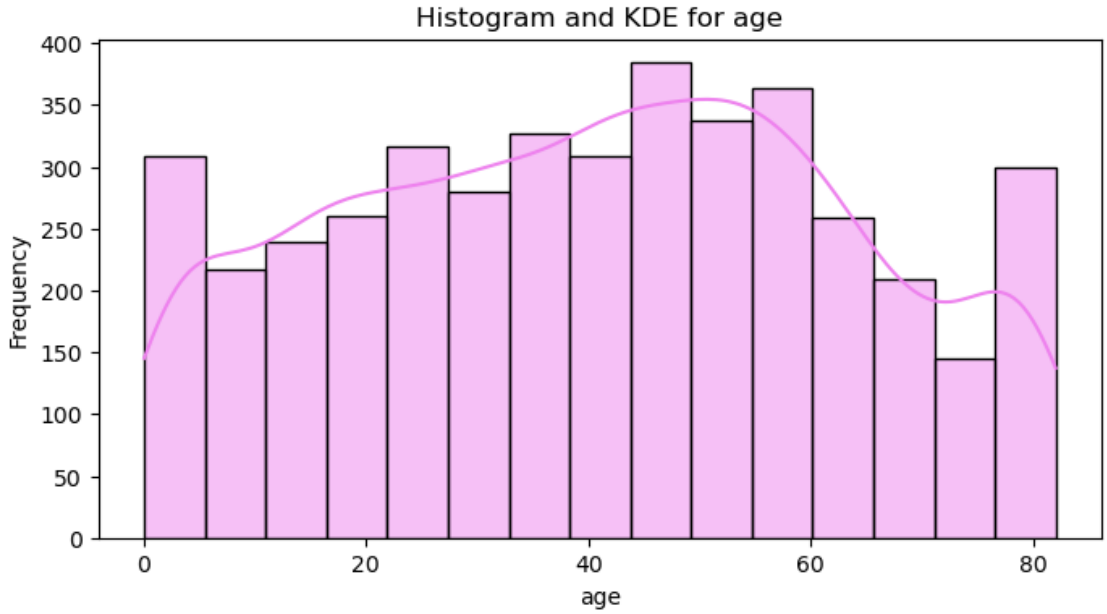
[123]:
```
## Checking for outliers
numeric_cols = df.select_dtypes(include = "float64")
numeric_cols.boxplot(figsize = (10, 8))
plt.show()
```

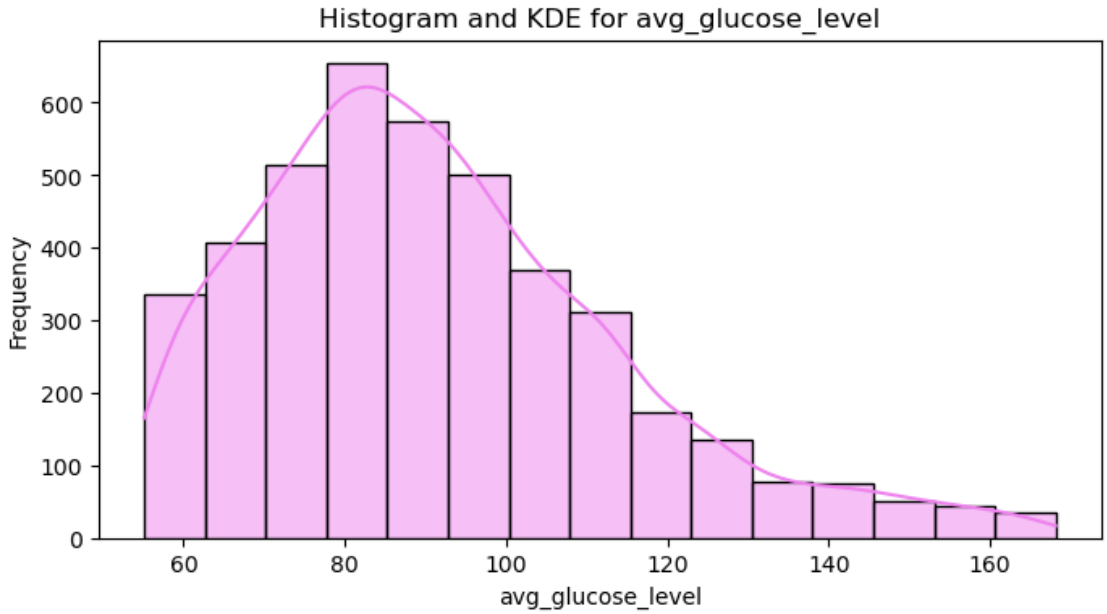


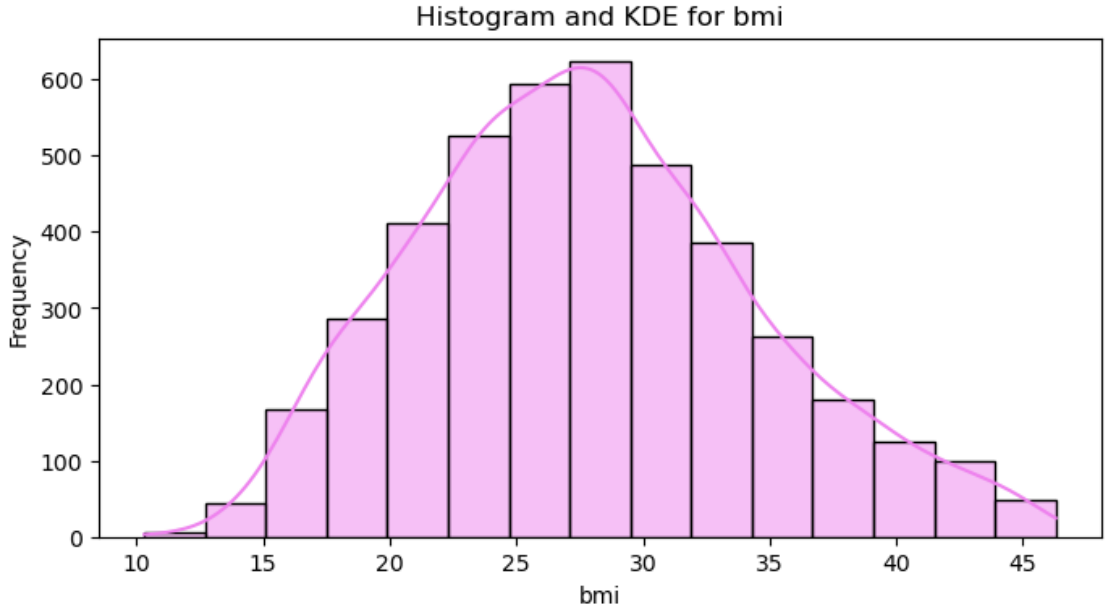
[124]:
```
## Remove outliers
## Average glucose level
Q1 = df["avg_glucose_level"].quantile(0.25)
Q3 = df["avg_glucose_level"].quantile(0.75)
IQR = Q3 - Q1

## Define the lower and upper bound
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

## Remove outliers
df = df[(df["avg_glucose_level"] >= lower_bound) & (df["avg_glucose_level"] <=␣
 ↪upper_bound)]

## bmi
Q1 = df["bmi"].quantile(0.25)
Q3 = df["bmi"].quantile(0.75)
IQR = Q3 - Q1
```

```
## Define the lower and upper bound
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

## Remove outliers
df = df[(df["bmi"] >= lower_bound) & (df["bmi"] <= upper_bound)]
```

[125]:
```
## Plotting histograms and kde plots for numeric columns
for col in numeric_cols:
    plt.figure(figsize = (8, 4))
    sns.histplot(df[col], kde = True, bins = 15, color = "violet")
    plt.title(f'Histogram and KDE for {col}')
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()
```

