21BDS0379

SHREYASHA SHRESTHA

EDA THEORY- D1

Digital Assignment 1

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


# Load dataset
df = pd.read_csv('MyDataset_EDA.csv')


# Display basic info
df.info()
print("\nSummary Statistics:")
print(df.describe(include='all'))
```

```
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB

Summary Statistics:
       school  sex         age address famsize Pstatus        Medu  \
count     395  393  395.000000     394     394     395  395.000000
unique      2    2         NaN       3       2       3         NaN
top        GP    F         NaN       U     GT3       T         NaN
freq      349  208         NaN     304     280     348         NaN
mean      NaN  NaN   16.696203     NaN     NaN     NaN    2.749367
std       NaN  NaN    1.276043     NaN     NaN     NaN    1.094735
min       NaN  NaN   15.000000     NaN     NaN     NaN    0.000000
25%       NaN  NaN   16.000000     NaN     NaN     NaN    2.000000
50%       NaN  NaN   17.000000     NaN     NaN     NaN    3.000000
75%       NaN  NaN   18.000000     NaN     NaN     NaN    4.000000
max       NaN  NaN   22.000000     NaN     NaN     NaN    4.000000

              Fedu   Mjob   Fjob  ...      famrel    freetime       goout  \
count   395.000000    395    395  ...  395.000000  395.000000  395.000000
unique         NaN      5      5  ...         NaN         NaN         NaN
top            NaN  other  other  ...         NaN         NaN         NaN
freq           NaN    141    217  ...         NaN         NaN         NaN
mean      2.521519    NaN    NaN  ...    3.944304    3.235443    3.108861
std       1.088201    NaN    NaN  ...    0.896659    0.998862    1.113278
min       0.000000    NaN    NaN  ...    1.000000    1.000000    1.000000
25%       2.000000    NaN    NaN  ...    4.000000    3.000000    2.000000
50%       2.000000    NaN    NaN  ...    4.000000    3.000000    3.000000
75%       3.000000    NaN    NaN  ...    5.000000    4.000000    4.000000
max       4.000000    NaN    NaN  ...    5.000000    5.000000    5.000000

              Dalc        Walc      health    absences          G1  \
count   395.000000  395.000000  395.000000  395.000000  395.000000
unique         NaN         NaN         NaN         NaN         NaN
top            NaN         NaN         NaN         NaN         NaN
freq           NaN         NaN         NaN         NaN         NaN
mean      1.481013    2.291139    3.554430    5.708861   10.908861
std       0.890741    1.287897    1.390303    8.003096    3.319195
min       1.000000    1.000000    1.000000    0.000000    3.000000
25%       1.000000    1.000000    3.000000    0.000000    8.000000
50%       1.000000    2.000000    4.000000    4.000000   11.000000
75%       2.000000    3.000000    5.000000    8.000000   13.000000
max       5.000000    5.000000    5.000000   75.000000   19.000000

                G2          G3
count   395.000000  395.000000
unique         NaN         NaN
top            NaN         NaN
freq           NaN         NaN
mean     10.713924   10.415190
std       3.761505    4.581443
min       0.000000    0.000000
25%       9.000000    8.000000
50%      11.000000   11.000000
75%      13.000000   14.000000
max      19.000000   20.000000

[11 rows x 33 columns]
```

```
# Handling missing values
df['sex'].fillna(df['sex'].mode()[0], inplace=True)
df['address'].fillna(df['address'].mode()[0], inplace=True)
df['famsize'].fillna(df['famsize'].mode()[0], inplace=True)
df['schoolsup'].fillna(df['schoolsup'].mode()[0], inplace=True)
```

```
<ipython-input-19-f698c8d2acdd>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c


  df['sex'].fillna(df['sex'].mode()[0], inplace=True)
<ipython-input-19-f698c8d2acdd>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c


  df['address'].fillna(df['address'].mode()[0], inplace=True)
<ipython-input-19-f698c8d2acdd>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c


  df['famsize'].fillna(df['famsize'].mode()[0], inplace=True)
<ipython-input-19-f698c8d2acdd>:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c


  df['schoolsup'].fillna(df['schoolsup'].mode()[0], inplace=True)
```

```
# Remove duplicate rows
df = df.drop_duplicates()
```

```
# Convert data types if needed
def convert_dtype(df):
    for col in df.columns:
        if df[col].dtype == 'object':
            try:
                df[col] = pd.to_datetime(df[col])  # Convert date-like columns
            except:
                pass  # If conversion fails, leave as object
        elif df[col].dtype in ['int64', 'float64']:
            df[col] = pd.to_numeric(df[col], errors='coerce')  # Convert numeric values
    return df

df = convert_dtype(df)
```

```
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
<ipython-input-21-369a7d50bdcc>:6: UserWarning: Could not infer format, so each element will be parsed individually, fal
  df[col] = pd.to_datetime(df[col])  # Convert date-like columns
```

```
# Handling outliers (Removing values beyond 1.5*IQR)
def remove_outliers(df):
    for col in ['absences']:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

df = remove_outliers(df)


# Encoding categorical variables
for col in df.select_dtypes(include=['object']).columns:
    df[col] = df[col].astype('category').cat.codes
```

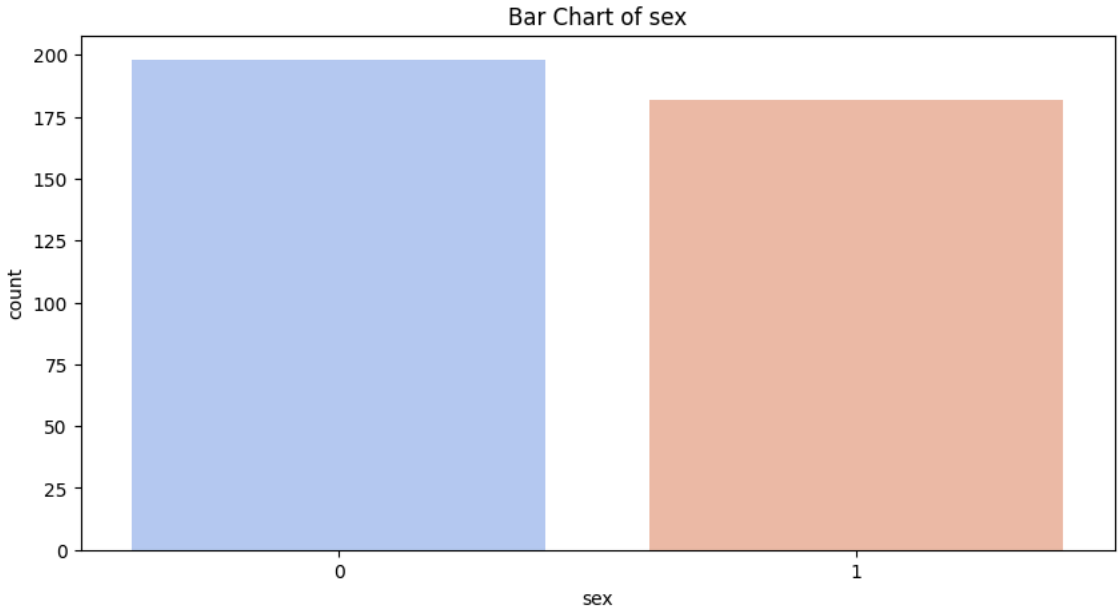
Univariate Analysis

```
# Central Tendency Measures
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    print(f"\n{col}:")
    print(f"Mean: {df[col].mean()}")
    print(f"Median: {df[col].median()}")
    print(f"Mode: {df[col].mode()[0]}")

# Dispersion Measures
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    print(f"\n{col}:")
    print(f"Range: {df[col].max() - df[col].min()}")
    print(f"Variance: {df[col].var()}")
    print(f"Standard Deviation: {df[col].std()}")
    print(f"Min: {df[col].min()}")
    print(f"Max: {df[col].max()}")
    print(f"Q1: {df[col].quantile(0.25)}")
    print(f"Q3: {df[col].quantile(0.75)}")
    print(f"IQR: {df[col].quantile(0.75) - df[col].quantile(0.25)}")
```

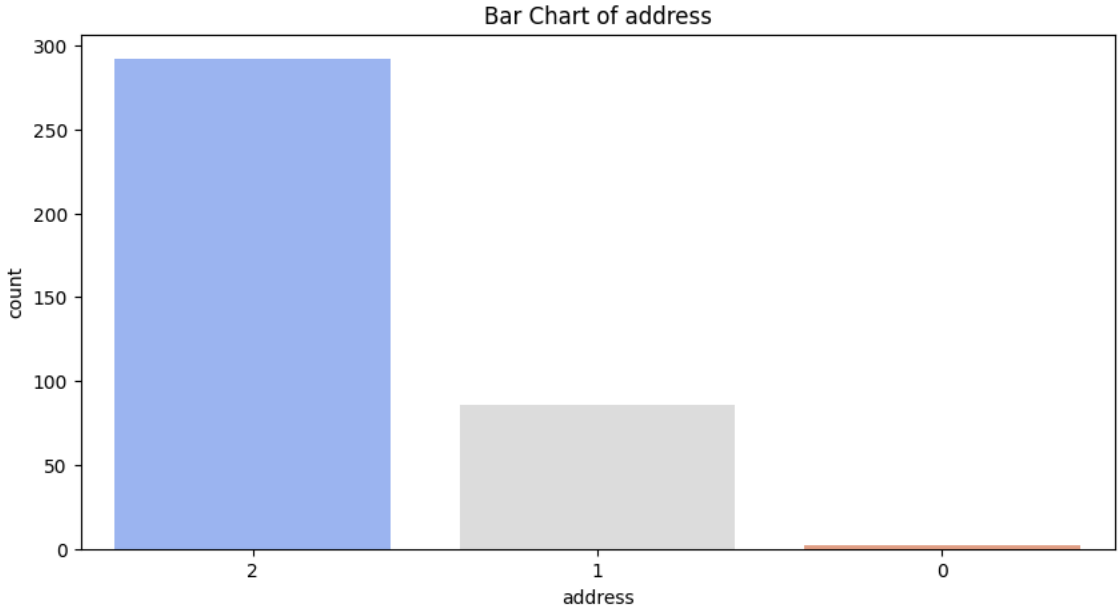
```
Min: 0
Max: 19
Q1: 9.0
Q3: 13.0
IQR: 4.0

G3:
Range: 20
Variance: 21.27343424524371
Standard Deviation: 4.612313329040397
Min: 0
Max: 20
Q1: 8.0
Q3: 14.0
IQR: 6.0
```

```
categorical_cols = ['sex', 'address', 'famsize', 'schoolsup', 'age_group']
for cat_col in categorical_cols:
    if cat_col in df.columns:
        plt.figure(figsize=(10,5))
        sns.countplot(x=df[cat_col].astype(str), palette='coolwarm')
        plt.title(f'Bar Chart of {cat_col}')
        plt.show()
    else:
        print(f"Column '{cat_col}' not found in DataFrame, skipping.")
```

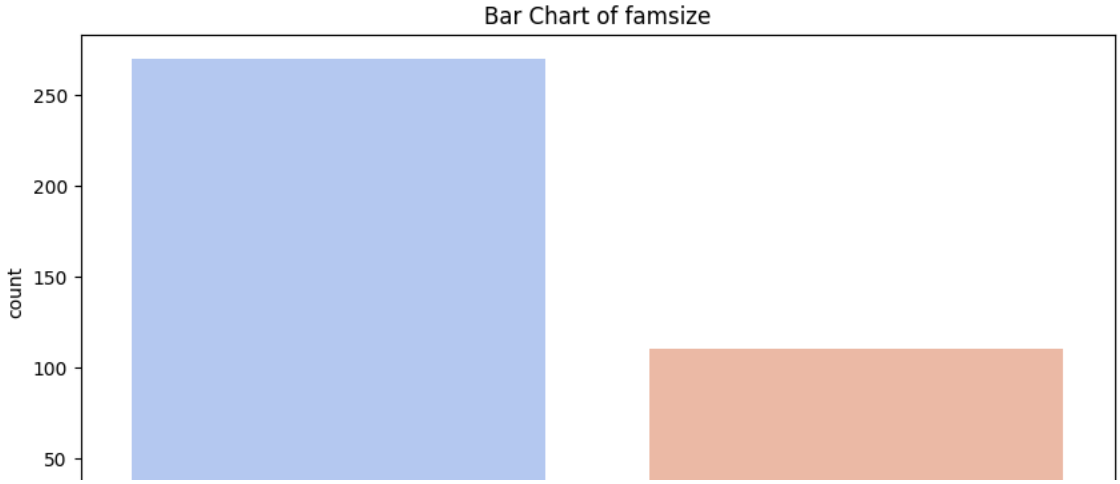
```
<ipython-input-37-c8f26cb134f5>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  sns.countplot(x=df[cat_col].astype(str), palette='coolwarm')
```

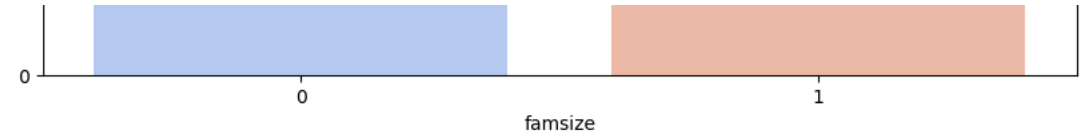



```
<ipython-input-37-c8f26cb134f5>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  sns.countplot(x=df[cat_col].astype(str), palette='coolwarm')
```

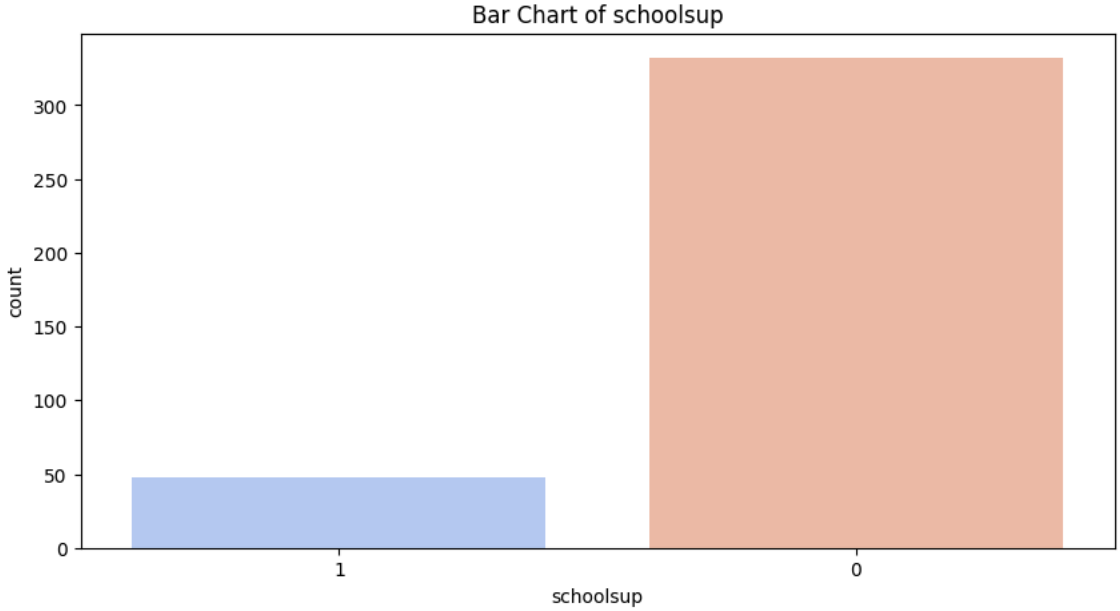



```
<ipython-input-37-c8f26cb134f5>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  sns.countplot(x=df[cat_col].astype(str), palette='coolwarm')
```



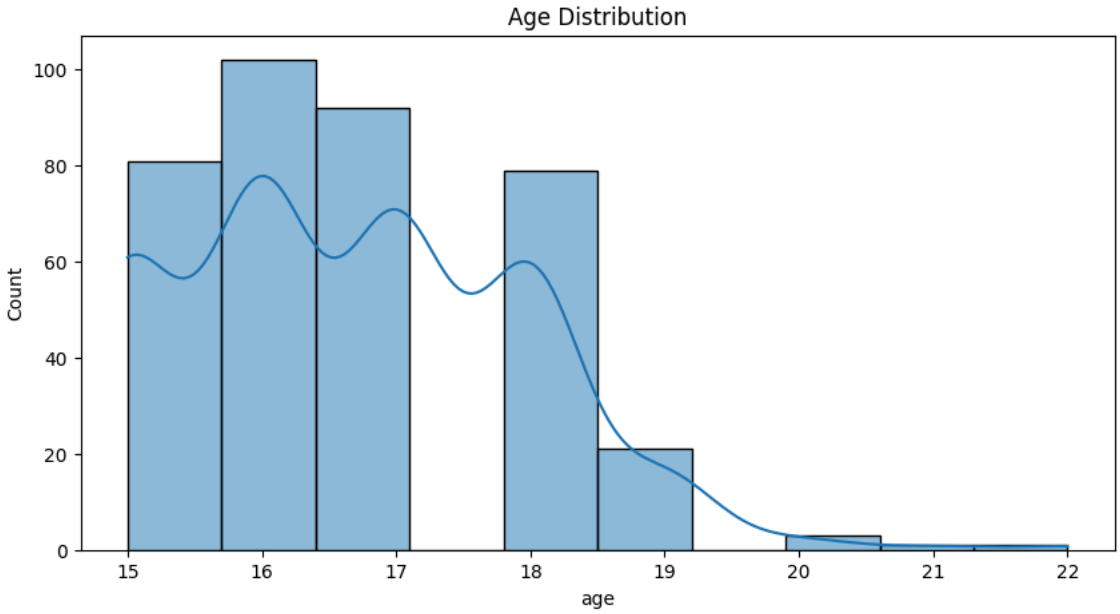