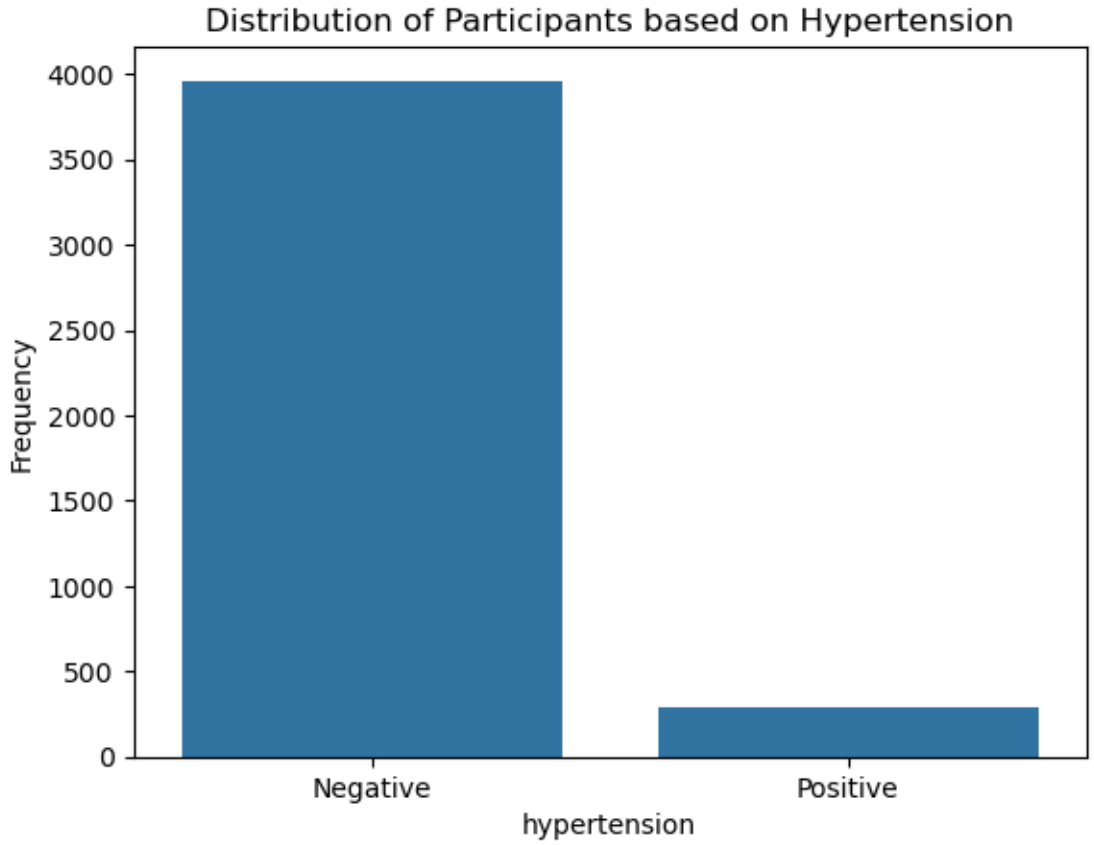




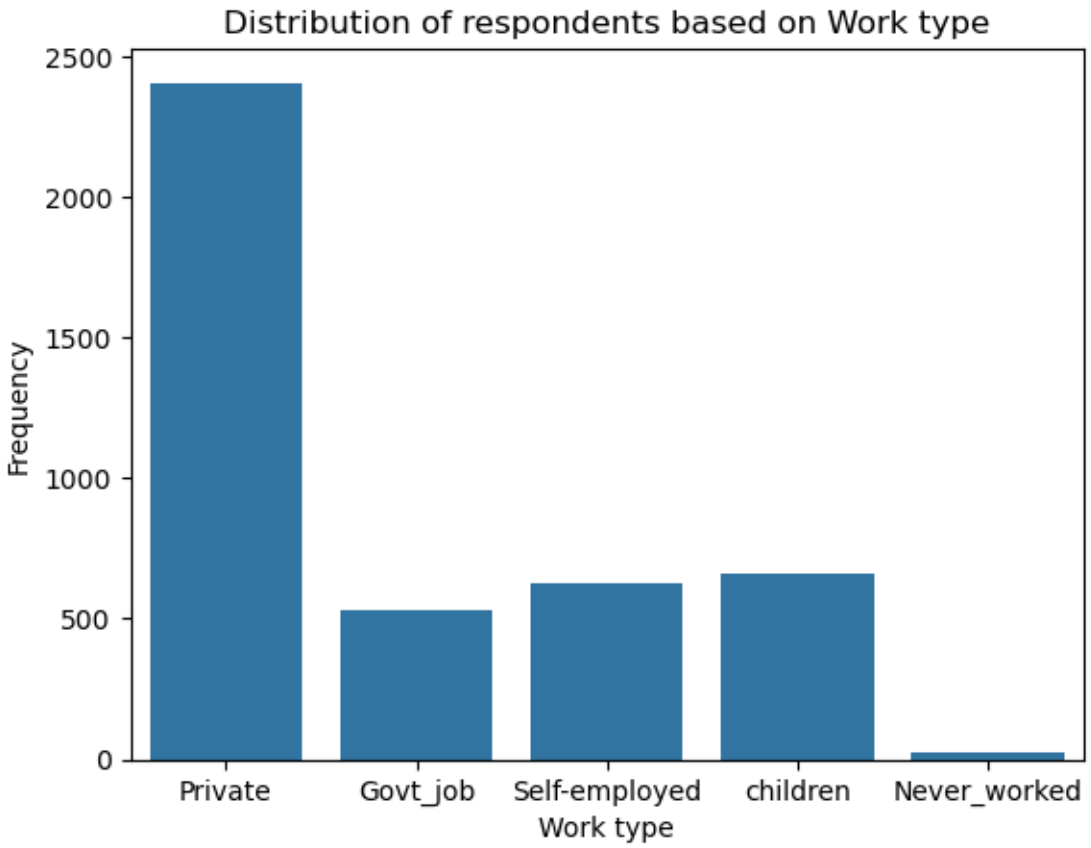


[126]:
```
## Bar charts for categorical variables
## Hypertension
sns.countplot(x = "hypertension", data = df)
plt.title('Distribution of Participants based on Hypertension')
plt.ylabel("Frequency")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
```

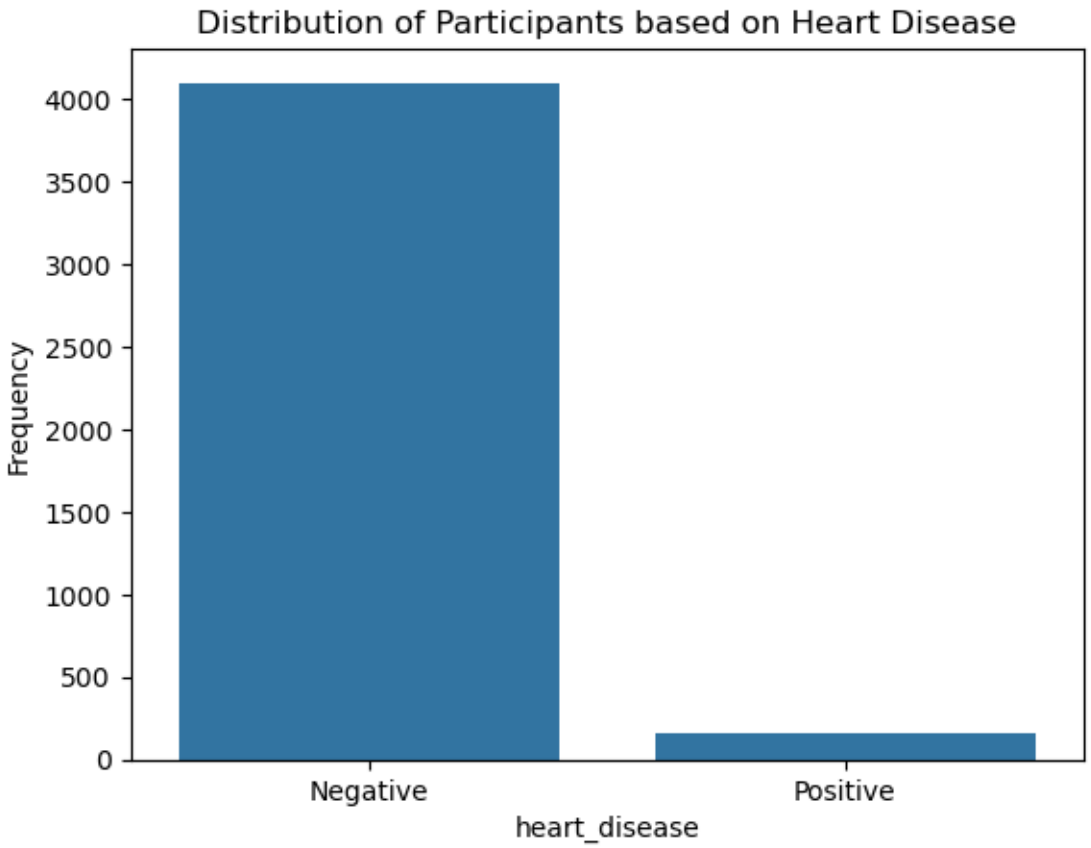
```
plt.show()
```



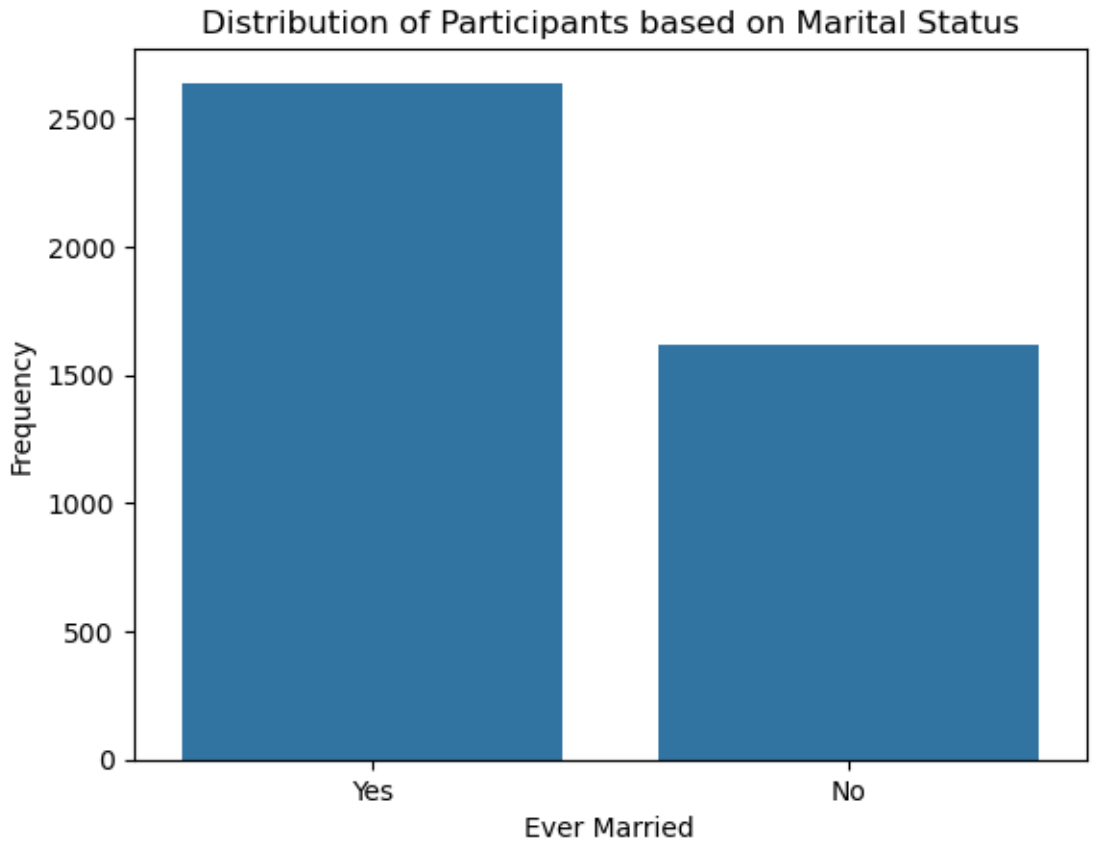
[127]:
```
## Work type
sns.countplot(x = "work_type", data = df)
plt.title('Distribution of respondents based on Work type')
plt.ylabel('Frequency')
plt.xlabel('Work type')
plt.show()
```