```
<ipython-input-37-c8f26cb134f5>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  sns.countplot(x=df[cat_col].astype(str), palette='coolwarm')
```

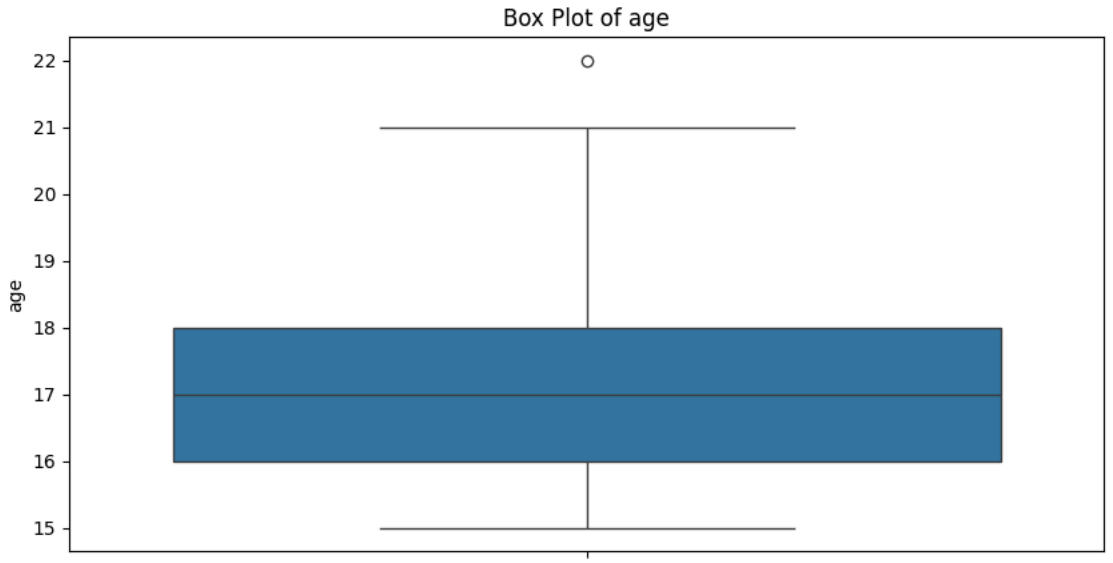


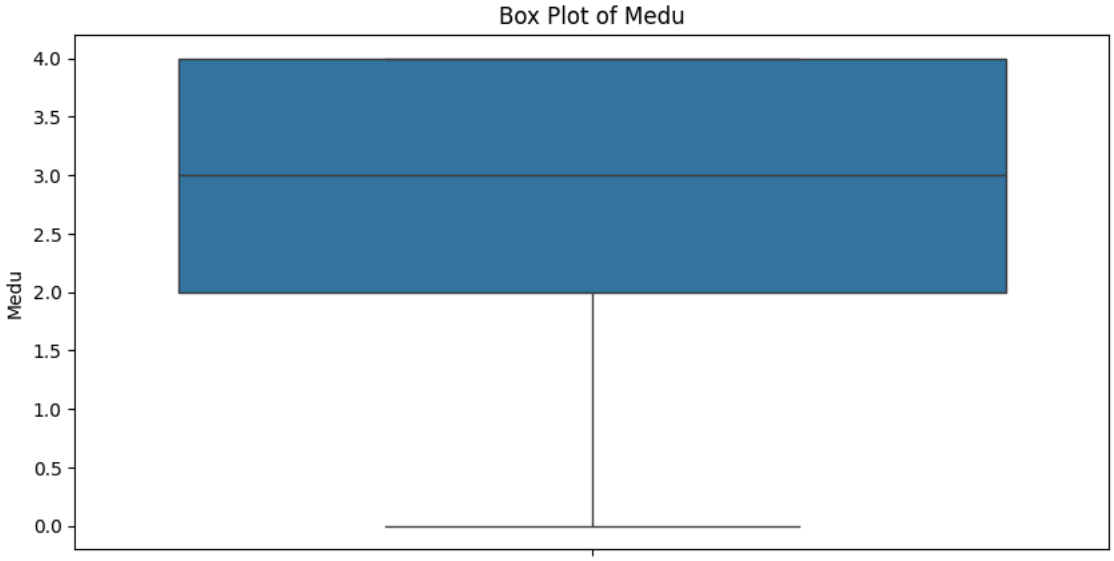

```
Column 'age_group' not found in DataFrame, skipping.
```

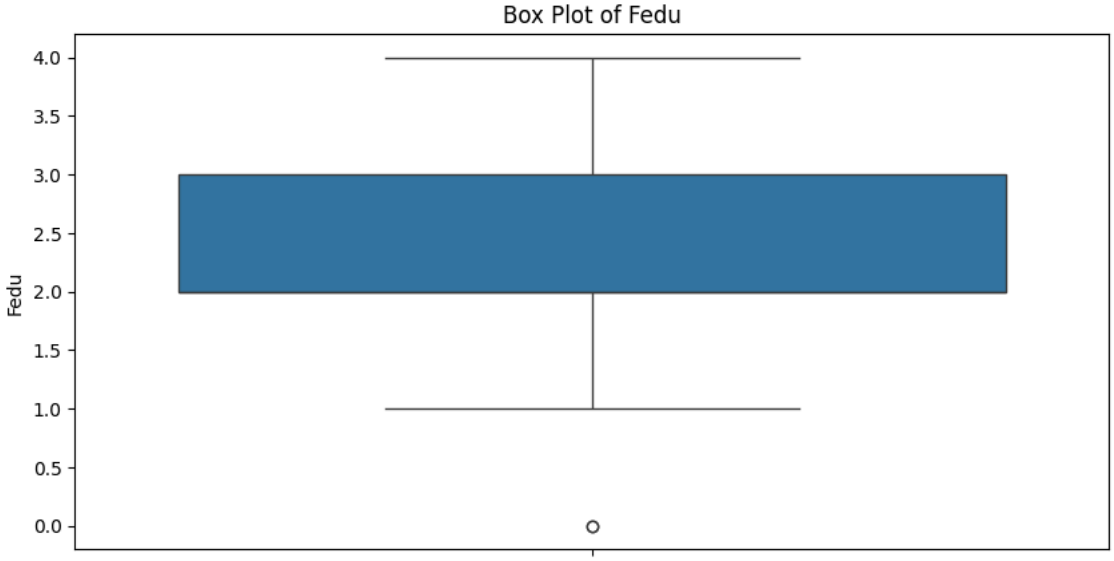
```
plt.figure(figsize=(10,5))
sns.histplot(df['age'], bins=10, kde=True)
plt.title('Age Distribution')
plt.show()
```

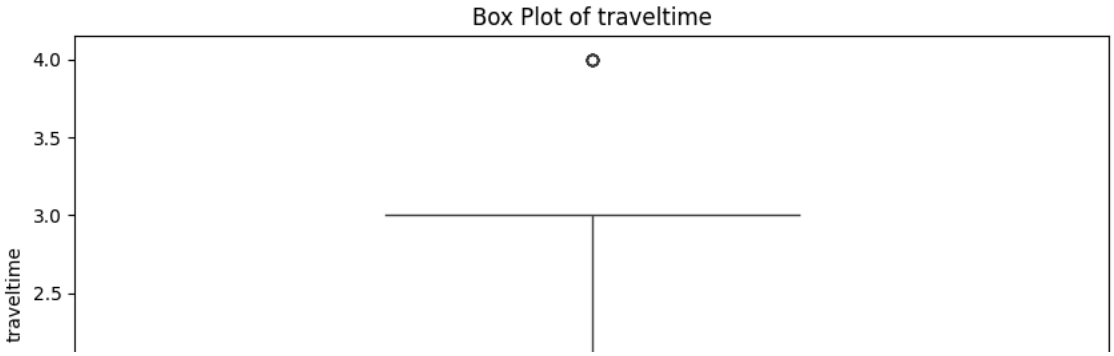


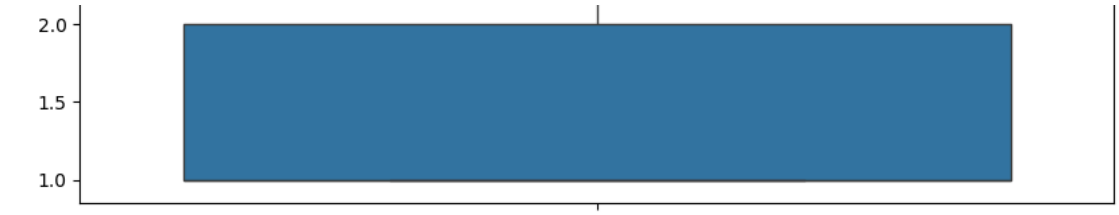