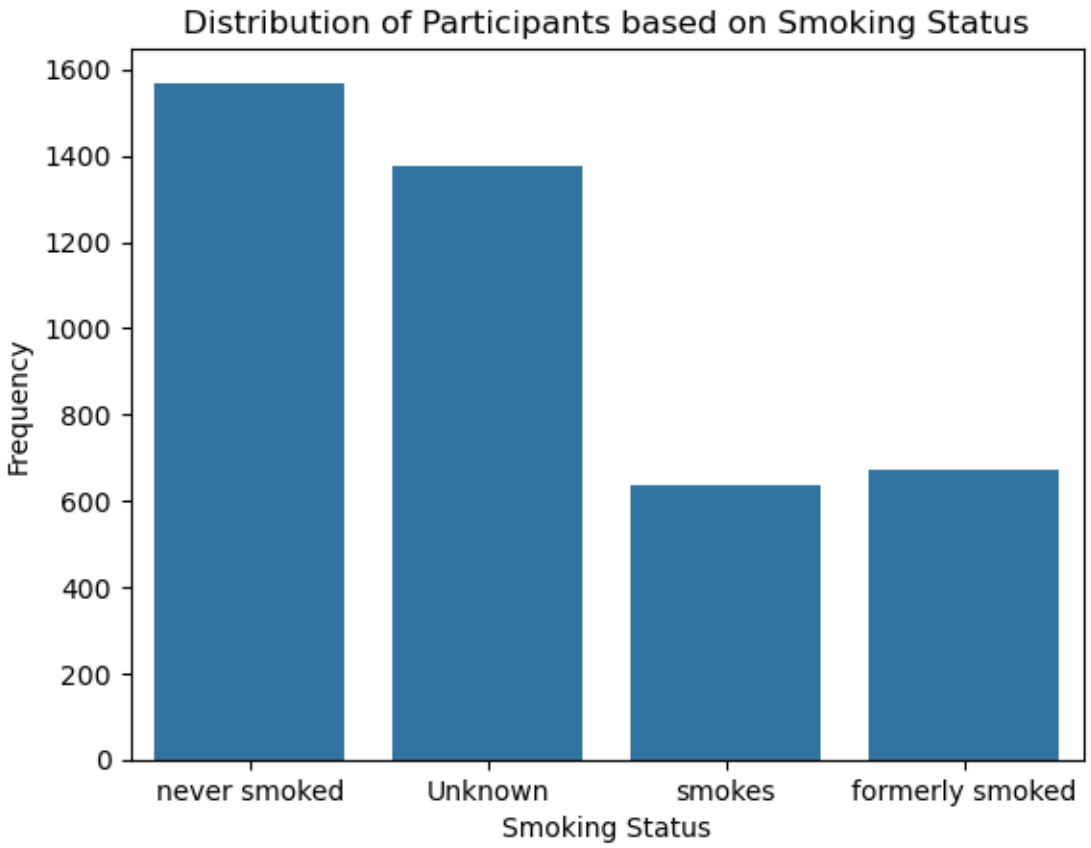
[128]:
```
## Heart Disease
sns.countplot(x = "heart_disease", data = df)
plt.title('Distribution of Participants based on Heart Disease')
plt.ylabel("Frequency")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



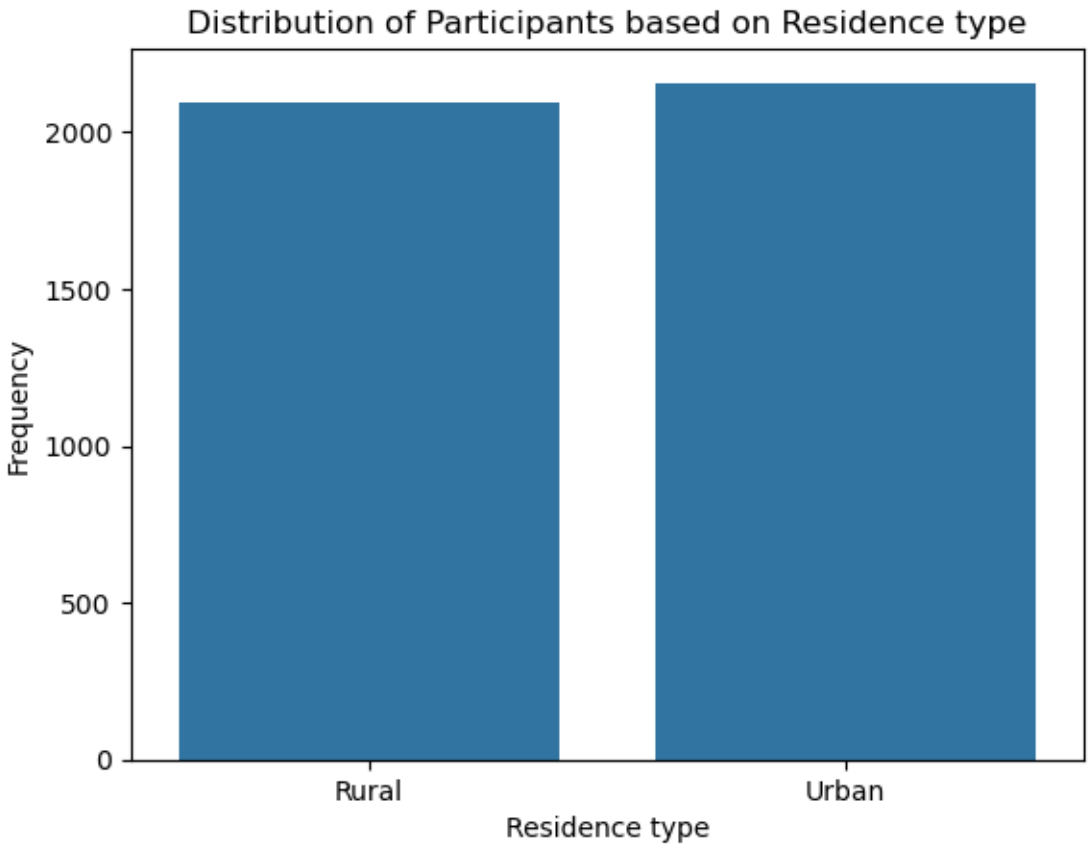
[129]:
```
## Marital Status
sns.countplot(x = "ever_married", data = df)
plt.title('Distribution of Participants based on Marital Status')
plt.ylabel("Frequency")
plt.xlabel("Ever Married")
plt.show()
```



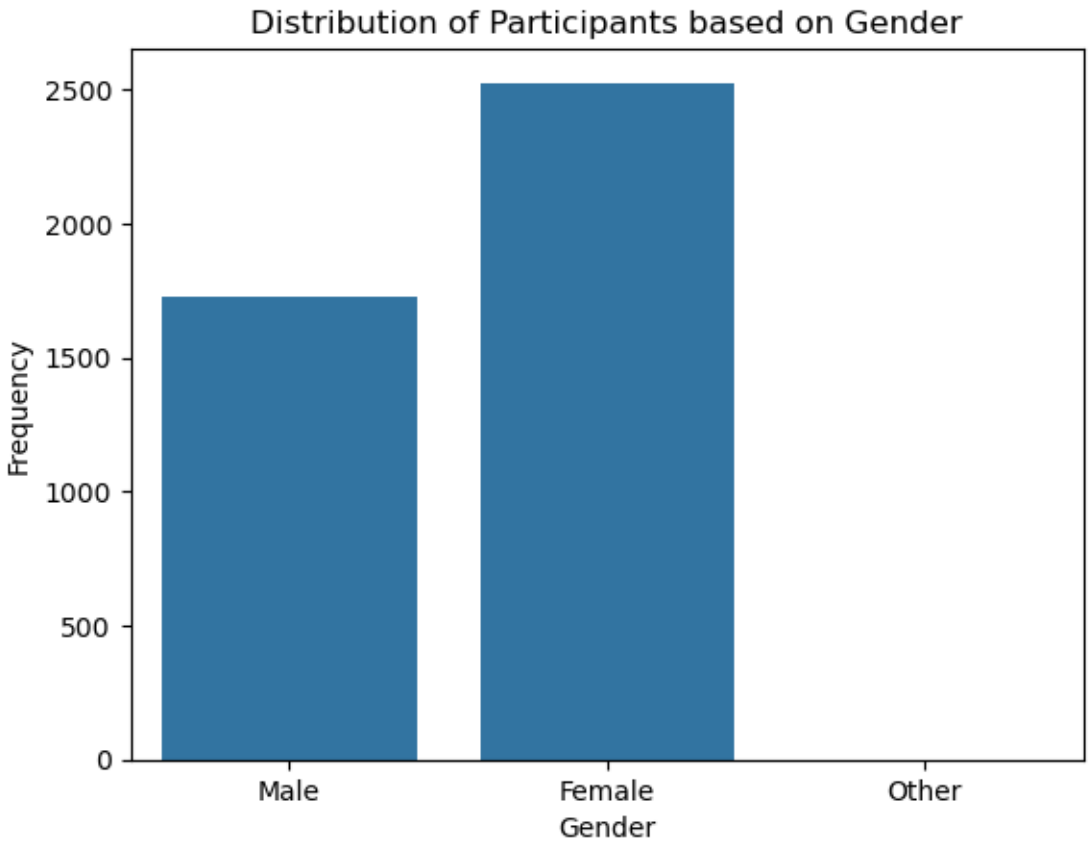
[130]:
```
## Smoking Status
sns.countplot(x = "smoking_status", data = df)
plt.title('Distribution of Participants based on Smoking Status')
plt.ylabel("Frequency")
plt.xlabel("Smoking Status")
plt.show()
```



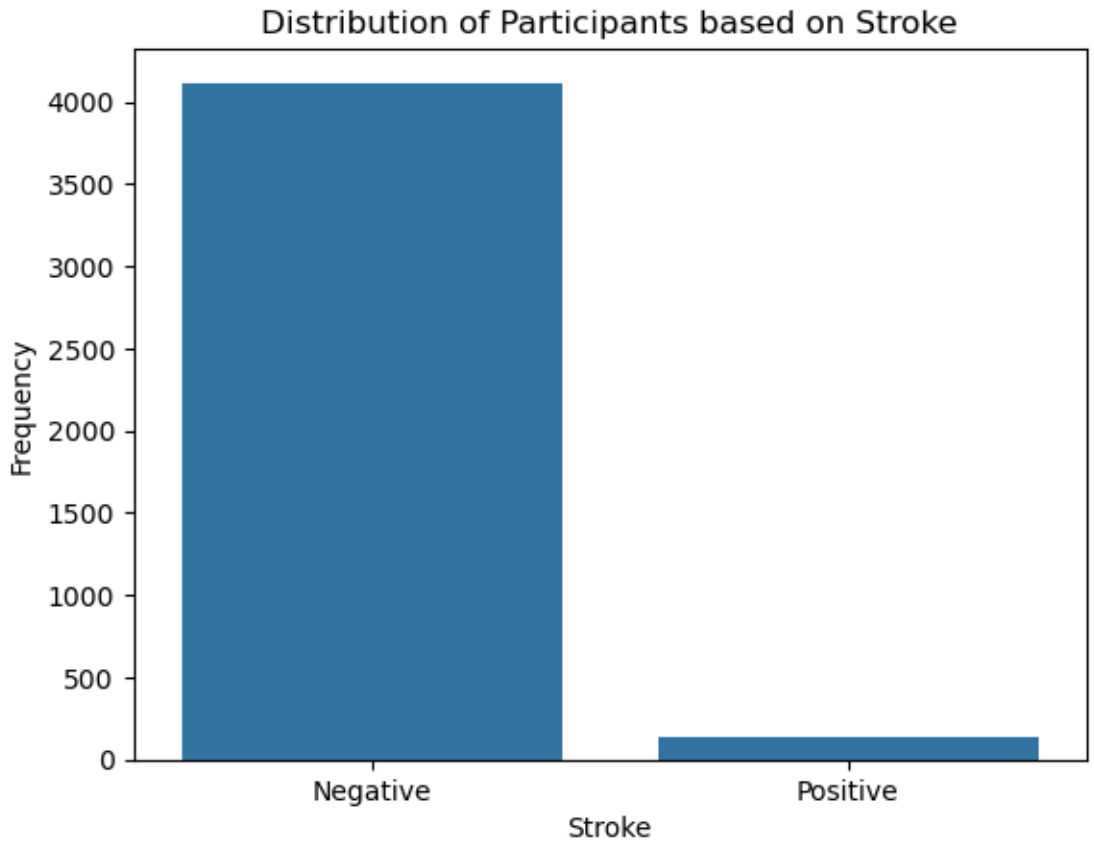
[131]:
```
## Residence type
sns.countplot(x = "Residence_type", data = df)
plt.title('Distribution of Participants based on Residence type')
plt.ylabel("Frequency")
plt.xlabel("Residence type")
plt.show()
```



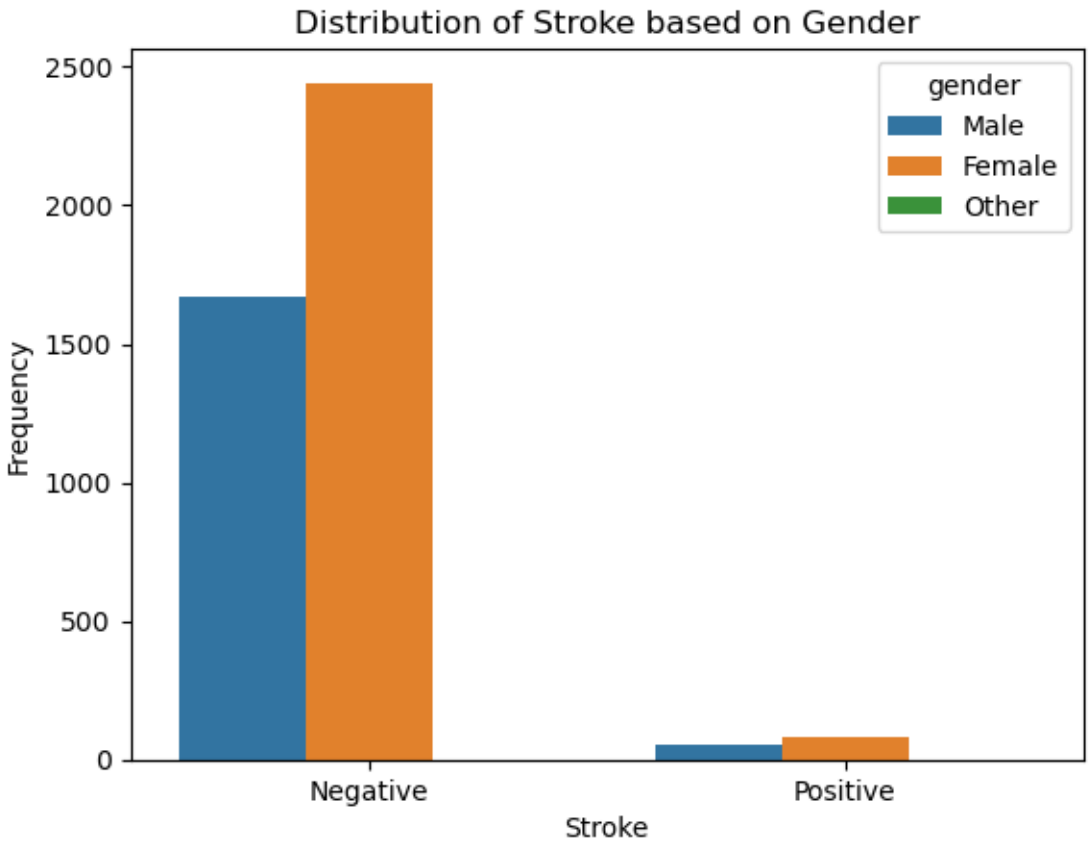
[132]:
```
## Gender
sns.countplot(x = "gender", data = df)
plt.title('Distribution of Participants based on Gender')
plt.ylabel("Frequency")
plt.xlabel("Gender")
plt.show()
```



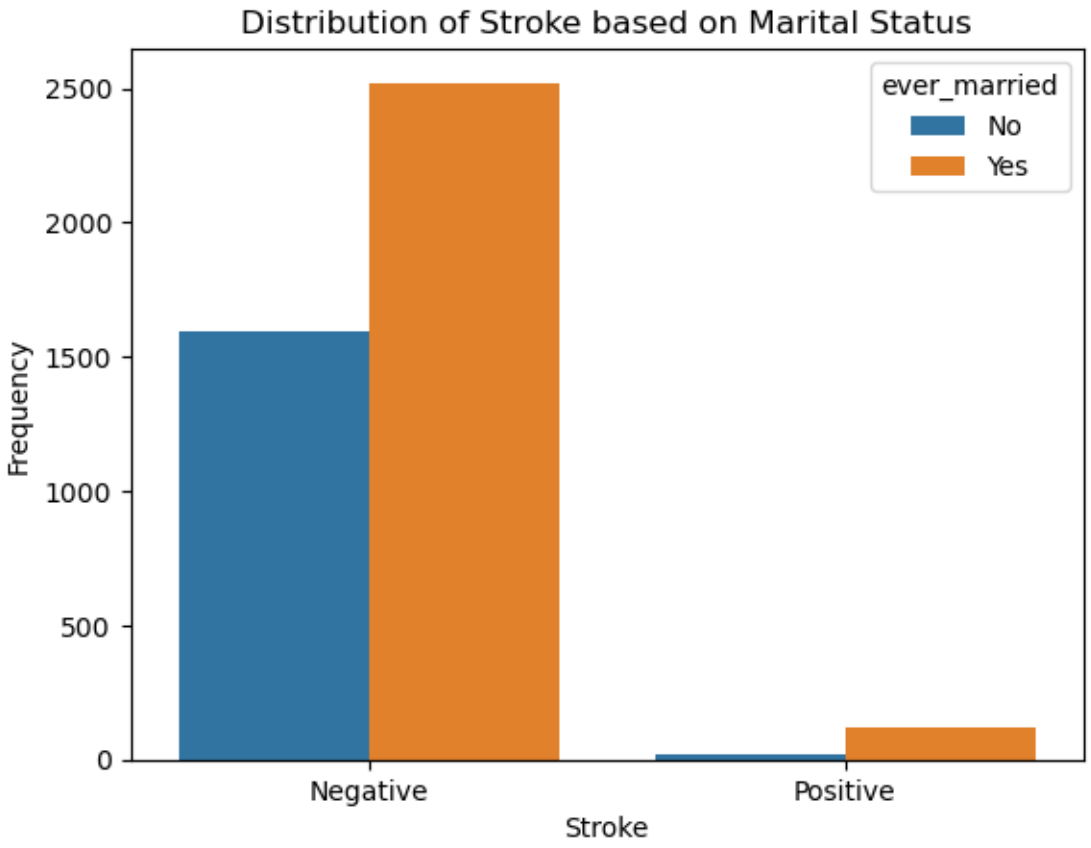
[133]:
```
## Stroke
sns.countplot(x = "stroke", data = df)
plt.title('Distribution of Participants based on Stroke')
plt.ylabel("Frequency")
plt.xlabel("Stroke")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