```
# Box plot for numeric columns
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    plt.figure(figsize=(10,5))
    sns.boxplot(y=df[col])
    plt.title(f'Box Plot of {col}')
    plt.show()
```

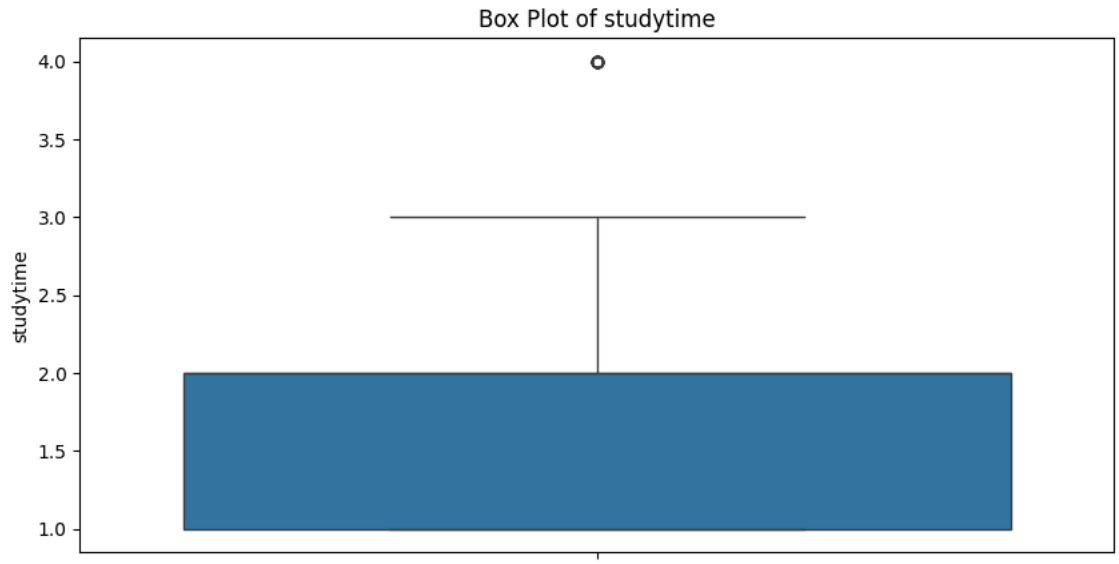

Box Plot of age
age
22
21
20
19
18
17
16
15

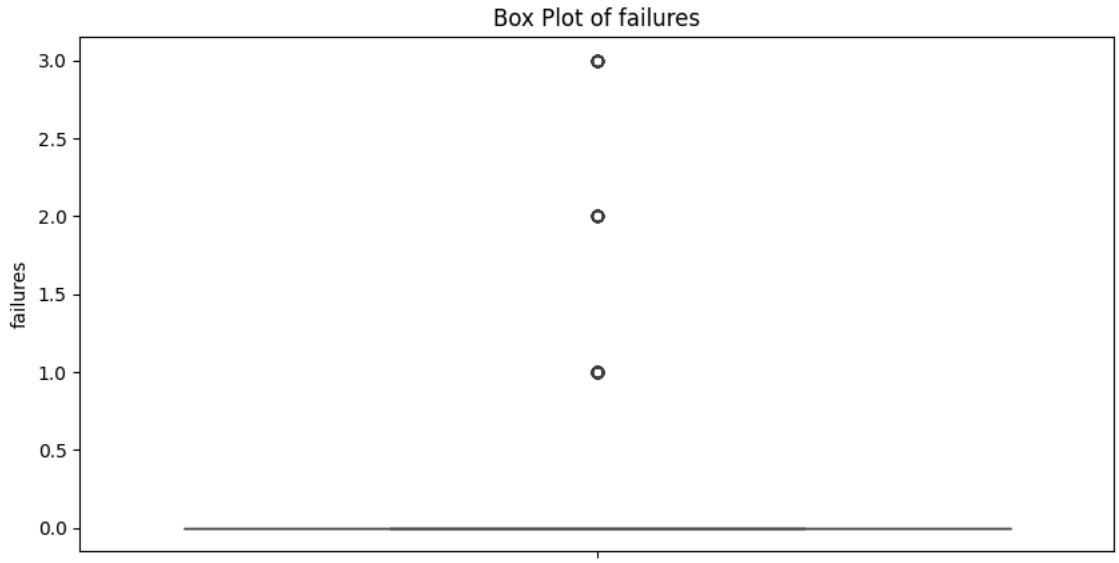

Box Plot of Medu
Medu
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

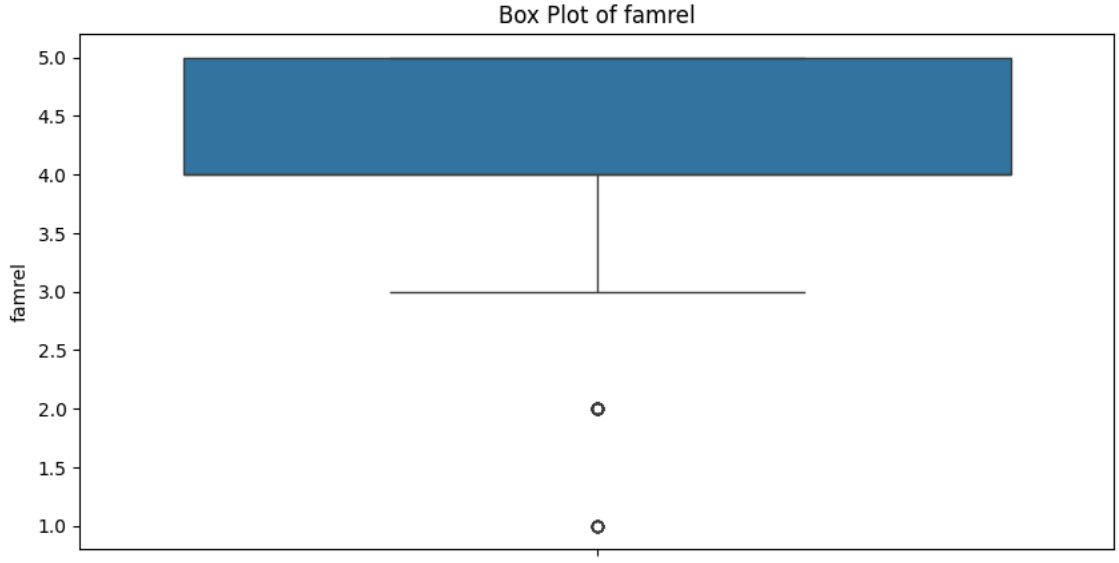

Box Plot of Fedu
Fedu
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

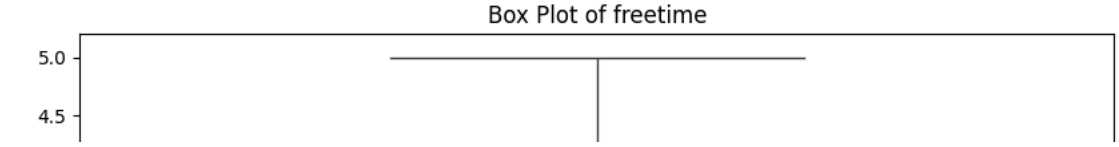

Box Plot of traveltime
traveltime
4.0
3.5
3.0
2.5

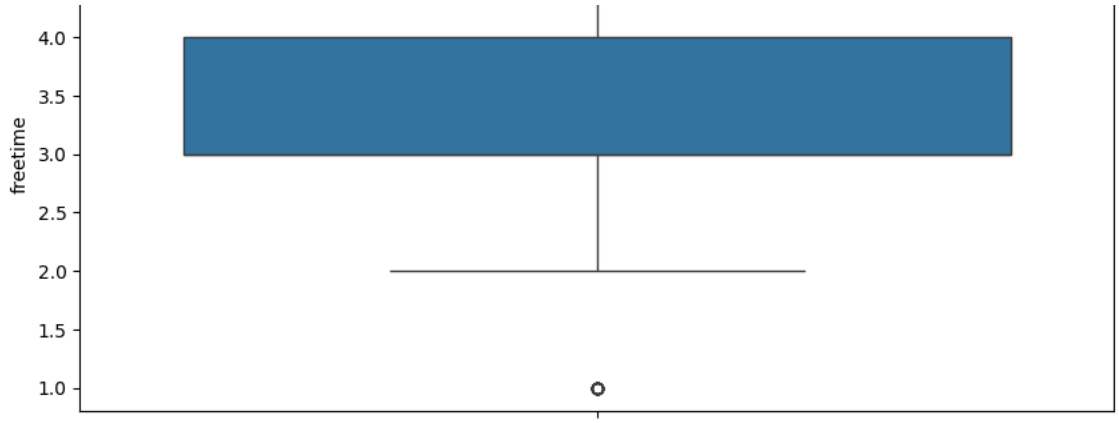

2.0
1.5
1.0

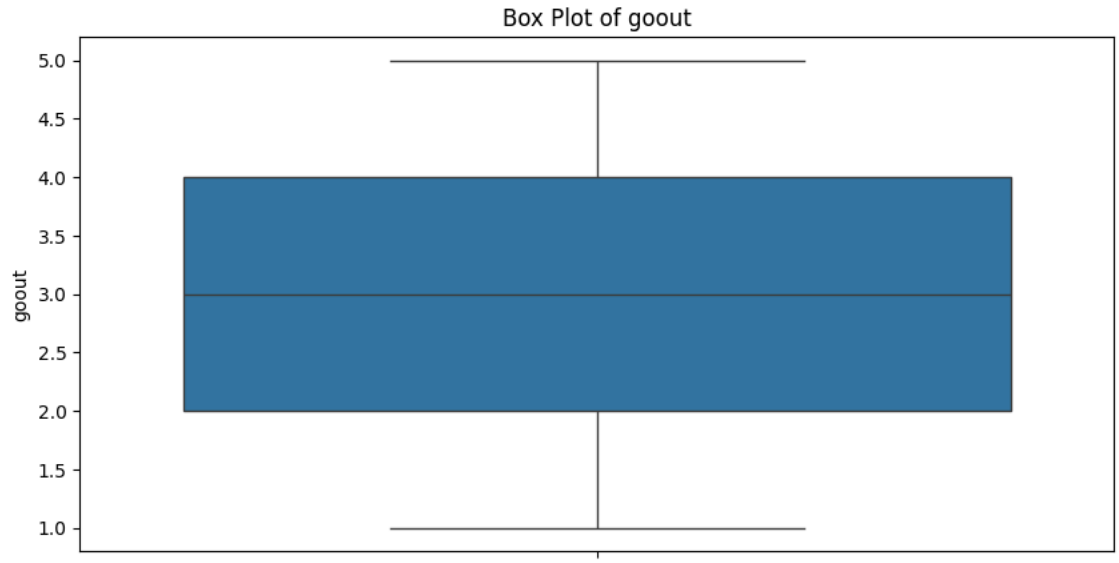

Box Plot of studytime
studytime
4.0
3.5
3.0
2.5
2.0
1.5
1.0

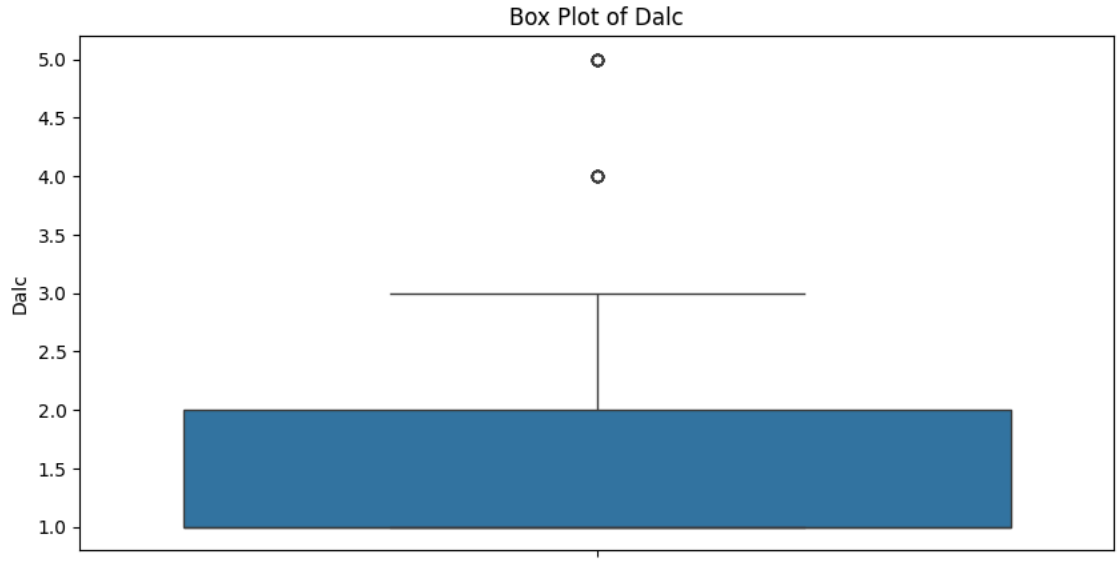

Box Plot of failures
failures
3.0
2.5
2.0
1.5
1.0
0.5
0.0

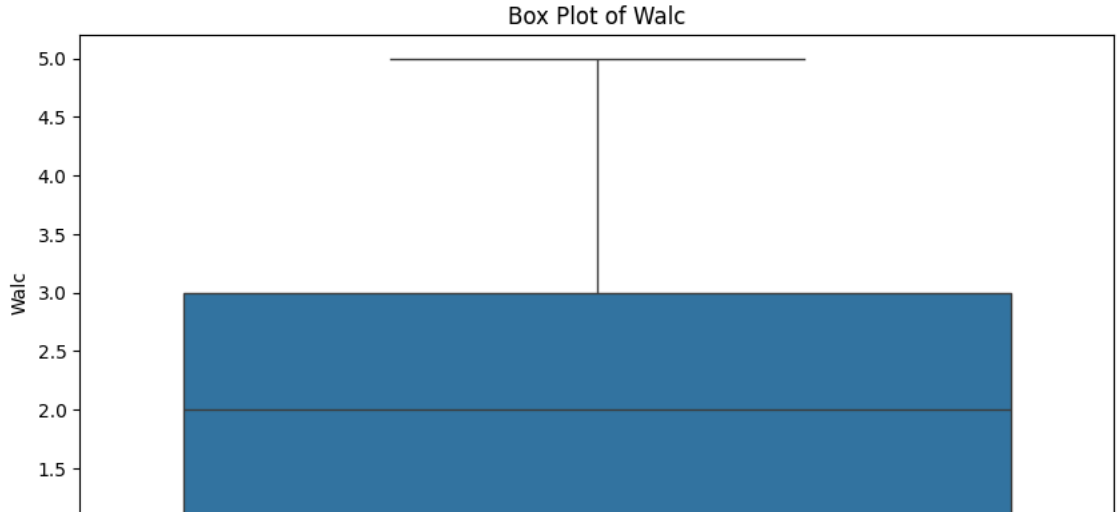

Box Plot of famrel
famrel
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0


Box Plot of freetime
5.0
4.5

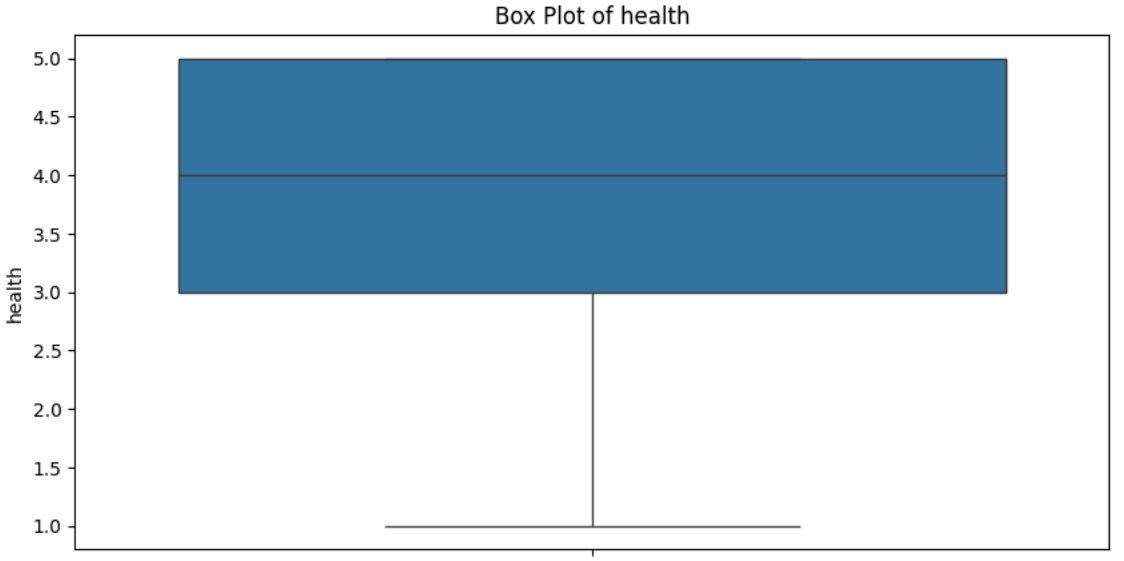

freetime
4.0
3.5
3.0
2.5
2.0
1.5
1.0

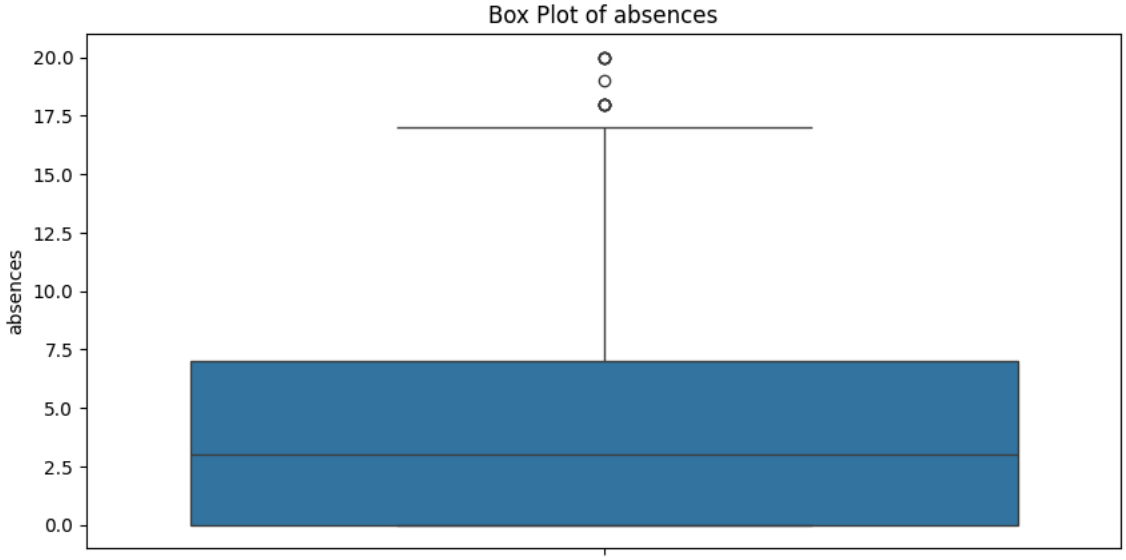

Box Plot of goout
goout
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0

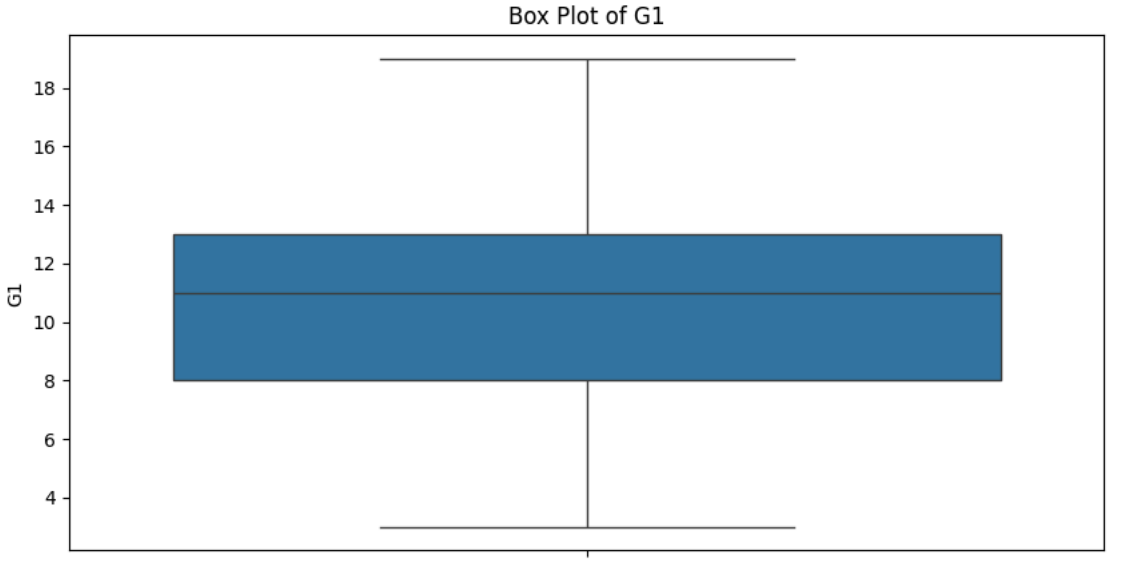

Box Plot of Dalc
Dalc
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0

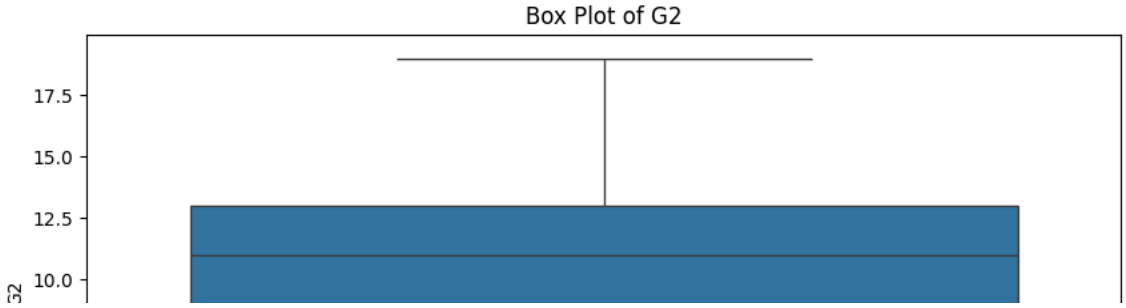

Box Plot of Walc
Walc
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5

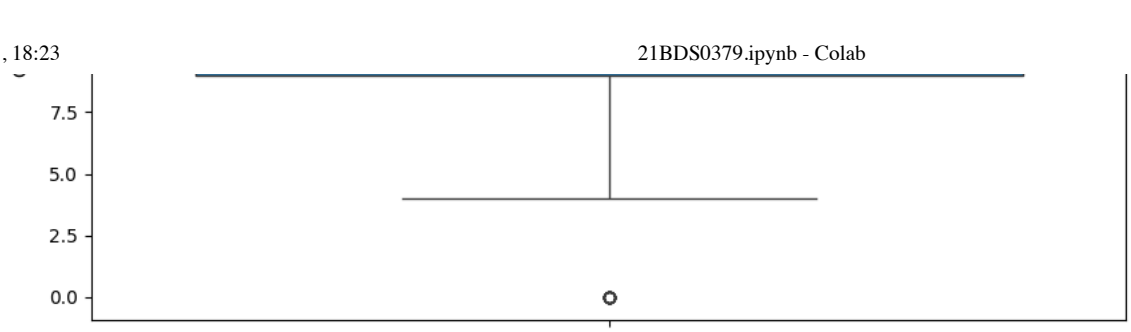

1.0

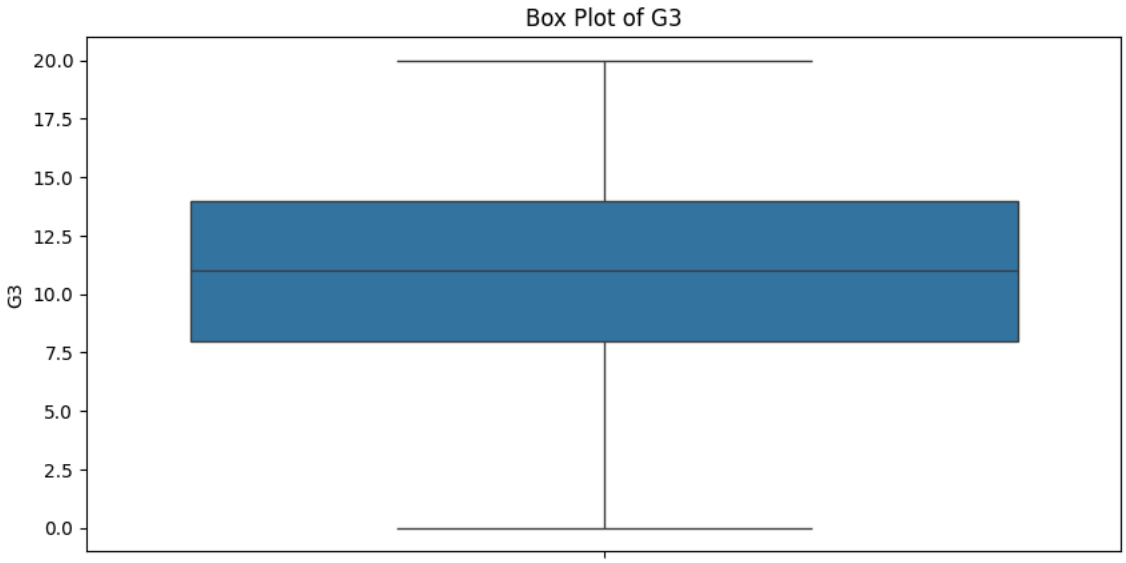

Box Plot of health
health
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0


Box Plot of absences
absences
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0


Box Plot of G1
G1
18
16
14
12
10
8
6
4


Box Plot of G2
17.5
15.0
12.5
10.0

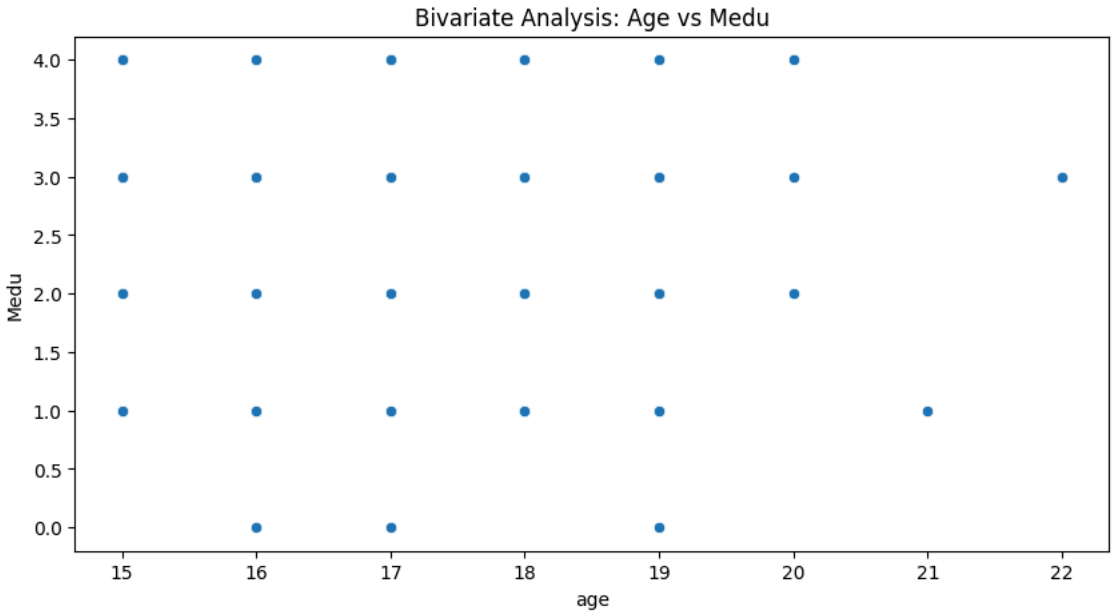

7.5
5.0
2.5
0.0

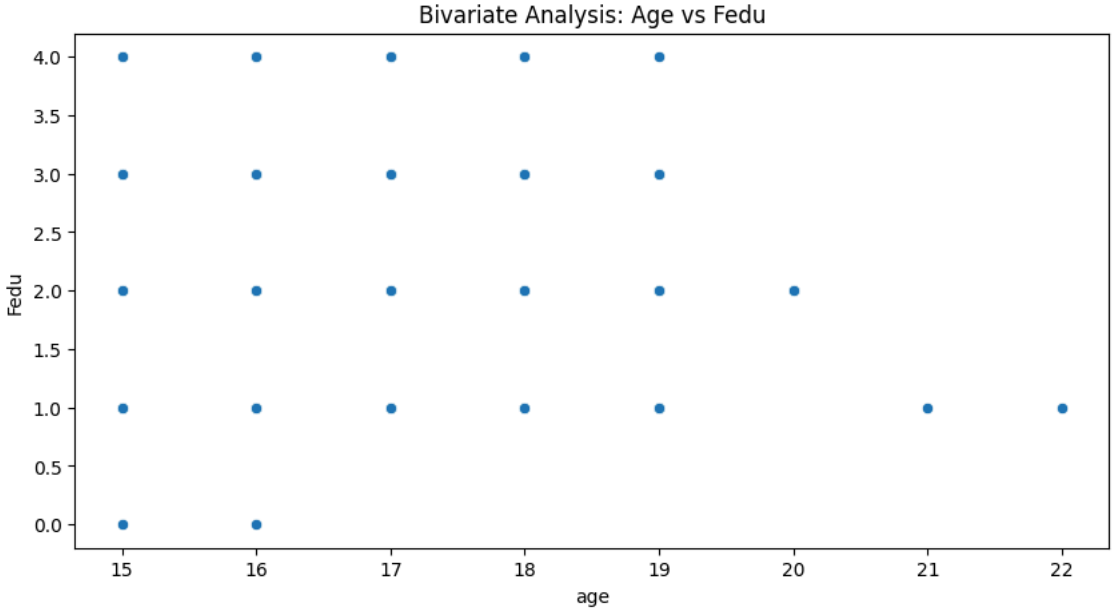

Box Plot of G3
G3
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0

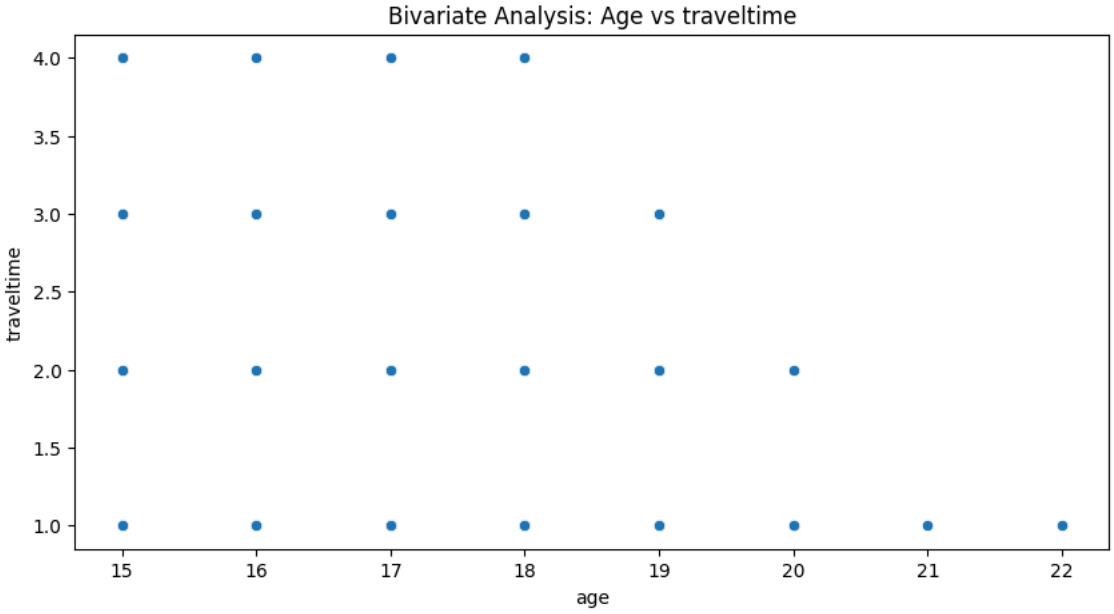
Double-click (or enter) to edit

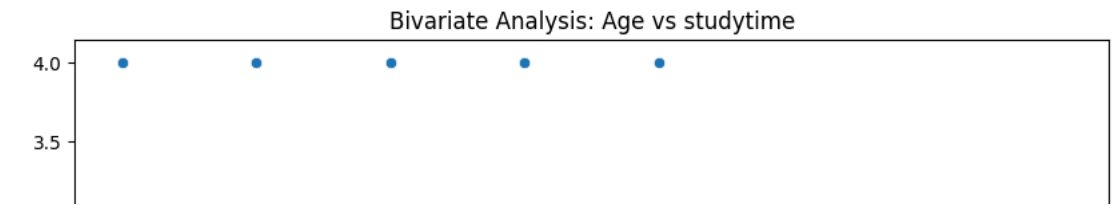
Bivariate Analysis

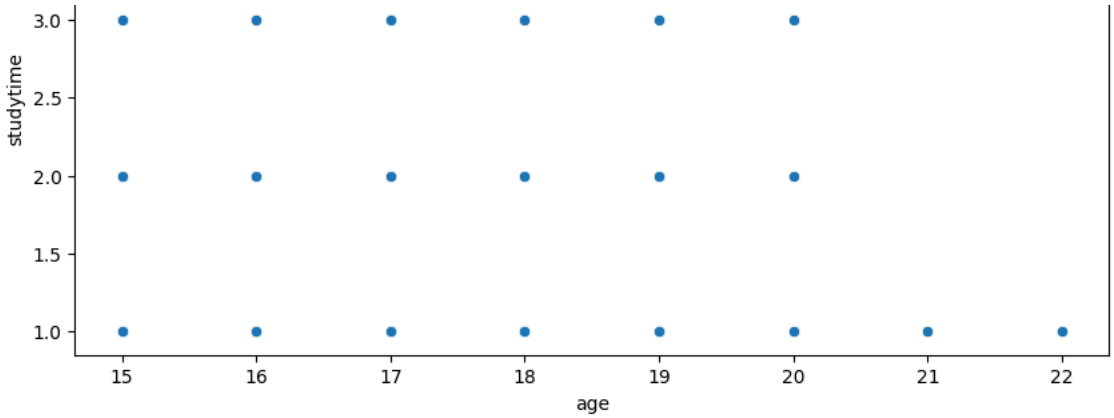
```
#Bivariate Analysis
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    if col != 'age':  # Avoid redundant comparison
        plt.figure(figsize=(10,5))
        sns.scatterplot(x=df['age'], y=df[col])
        plt.title(f'Bivariate Analysis: Age vs {col}')
        plt.show()
```

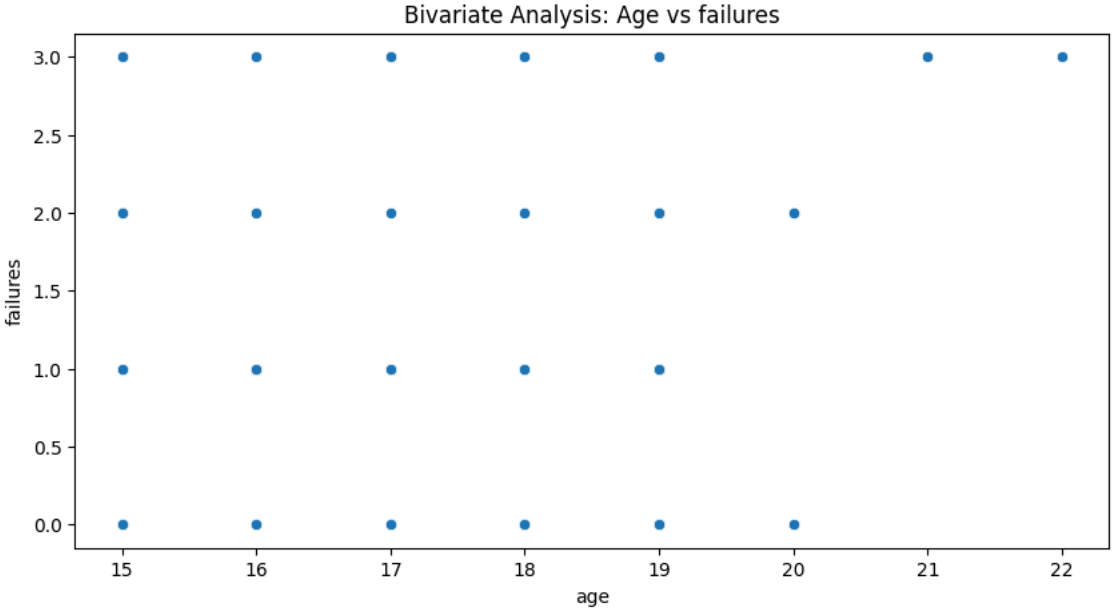

Bivariate Analysis: Age vs Medu
Medu
age
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
15
16
17
18
19
20
21
22

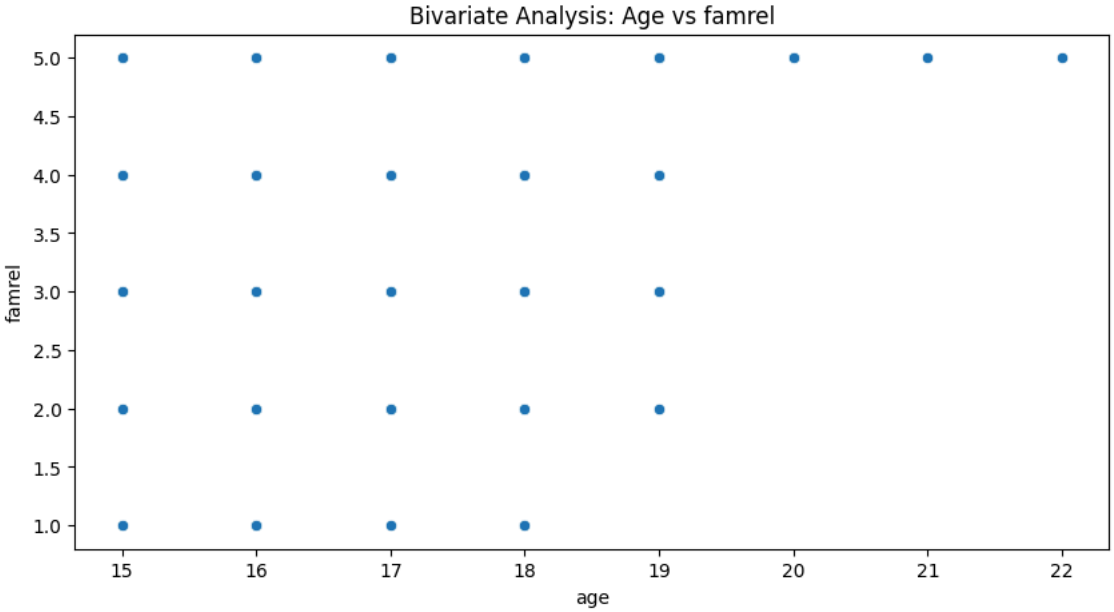

Bivariate Analysis: Age vs Fedu
Fedu
age
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
15
16
17
18
19
20
21
22

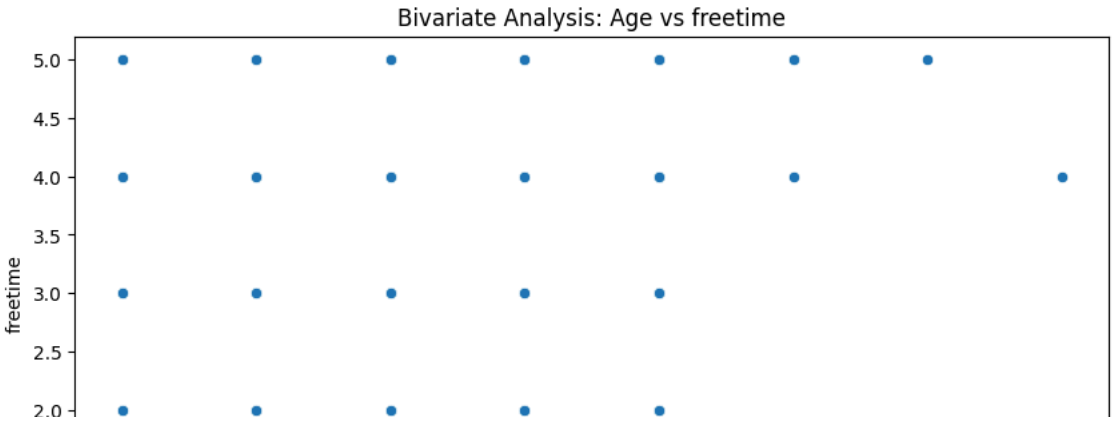

Bivariate Analysis: Age vs traveltime