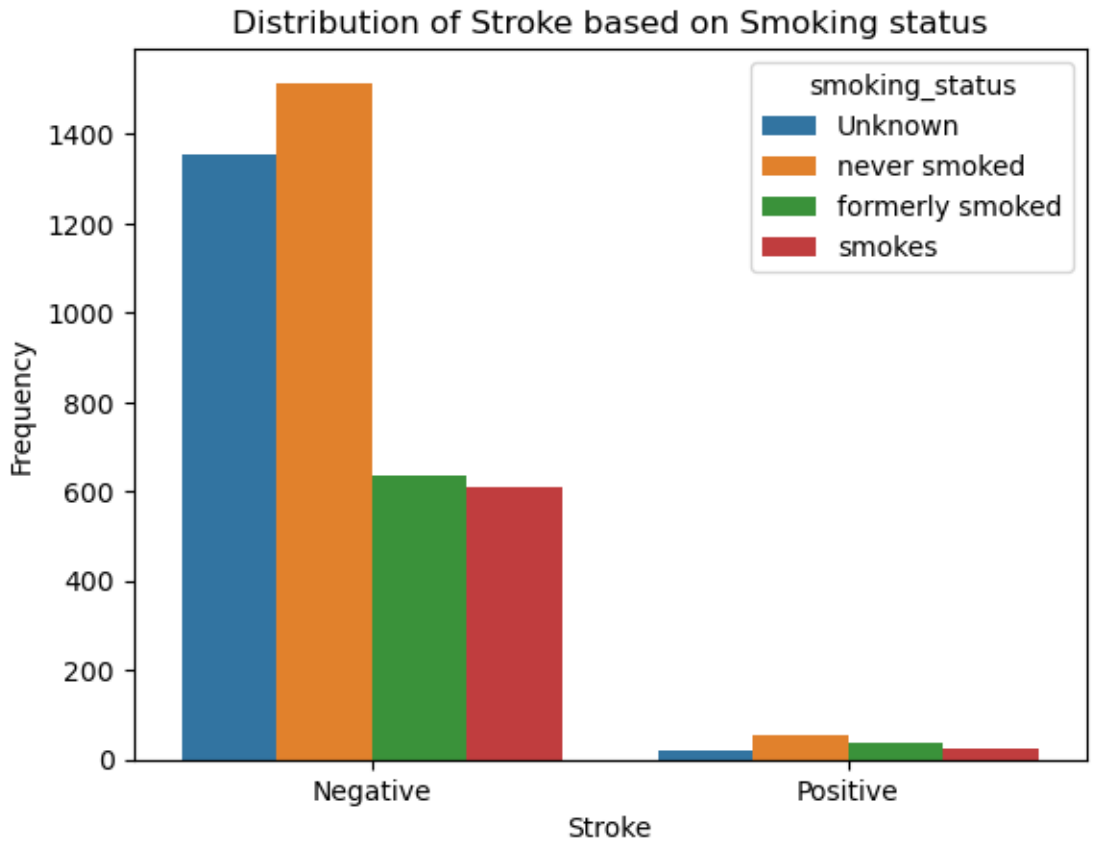
[134]:
```
## Distribution of stroke based on Gender
sns.countplot(x = "stroke", data = df, hue = "gender")
plt.ylabel("Frequency")
plt.xlabel("Stroke")
plt.title("Distribution of Stroke based on Gender")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



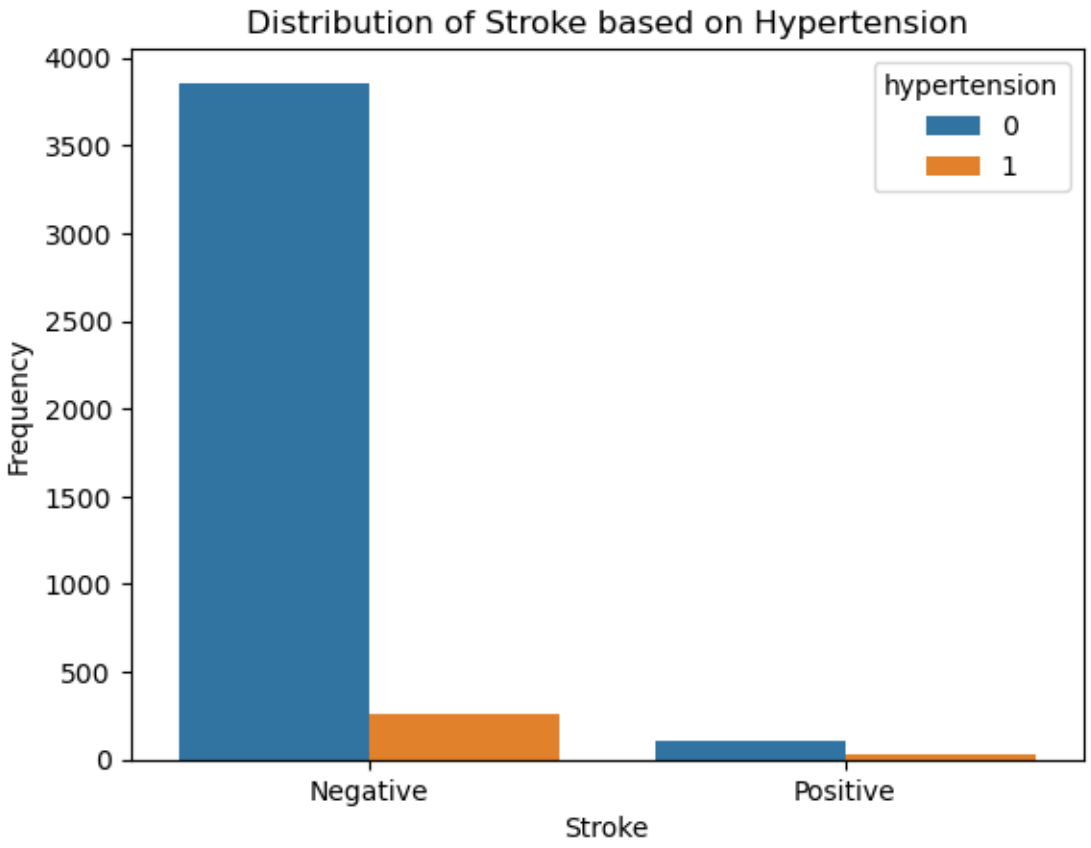
[135]:
```
## Distribution of stroke based on Marital Status
sns.countplot(x = "stroke", data = df, hue = "ever_married")
plt.ylabel("Frequency")
plt.xlabel("Stroke")
plt.title("Distribution of Stroke based on Marital Status")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