traveltime
age
1.0
1.5
2.0
2.5
3.0
3.5
4.0
15
16
17
18
19
20
21
22

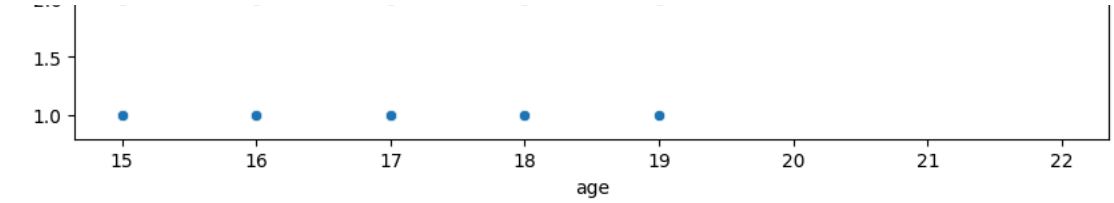

Bivariate Analysis: Age vs studytime
3.5
4.0

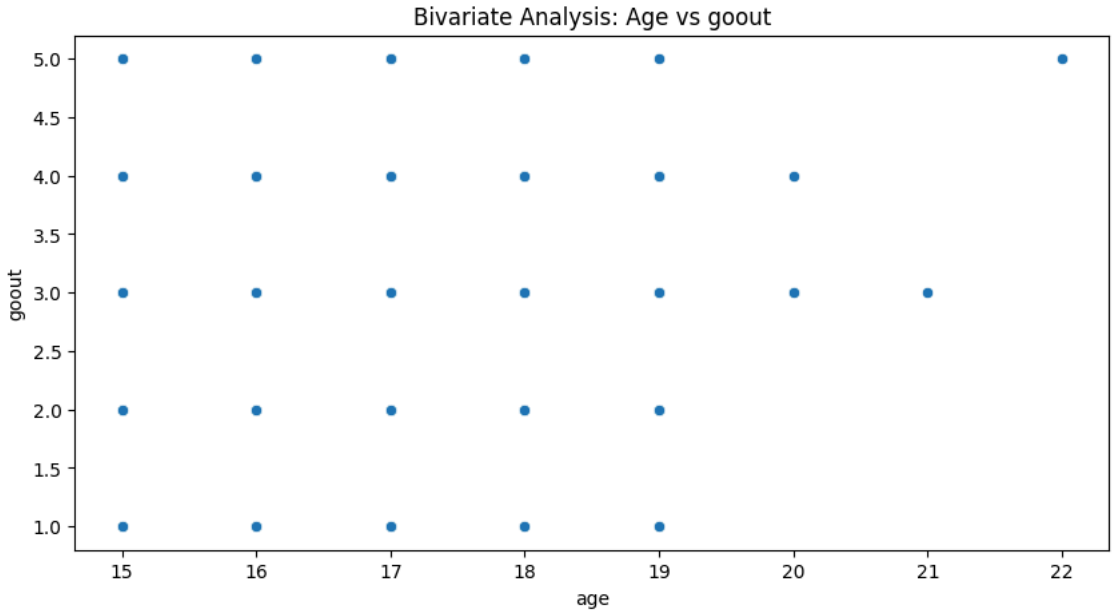

studytime
age

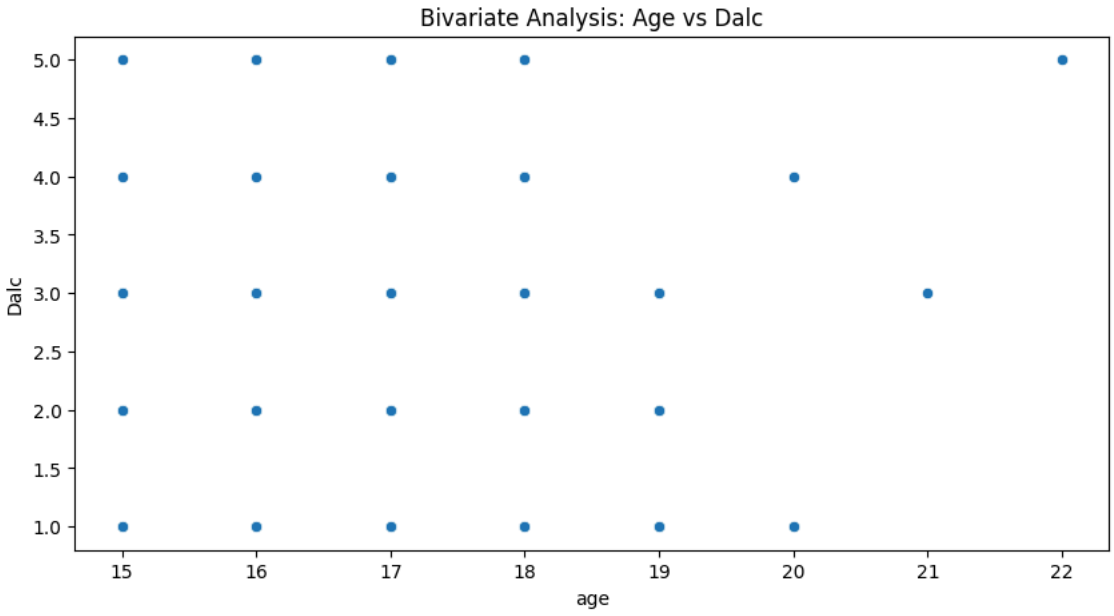

Bivariate Analysis: Age vs failures
failures
age

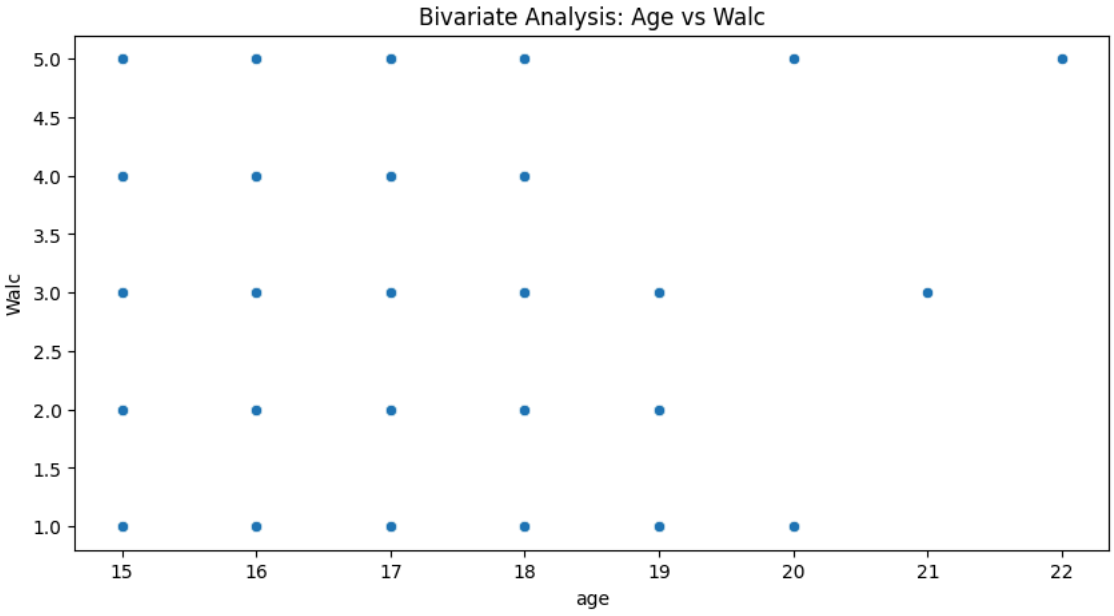

Bivariate Analysis: Age vs famrel
famrel
age

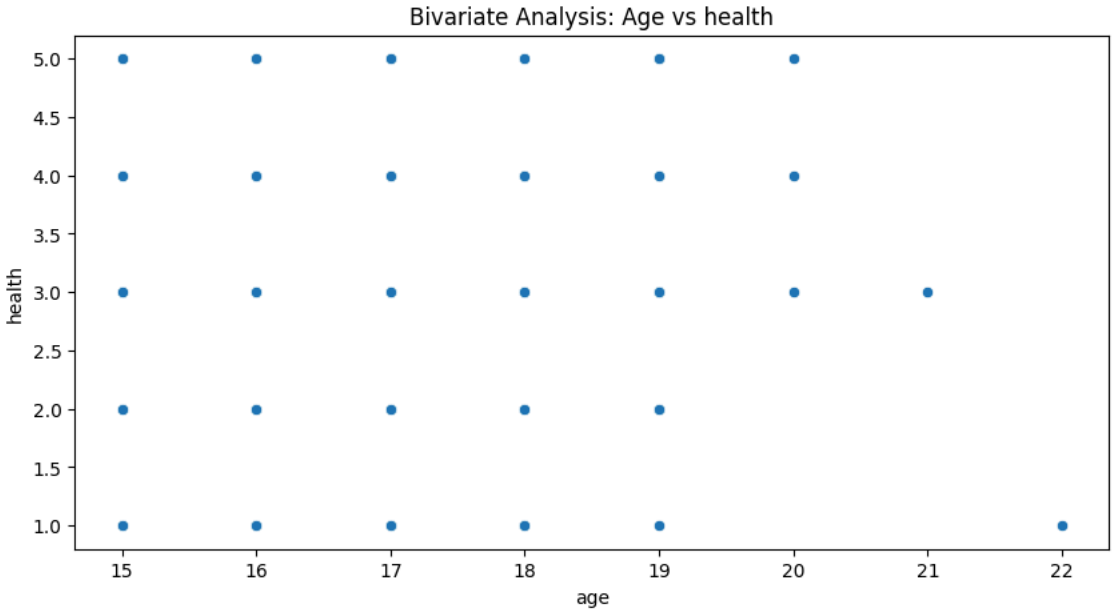

Bivariate Analysis: Age vs freetime
freetime

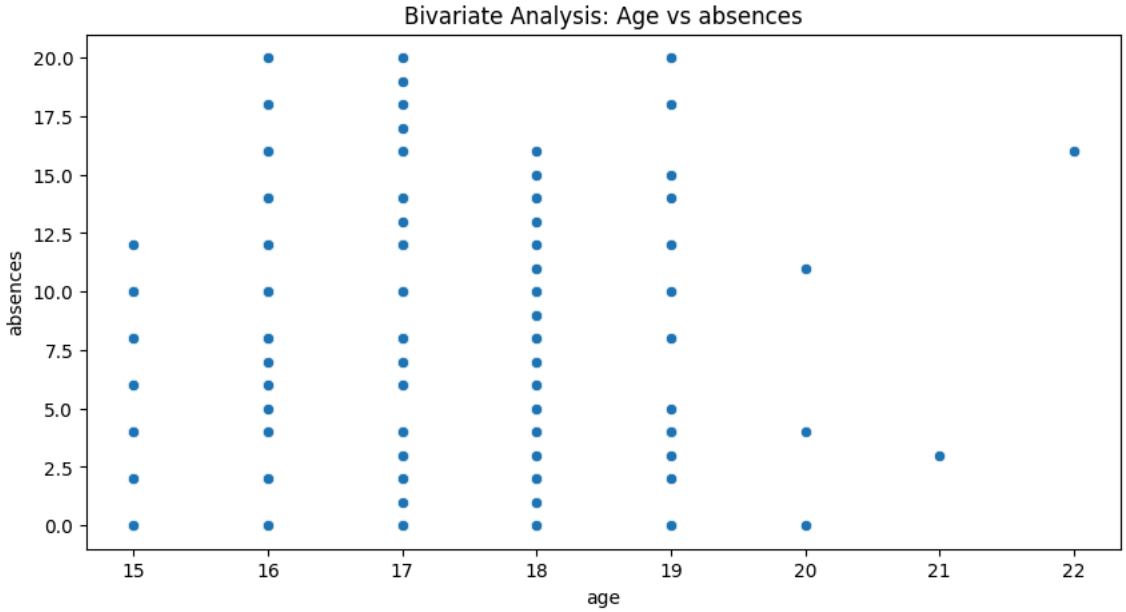

1.5
1.0
15
16
17
18
19
20
21
22
age

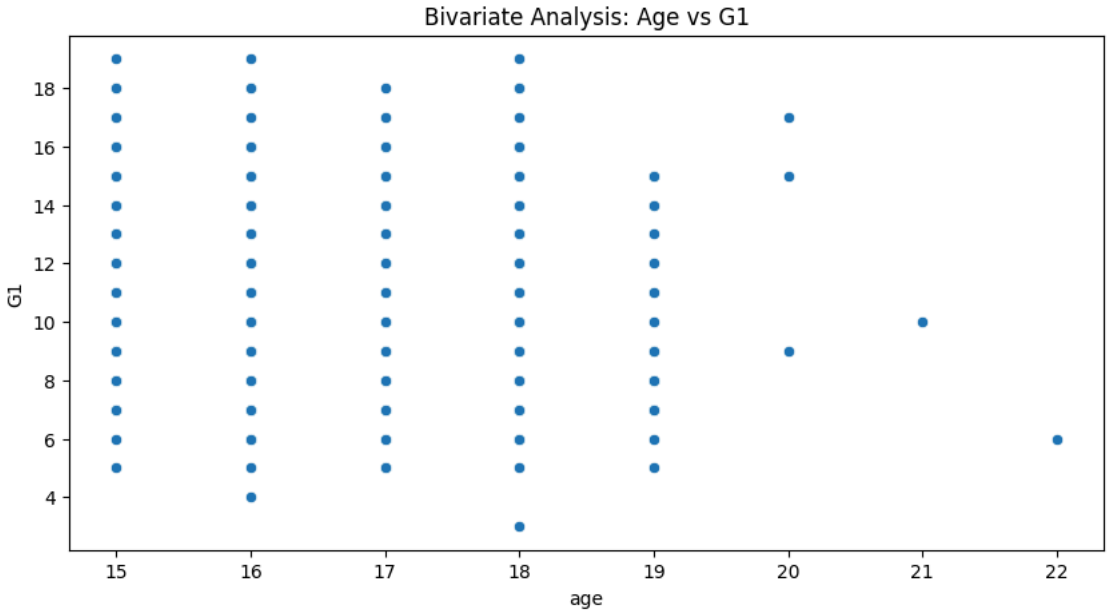

Bivariate Analysis: Age vs goout
goout
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
15
16
17
18
19
20
21
22
age

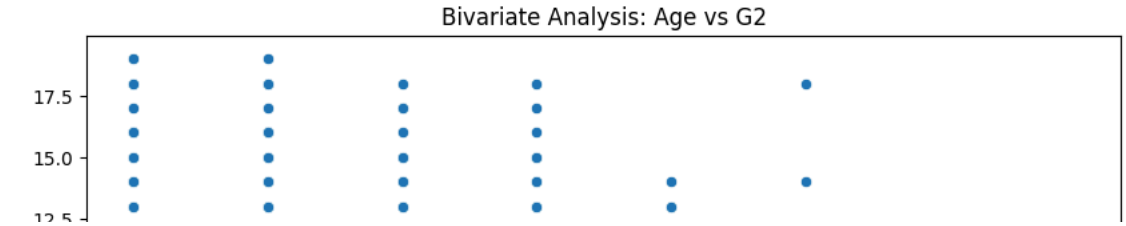

Bivariate Analysis: Age vs Dalc
Dalc
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
15
16
17
18
19
20
21
22
age

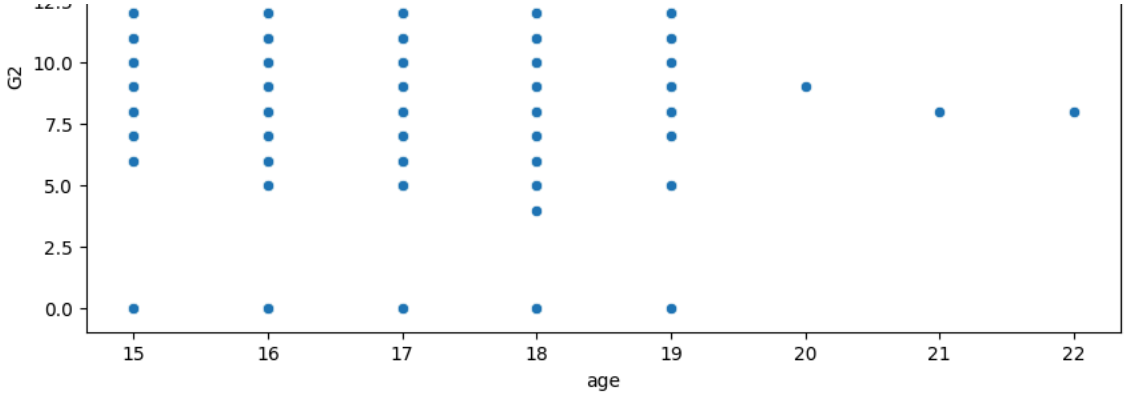

Bivariate Analysis: Age vs Walc
Walc
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
15
16
17
18
19
20
21
22
age

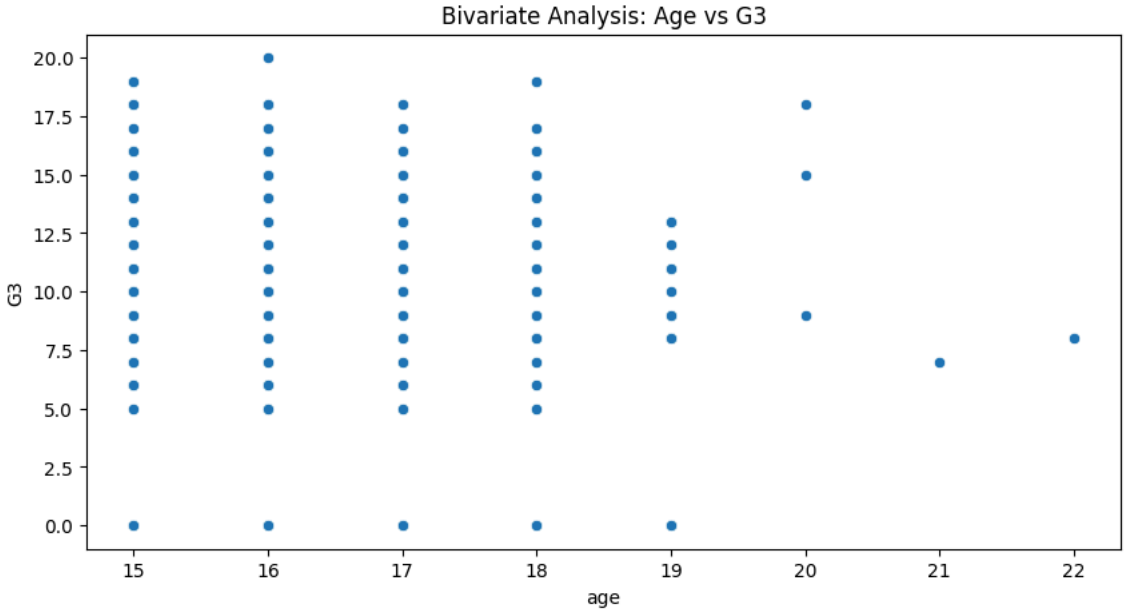

Bivariate Analysis: Age vs health
health
age
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
15
16
17
18
19
20
21
22


Bivariate Analysis: Age vs absences
absences
age
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
15
16
17
18
19
20
21
22

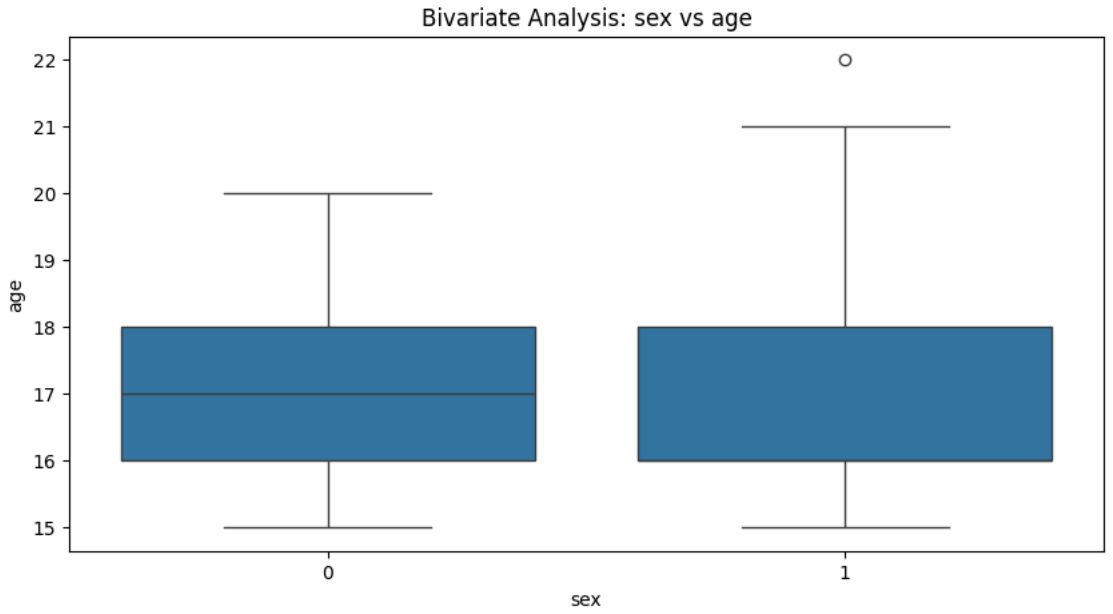

Bivariate Analysis: Age vs G1
G1