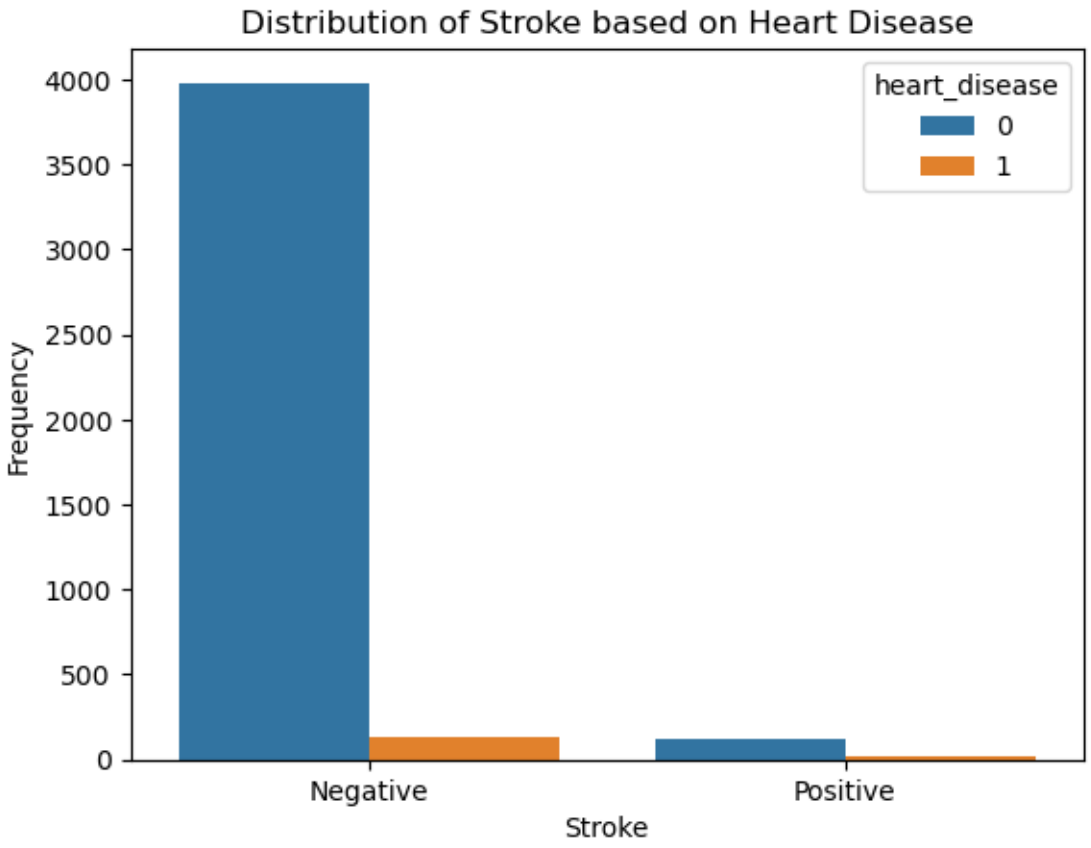
[136]:
```
## Distribution of stroke based on Smoking Status
sns.countplot(x = "stroke", data = df, hue = "smoking_status")
plt.ylabel("Frequency")
plt.xlabel("Stroke")
plt.title("Distribution of Stroke based on Smoking status")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



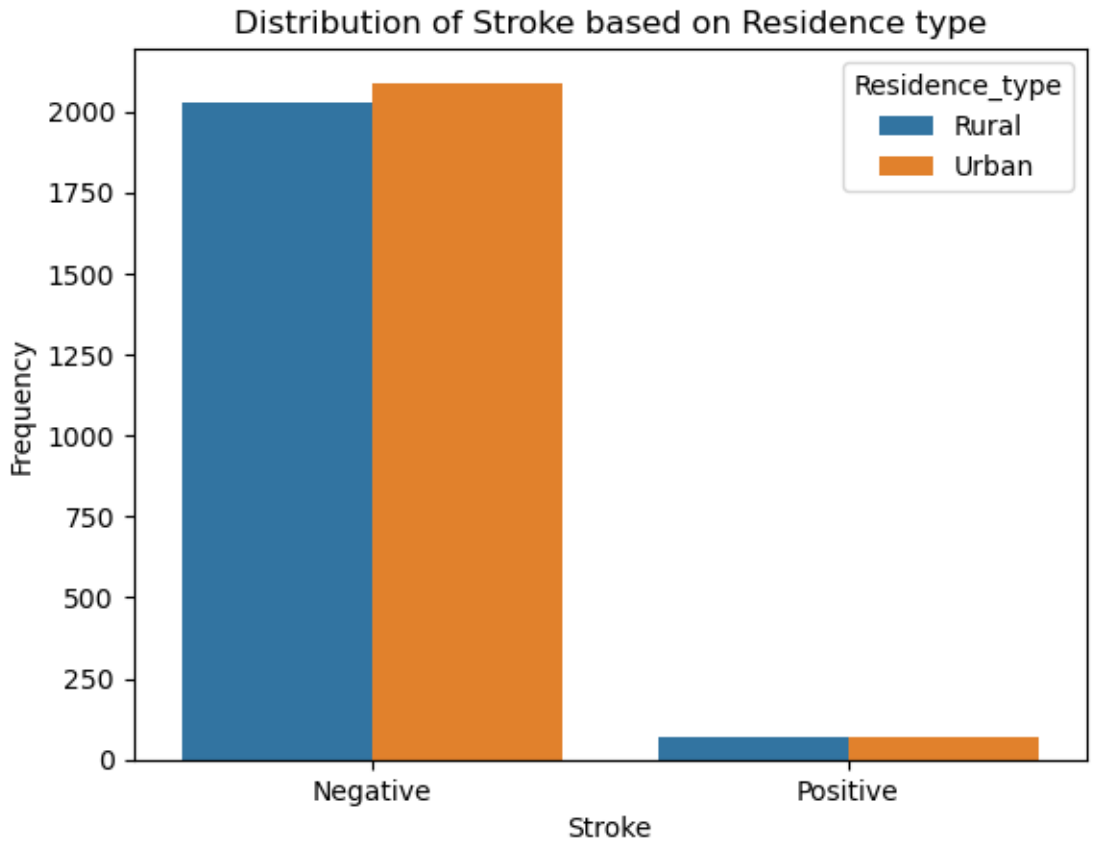
[137]:
```
## Distribution of stroke based on Hypertension
sns.countplot(x = "stroke", data = df, hue = "hypertension")
plt.ylabel("Frequency")
plt.xlabel("Stroke")
plt.title("Distribution of Stroke based on Hypertension")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