age
18
16
14
12
10
8
6
4
15
16
17
18
19
20
21
22

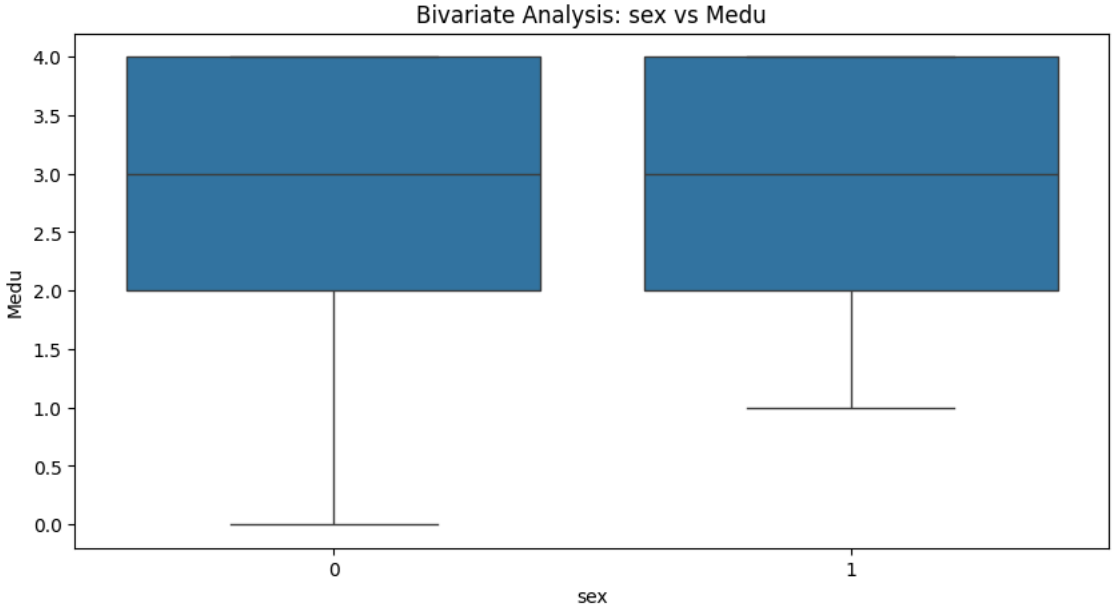

Bivariate Analysis: Age vs G2
17.5
15.0

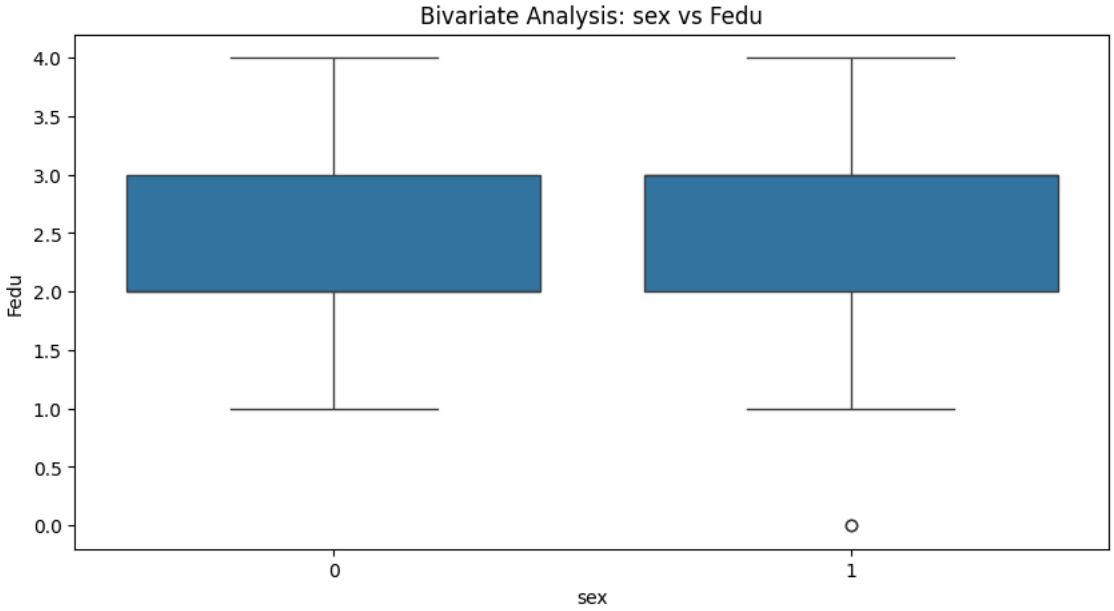

G2
12.5
10.0
7.5
5.0
2.5
0.0
15
16
17
18
19
20
21
22
age

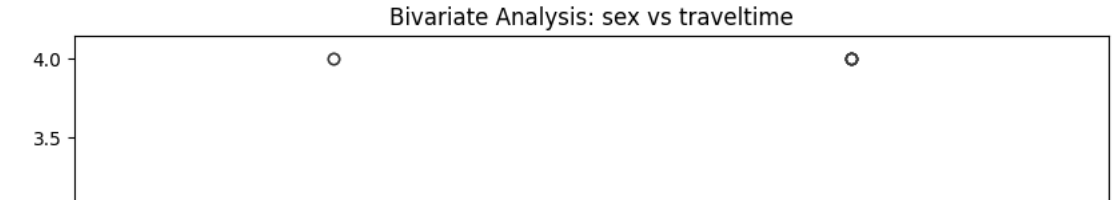

Bivariate Analysis: Age vs G3
G3
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
15
16
17
18
19
20
21
22
age

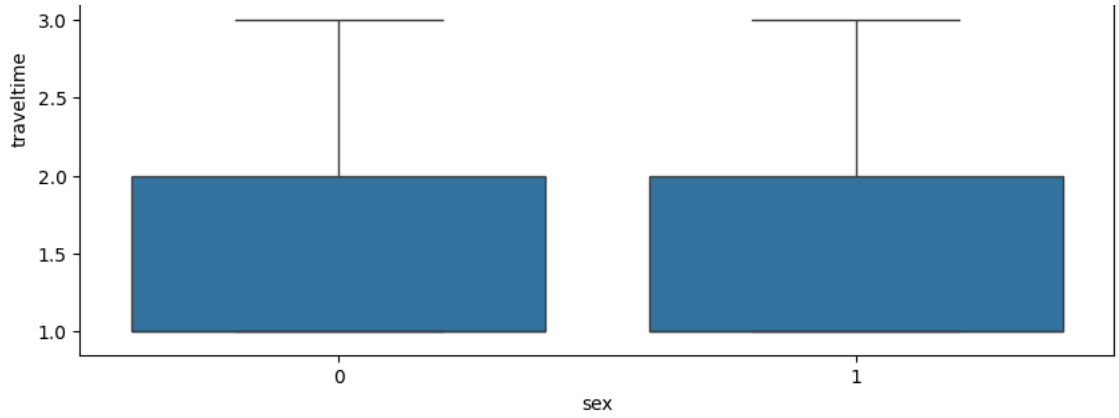
```
categorical_cols = ['sex', 'address', 'famsize', 'schoolsup', 'age_group']
for cat_col in categorical_cols:
    if cat_col in df.columns:
        for num_col in df.select_dtypes(include=['float64', 'int64']).columns:
            plt.figure(figsize=(10,5))
            sns.boxplot(x=df[cat_col].astype(str), y=df[num_col])
            plt.title(f'Bivariate Analysis: {cat_col} vs {num_col}')
            plt.show()
    else:
        print(f"Column '{cat_col}' not found in DataFrame, skipping.")
```

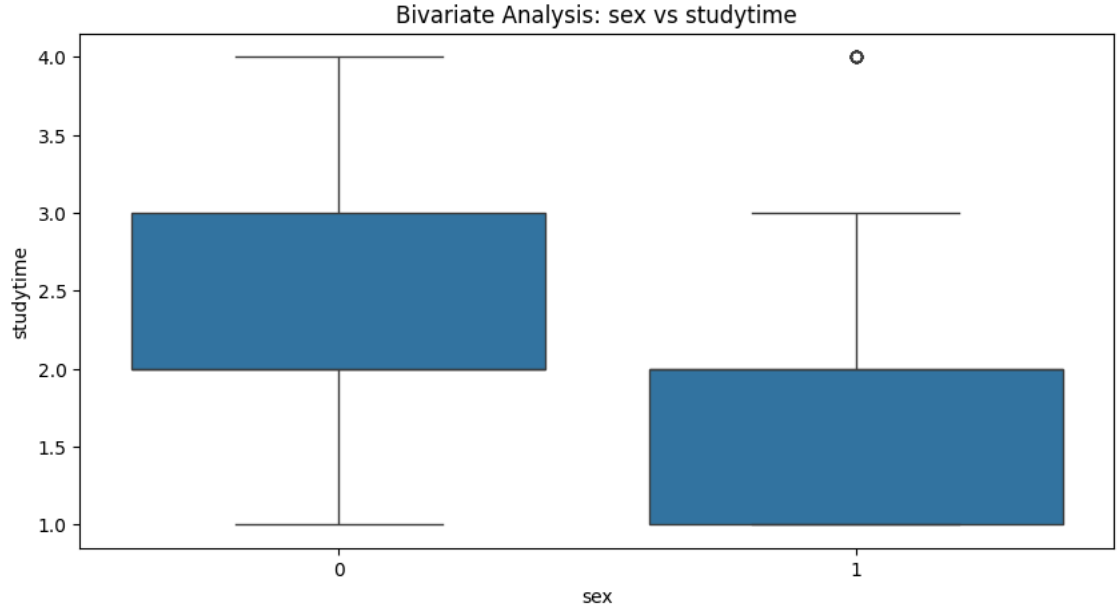

Bivariate Analysis: sex vs age
age
sex
22
21
20
19
18
17
16
15
0
1

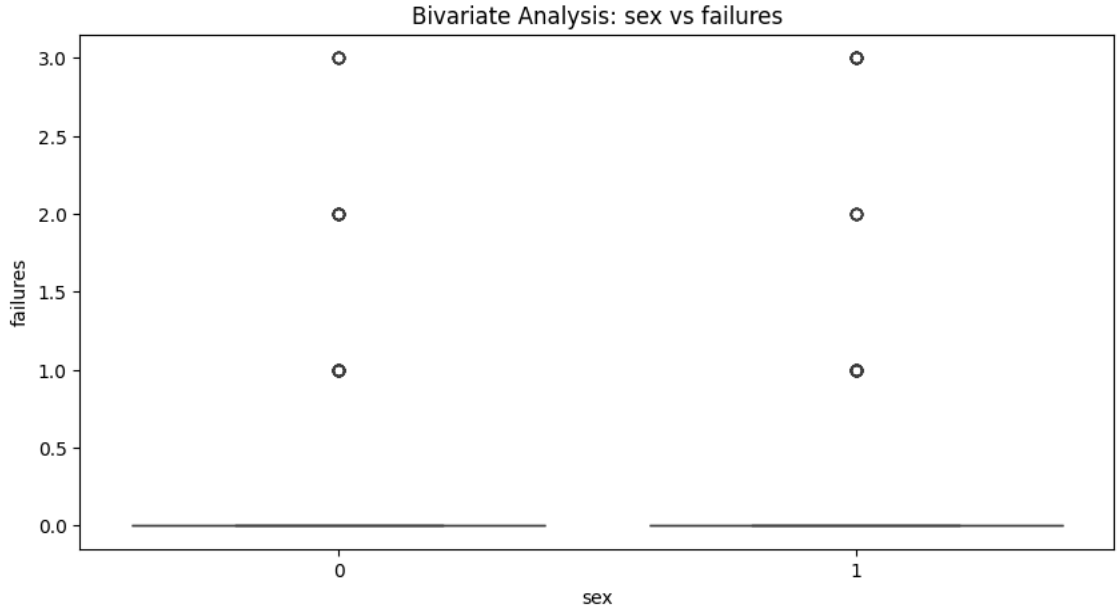

Bivariate Analysis: sex vs Medu
Medu
sex
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1

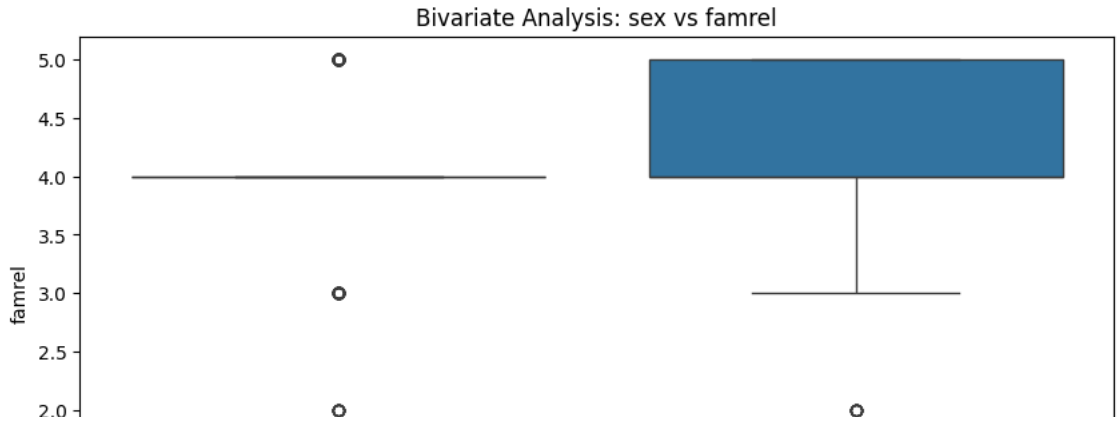

Bivariate Analysis: sex vs Fedu
Fedu
sex
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1


Bivariate Analysis: sex vs traveltime
4.0
3.5

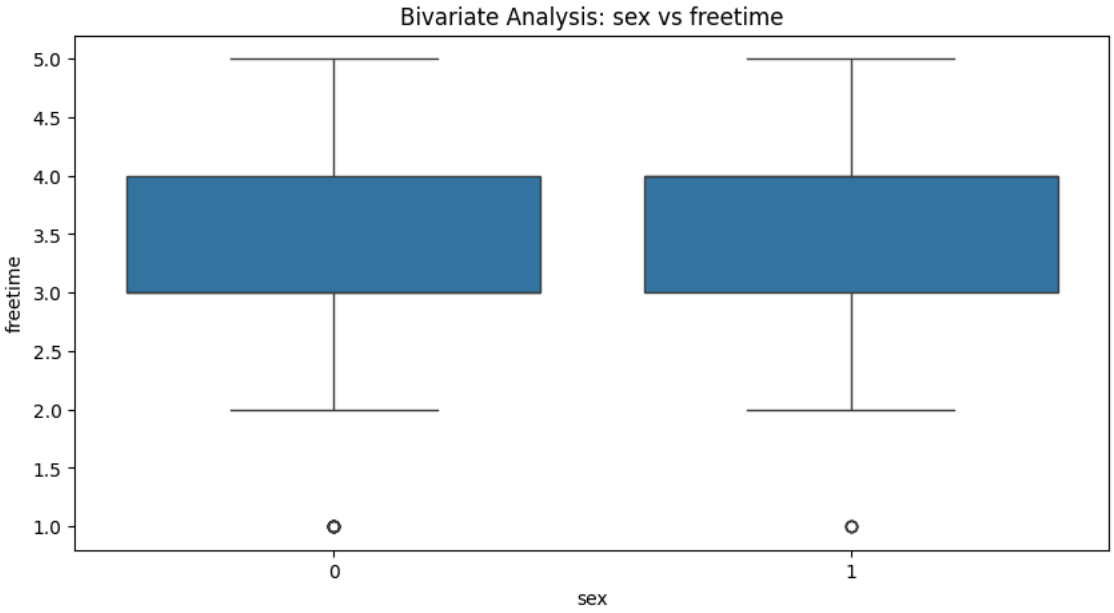

traveltime