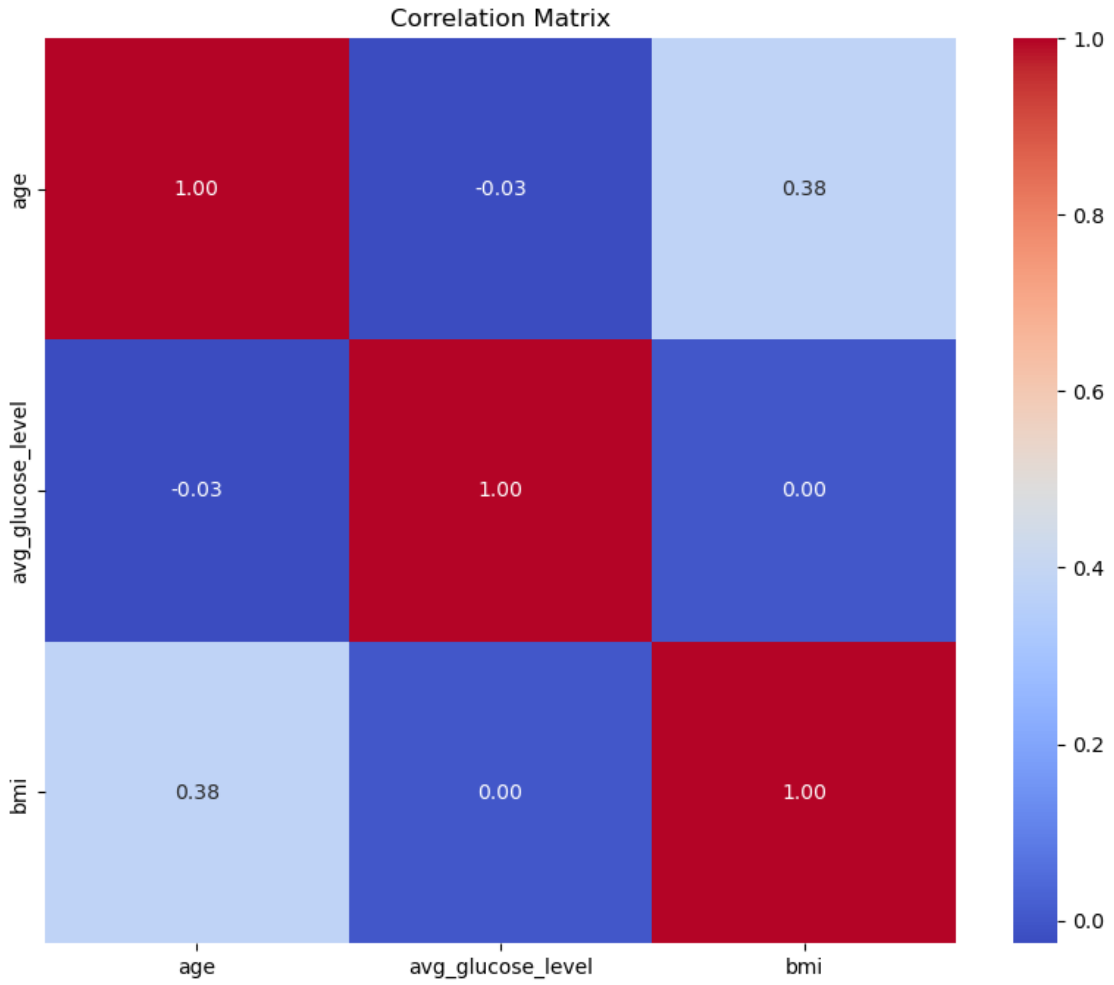
[138]:
```
## Distribution of stroke based on Heart Disease
sns.countplot(x = "stroke", data = df, hue = "heart_disease")
plt.ylabel("Frequency")
plt.xlabel("Stroke")
plt.title("Distribution of Stroke based on Heart Disease")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



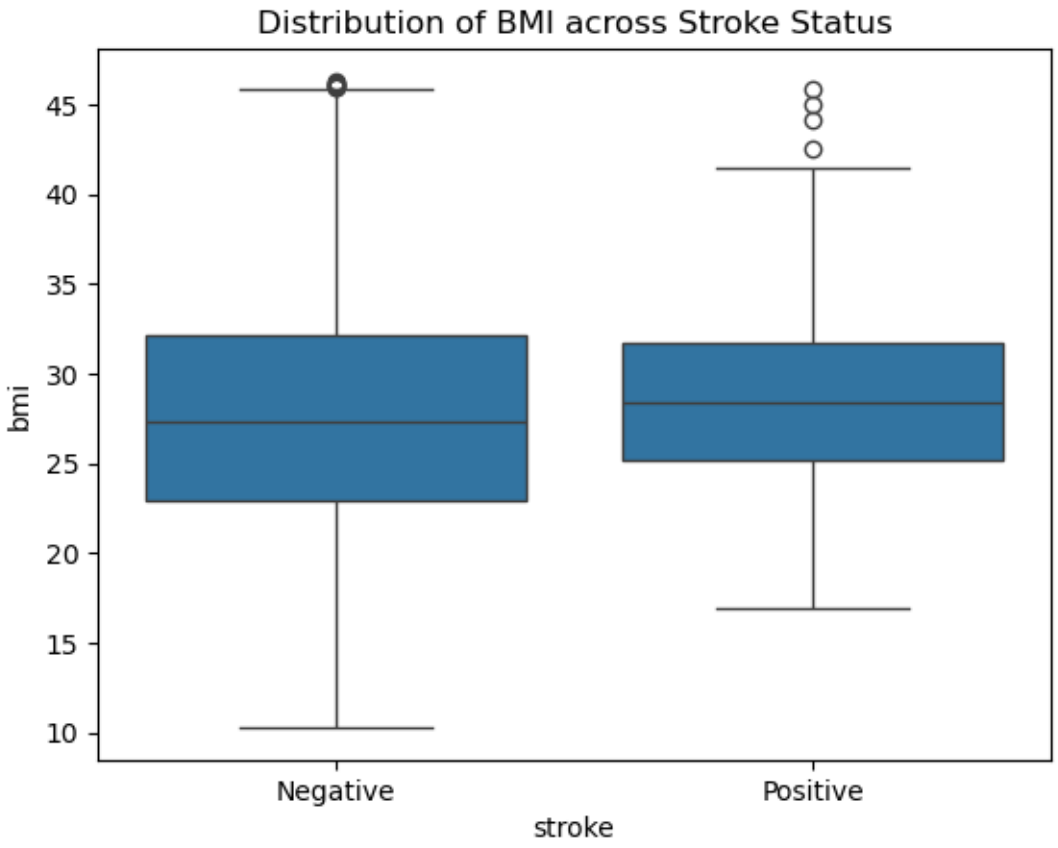
[139]:
```
## Distribution of stroke based on Residence type
sns.countplot(x = "stroke", data = df, hue = "Residence_type")
plt.ylabel("Frequency")
plt.xlabel("Stroke")
plt.title("Distribution of Stroke based on Residence type")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