3.0
2.5
2.0
1.5
1.0
0
1
sex

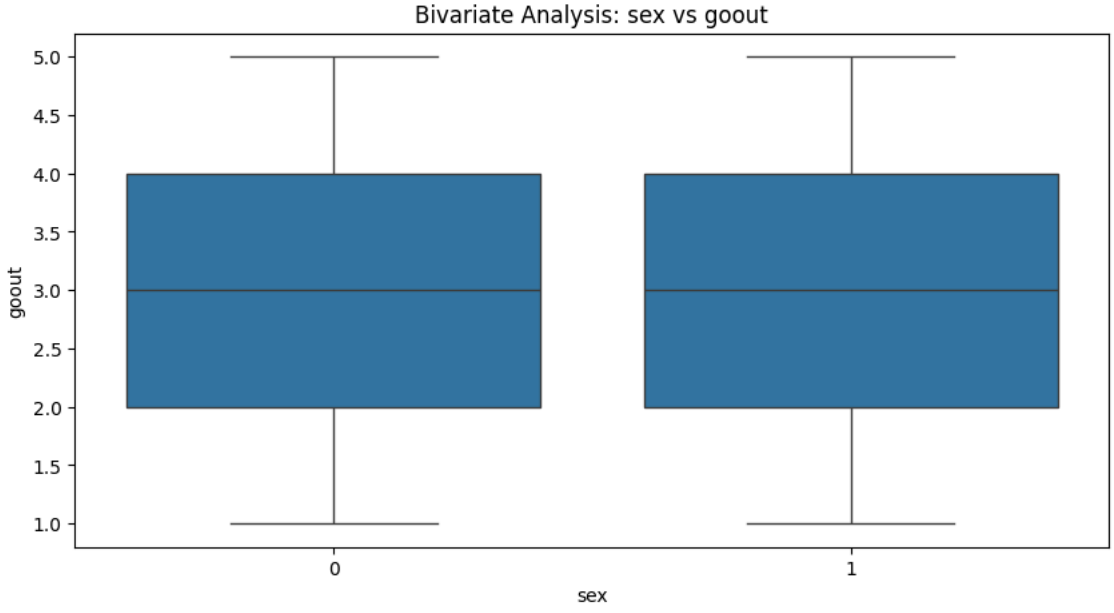

Bivariate Analysis: sex vs studytime
studytime
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
sex

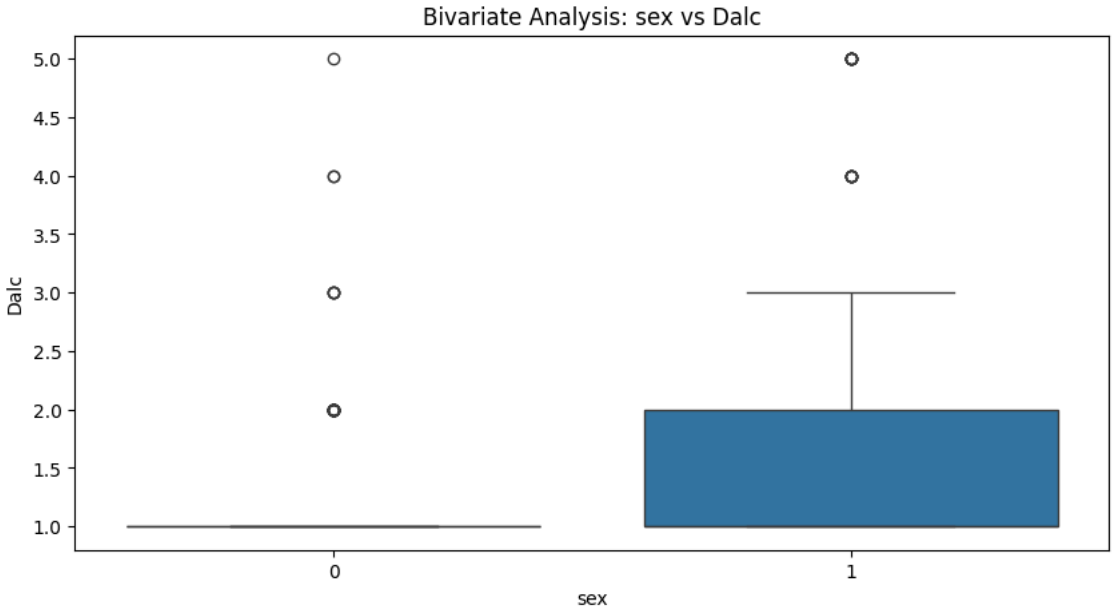

Bivariate Analysis: sex vs failures
failures
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
sex

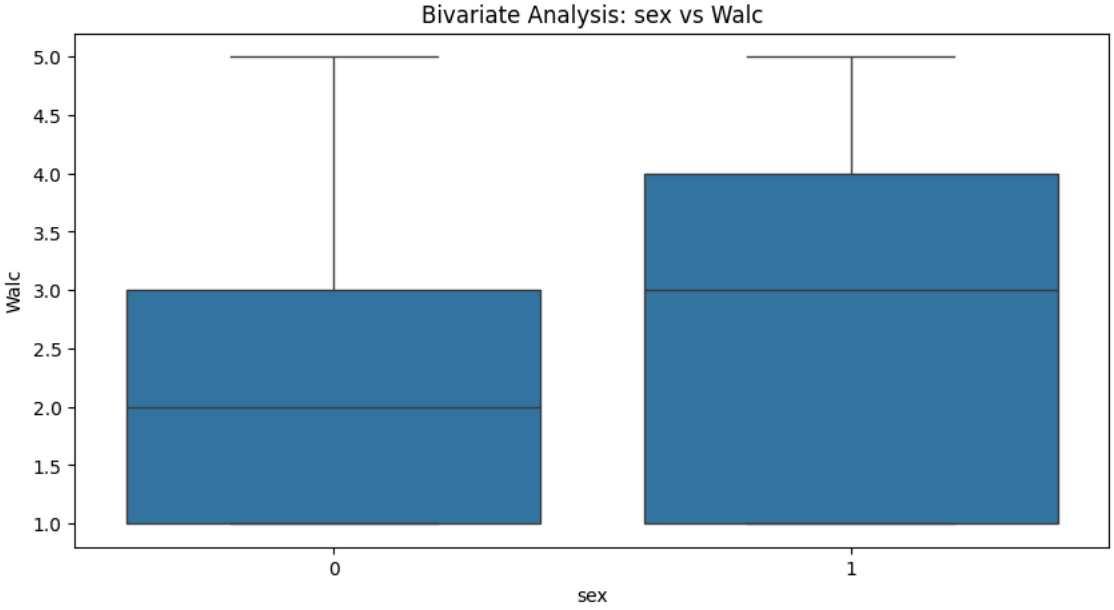

Bivariate Analysis: sex vs famrel
famrel
5.0
4.5
4.0
3.5
3.0
2.5
2.0

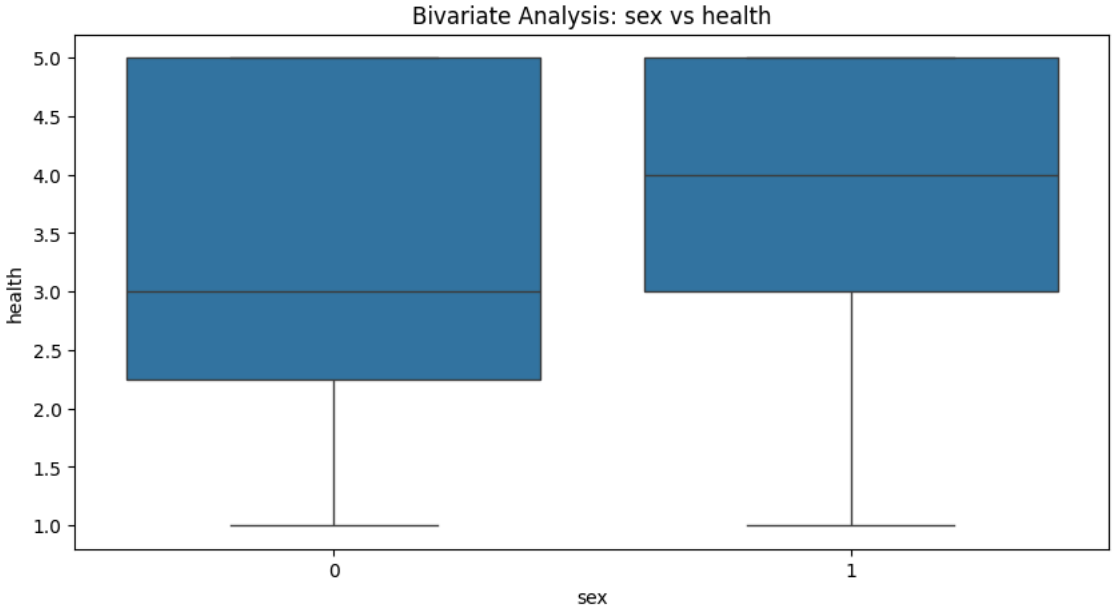

1.5
1.0
0
1
sex

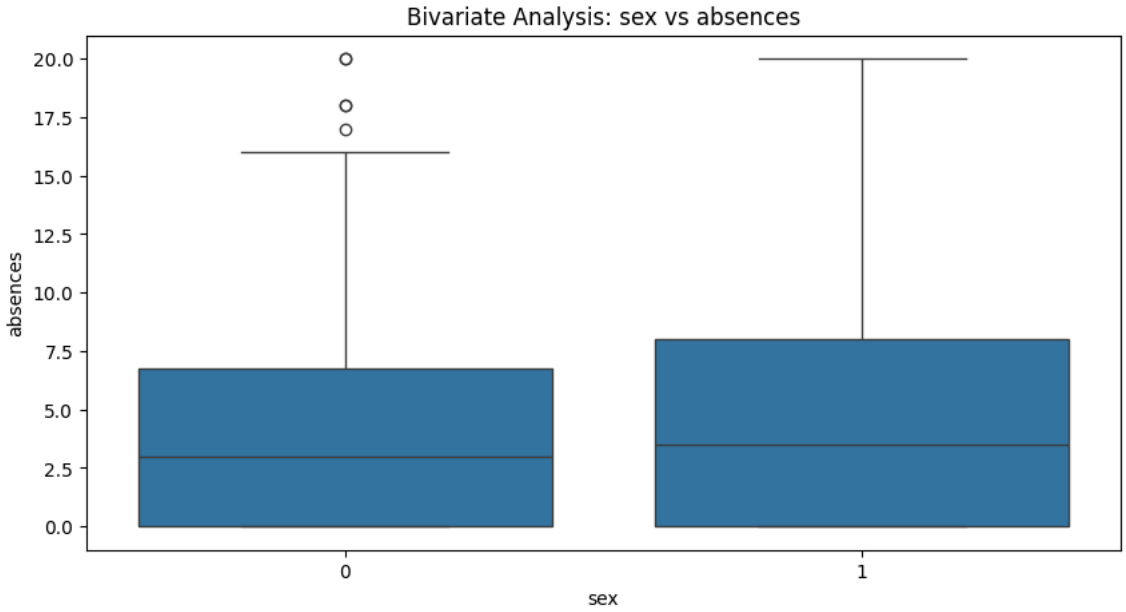

Bivariate Analysis: sex vs freetime
freetime
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
sex

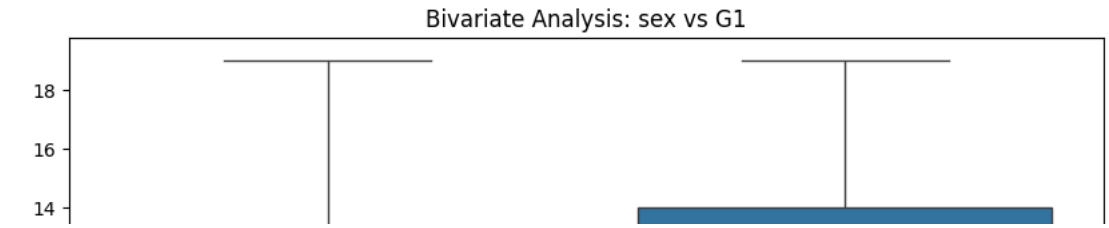

Bivariate Analysis: sex vs goout
goout
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
sex

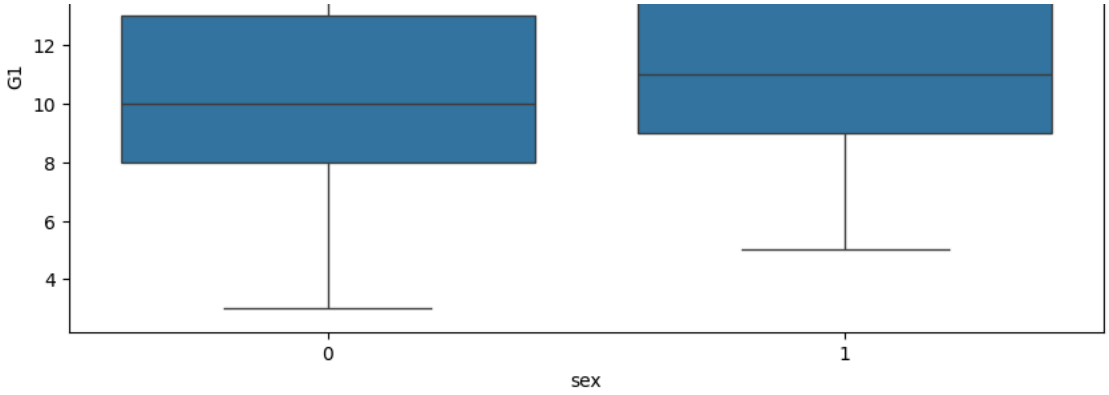

Bivariate Analysis: sex vs Dalc
Dalc
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
sex

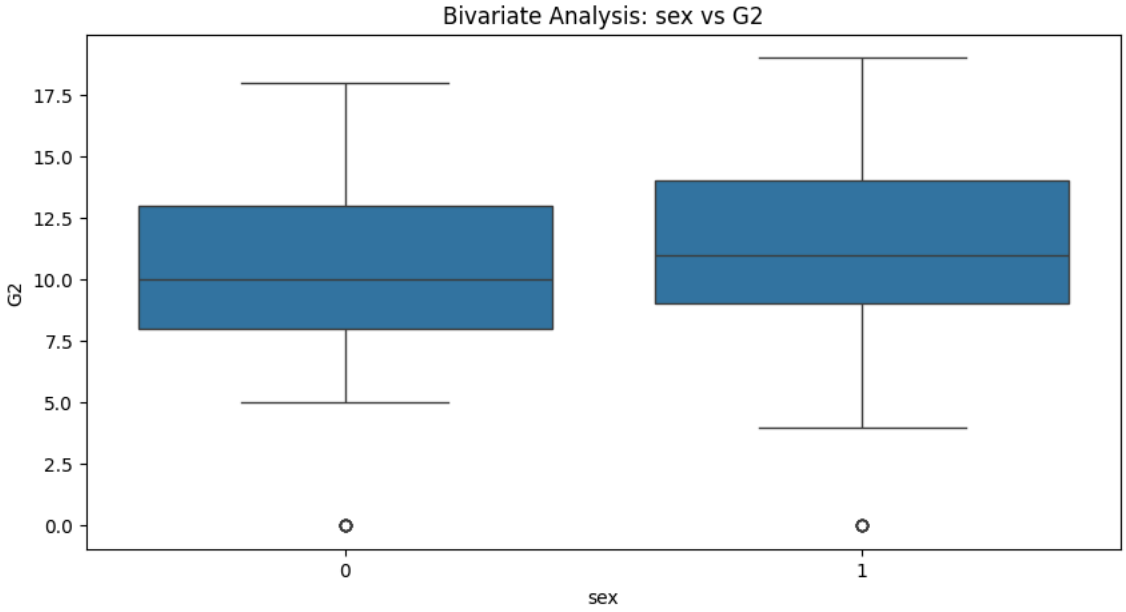

Bivariate Analysis: sex vs Walc
Walc
sex
0
1
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

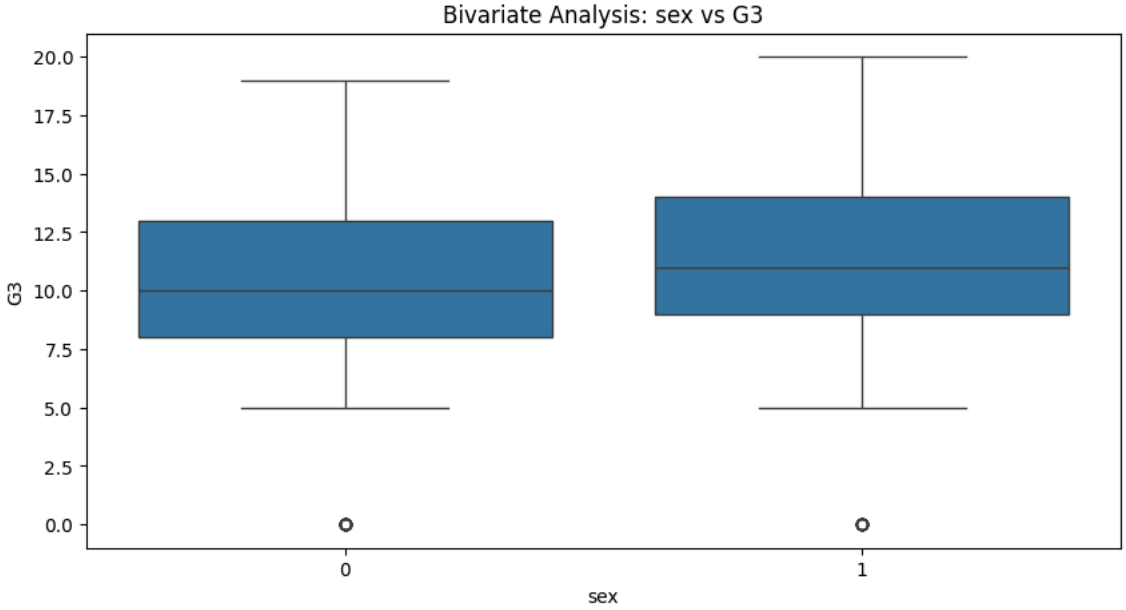

Bivariate Analysis: sex vs health
health
sex
0
1
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

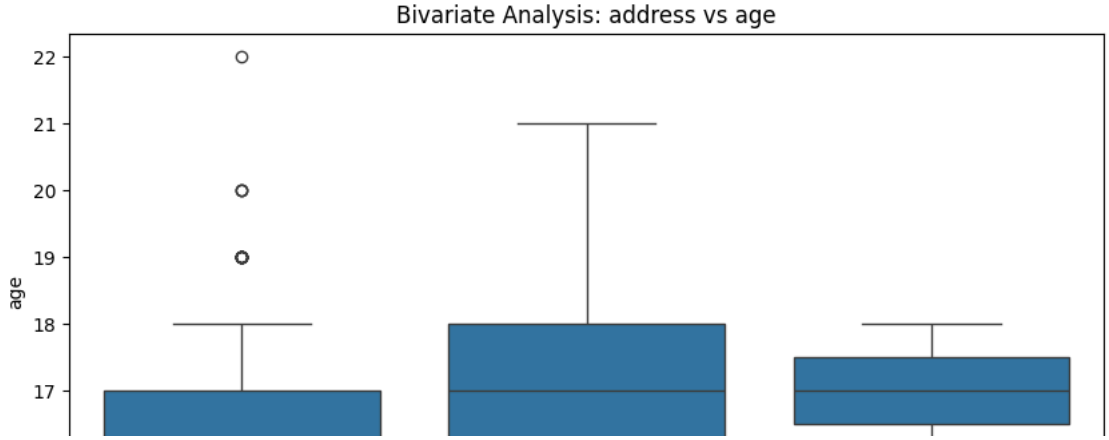

Bivariate Analysis: sex vs absences
absences
sex
0
1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

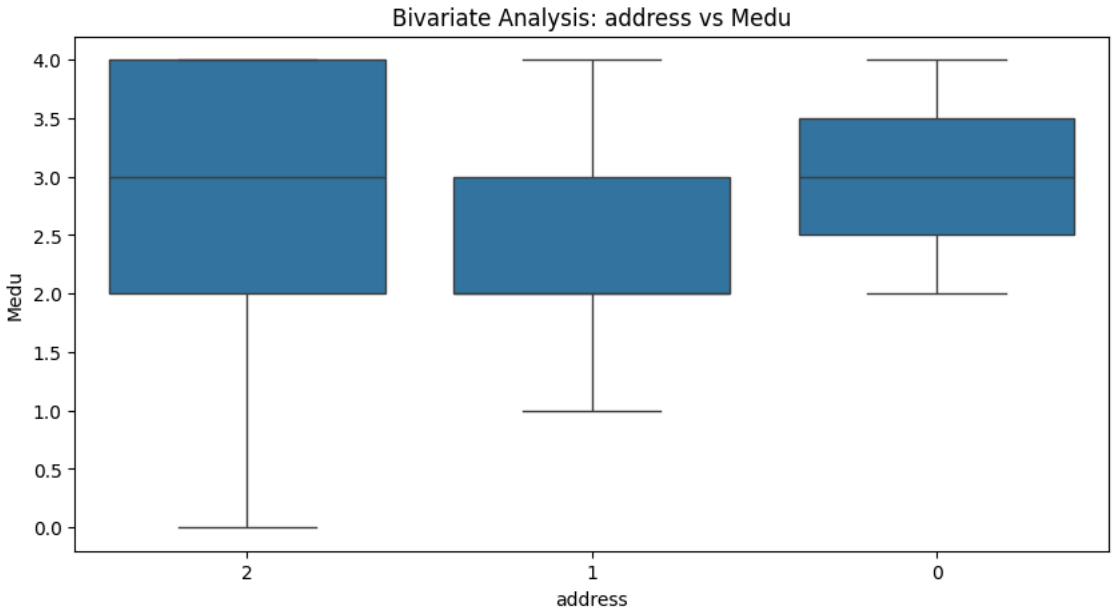

Bivariate Analysis: sex vs G1