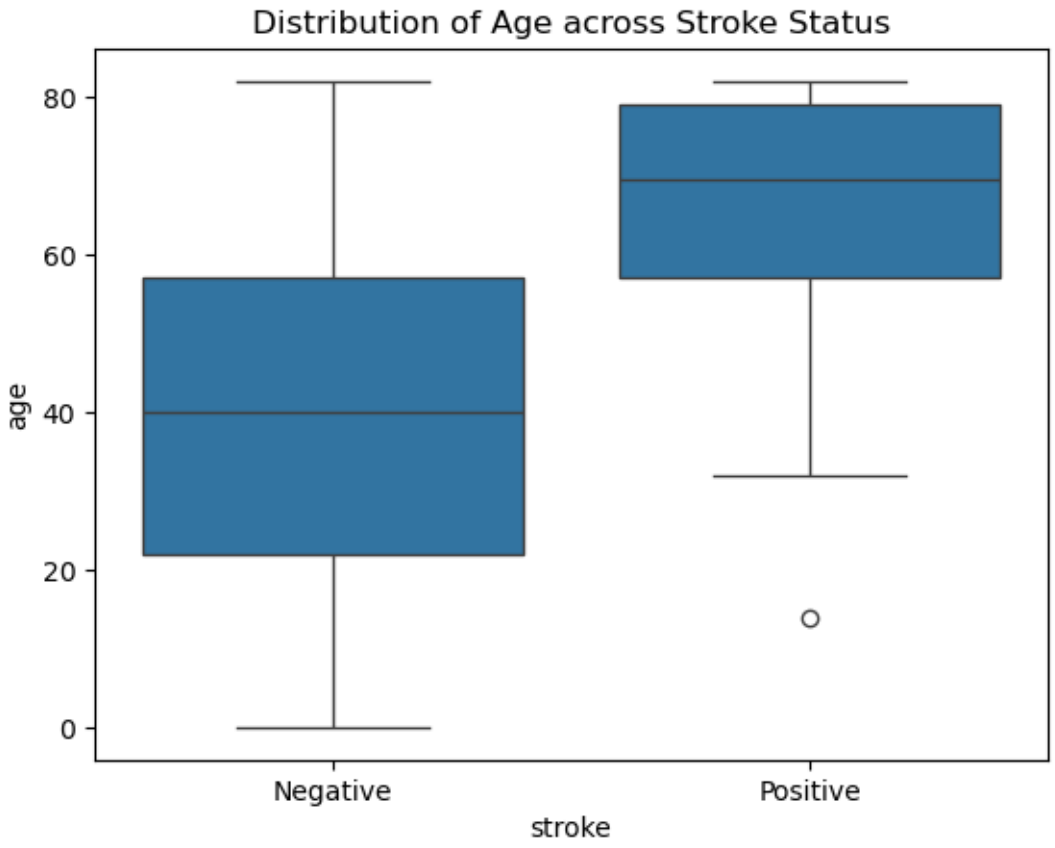
[140]:
```
## Relationship between numeric variables
## Select numeric columns
numeric_cols = df.select_dtypes(include = "float64")
## Compute Correlation Matrix
correlation_matrix = numeric_cols.corr()
## Visualize the correlation Matrix
plt.figure(figsize = (10, 8))
sns.heatmap(correlation_matrix, annot = True, cmap = "coolwarm", fmt = ".2f")
plt.title("Correlation Matrix")
plt.show()
```



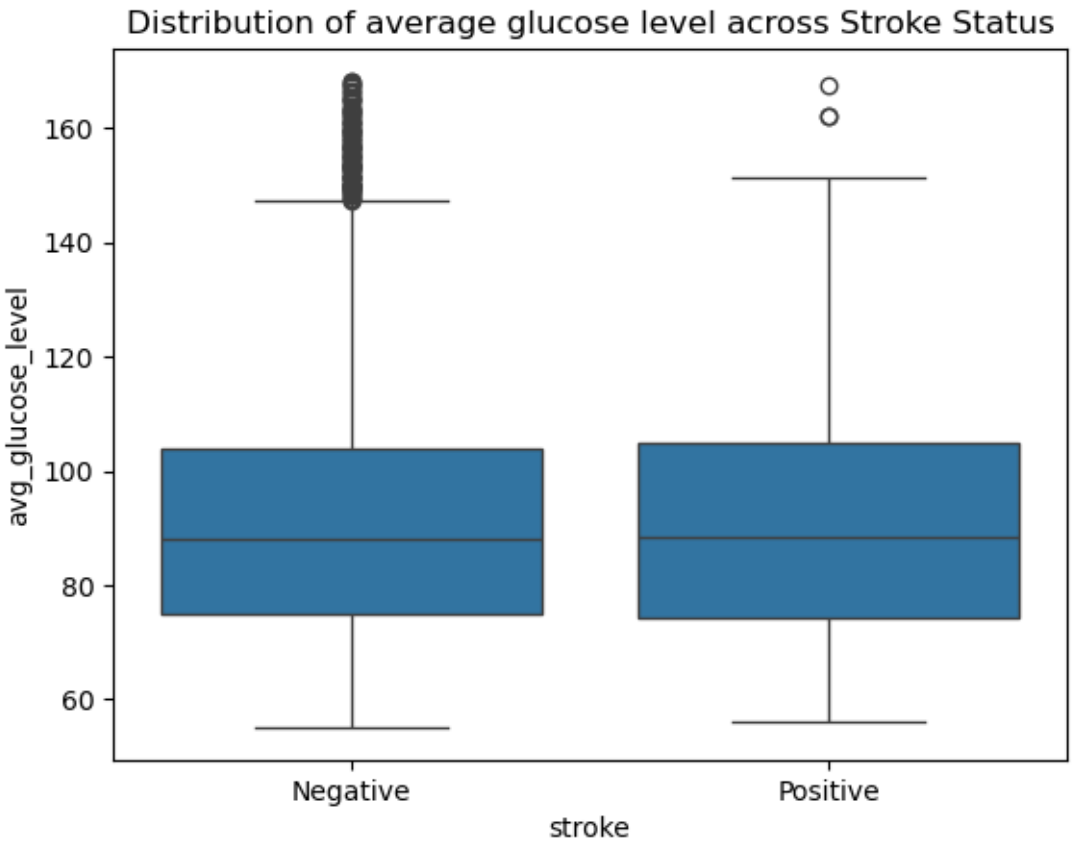
[141]:
```
## Box plots
## BMI vs Stroke
sns.boxplot(x = "stroke", y = "bmi", data = df)
plt.title("Distribution of BMI across Stroke Status")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



[142]:
```
## Age vs Stroke
sns.boxplot(x = "stroke", y = "age", data = df)
plt.title("Distribution of Age across Stroke Status")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



[143]:
```
## Glucose level vs Stroke
sns.boxplot(x = "stroke", y = "avg_glucose_level", data = df)
plt.title("Distribution of average glucose level across Stroke Status")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```


Distribution of average glucose level across Stroke Status
avg_glucose_level
160
140
120
100
80
60
Negative
Positive
stroke