14
16
18

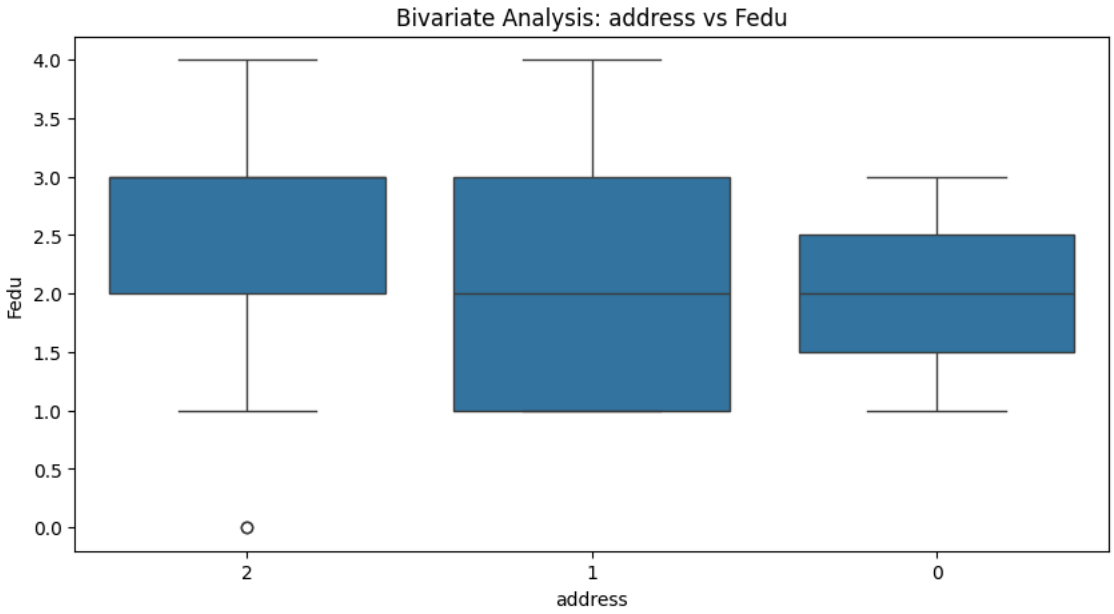

G1
12
10
8
6
4
0
1
sex

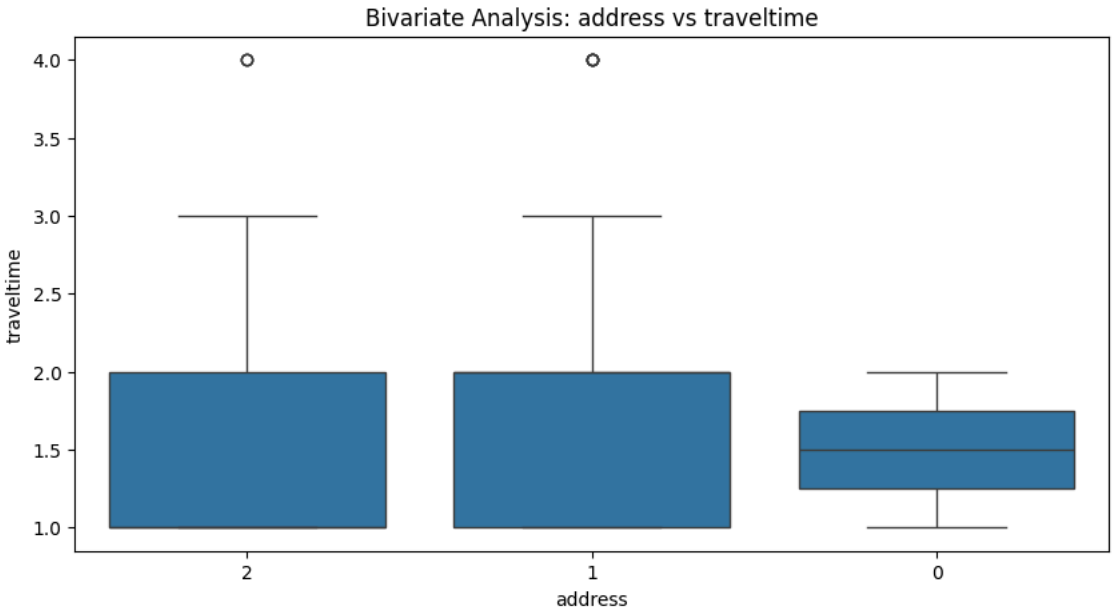

Bivariate Analysis: sex vs G2
G2
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
sex


Bivariate Analysis: sex vs G3
G3
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
sex

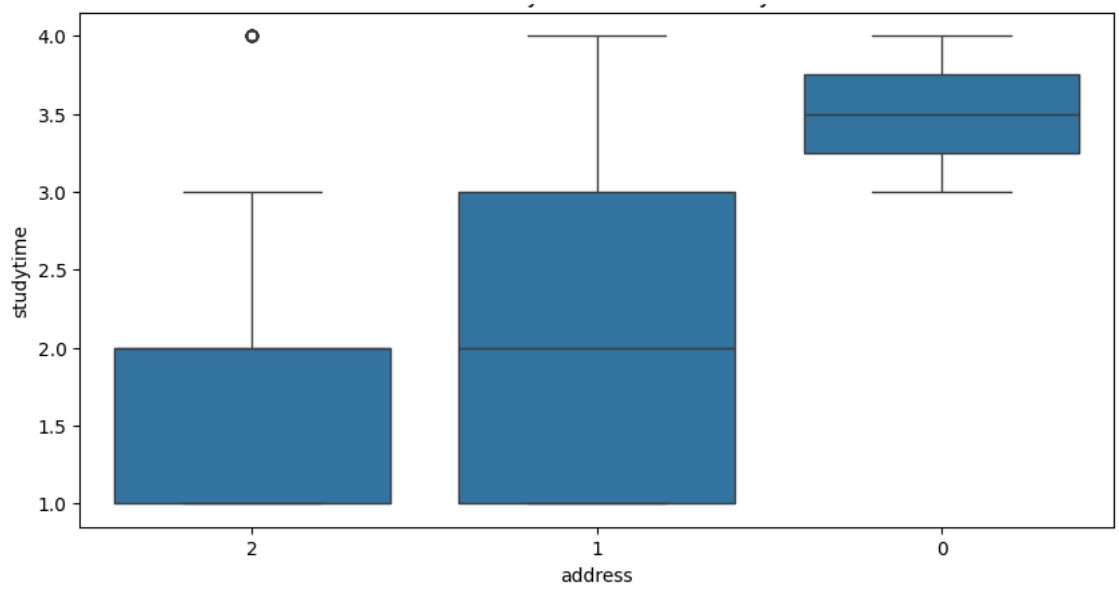

Bivariate Analysis: address vs age
age
22
21
20
19
18
17

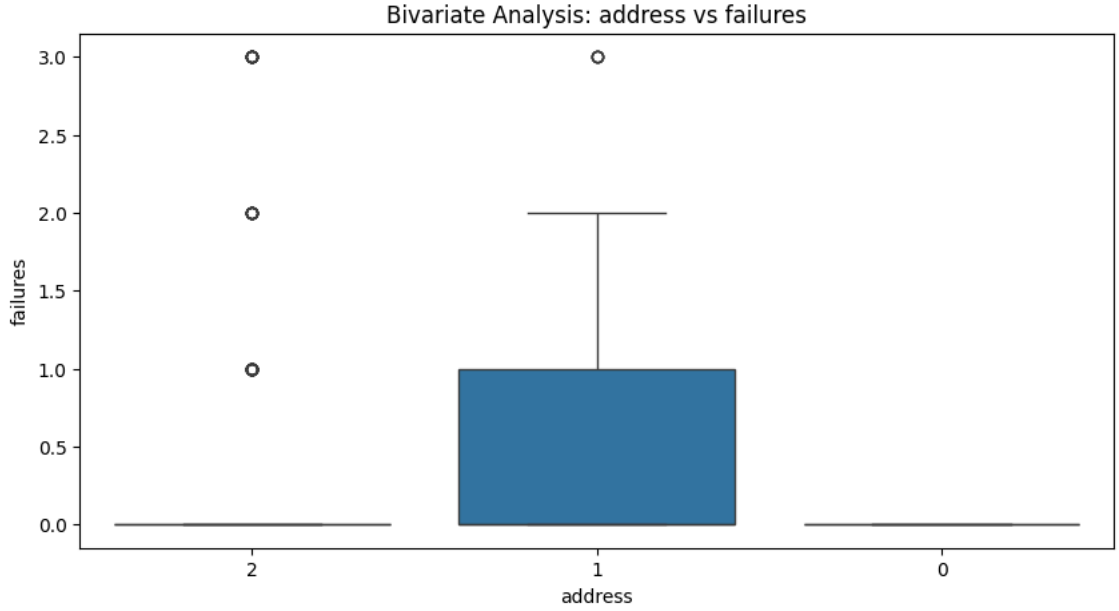


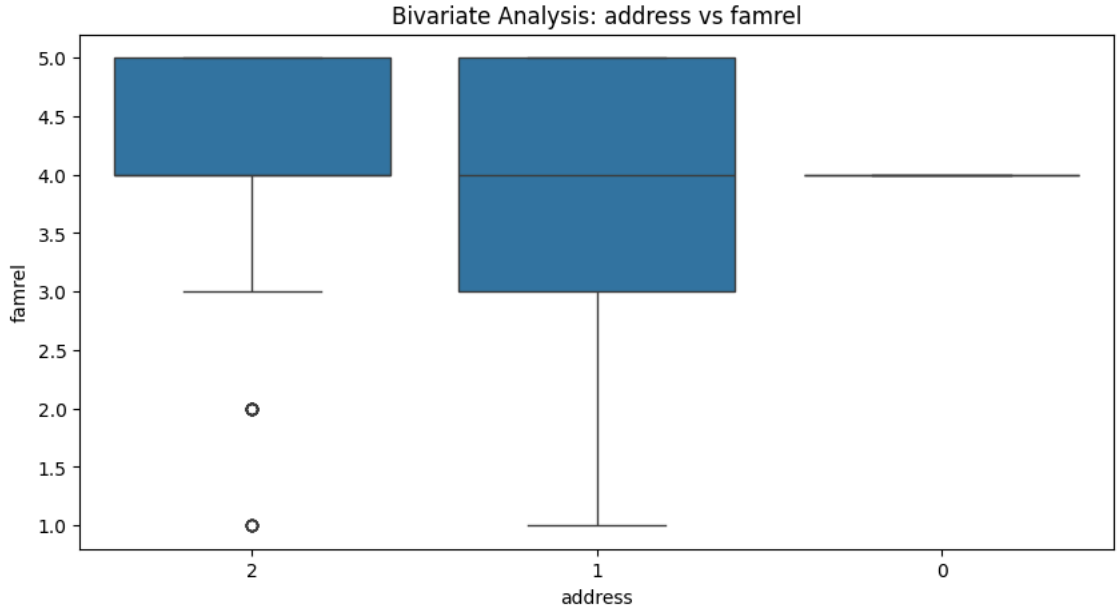


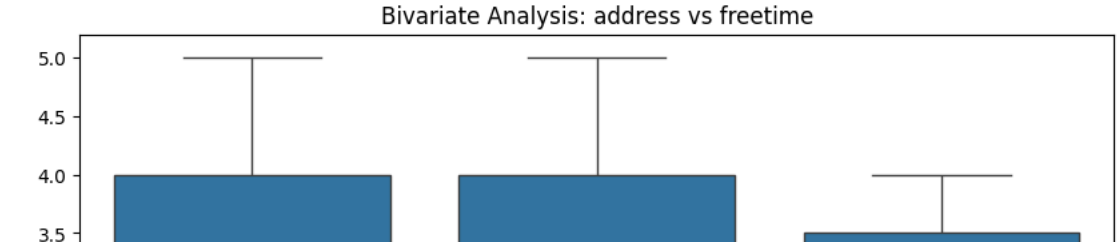


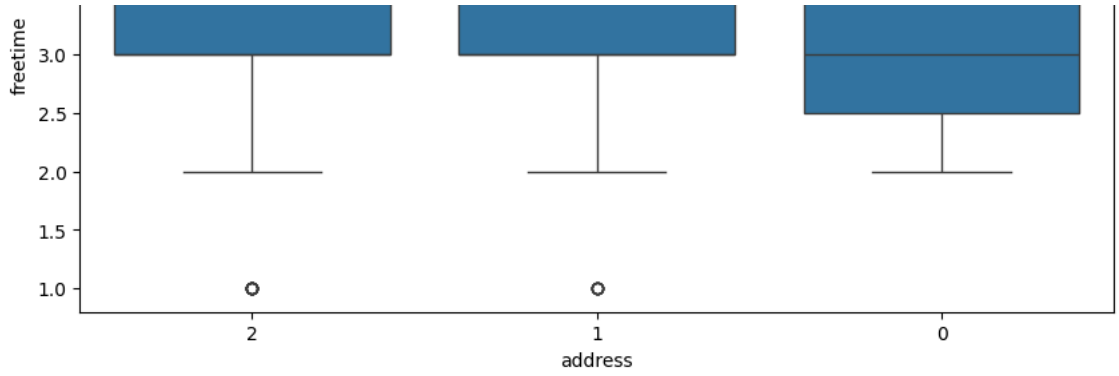
Bivariate Analysis: address vs studytime

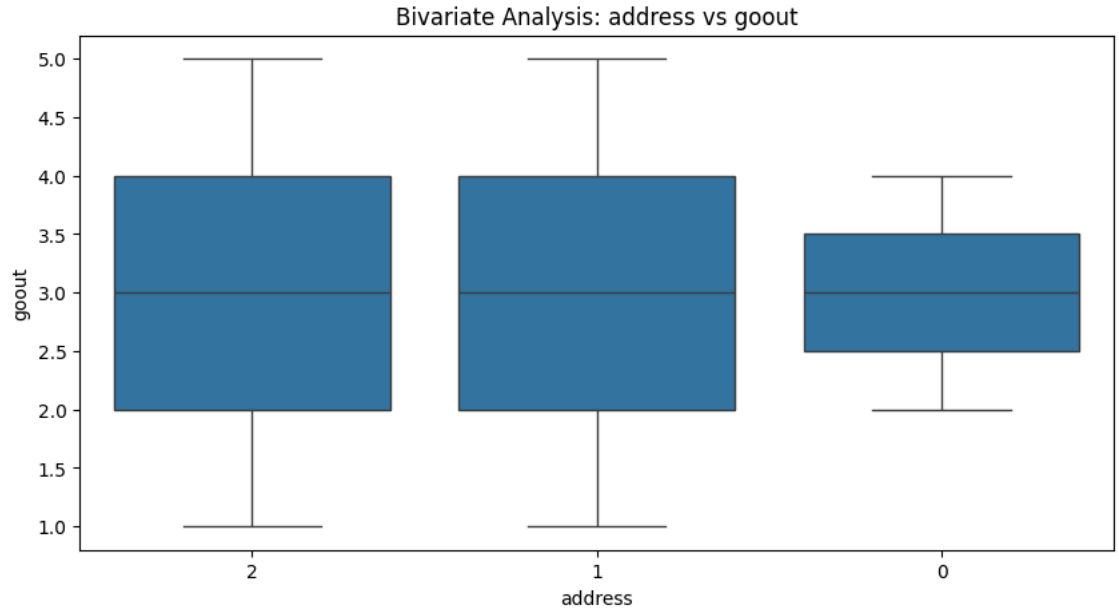

studytime
address
2
1
0

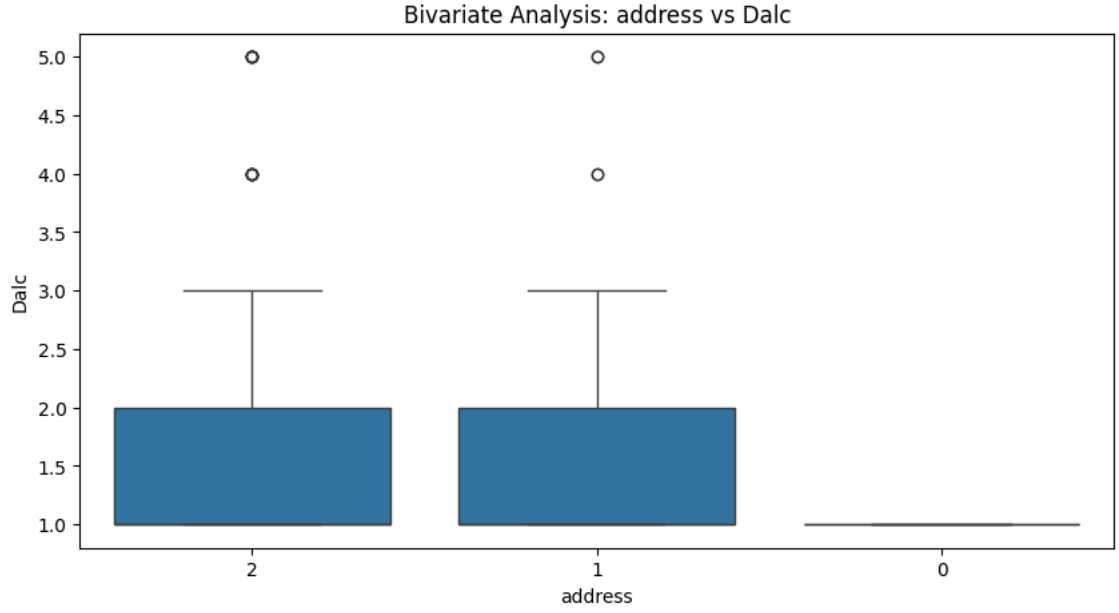

Bivariate Analysis: address vs failures
failures
address

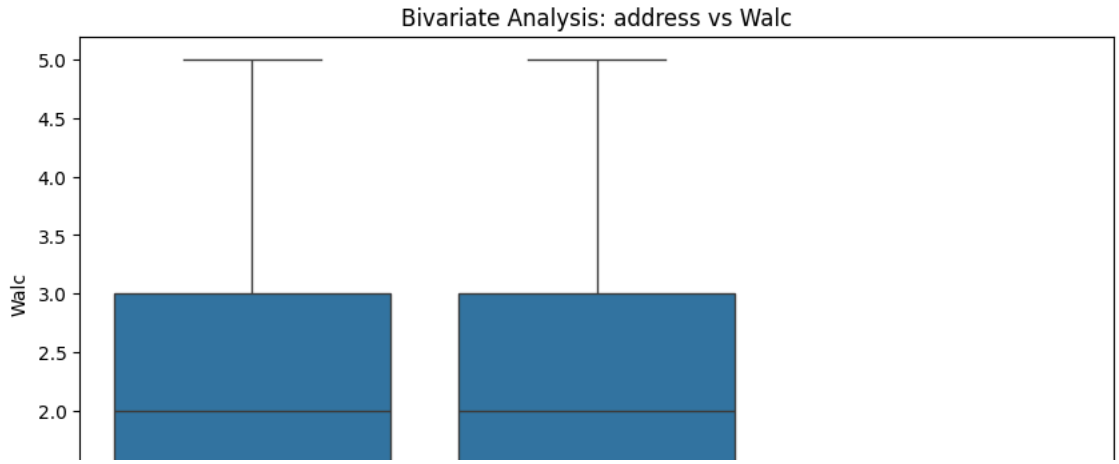

Bivariate Analysis: address vs famrel
famrel
address


Bivariate Analysis: address vs freetime

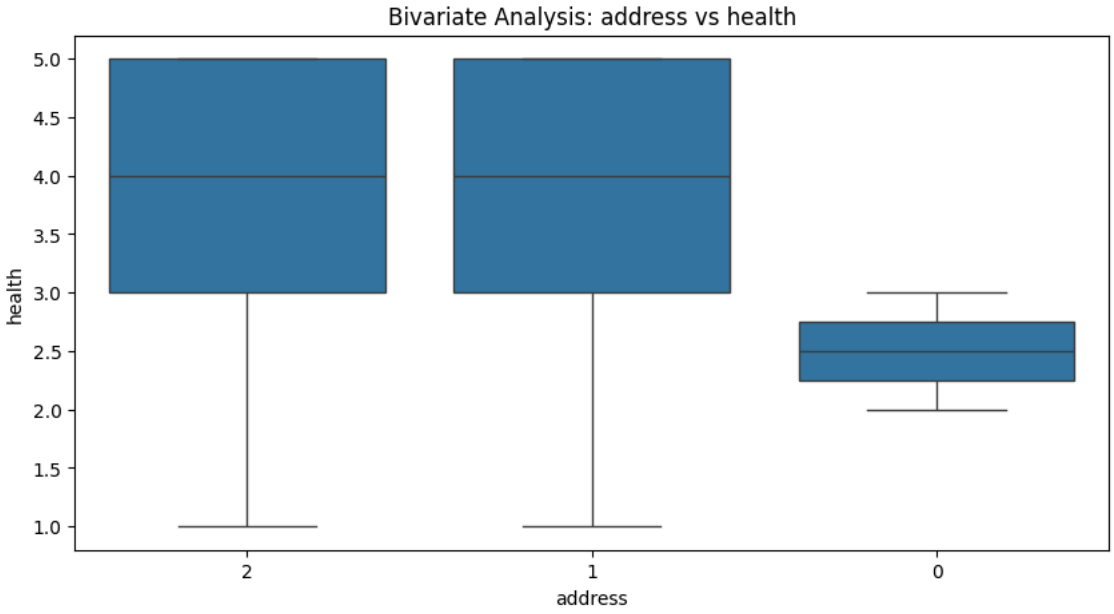

freetime
3.0
2.5
2.0
1.5
1.0
2
1
0
address

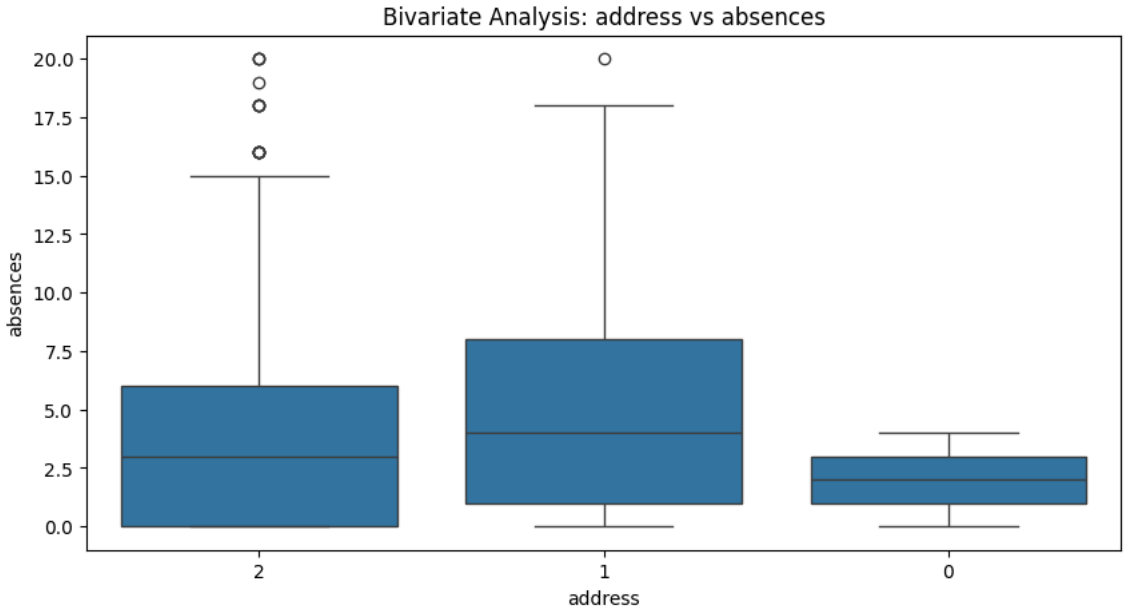

Bivariate Analysis: address vs goout
goout
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
2
1
0
address

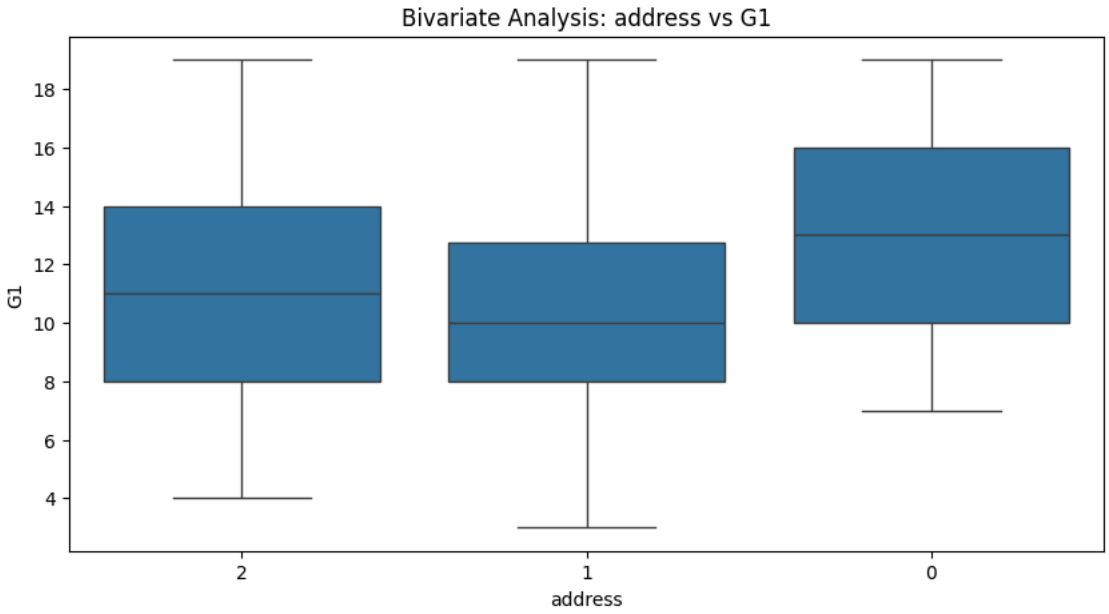

Bivariate Analysis: address vs Dalc
Dalc
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
2
1
0
address

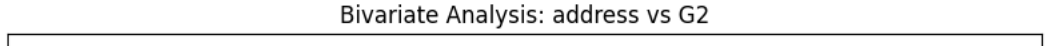

Bivariate Analysis: address vs Walc
Walc
5.0
4.5
4.0
3.5
3.0
2.5
2.0

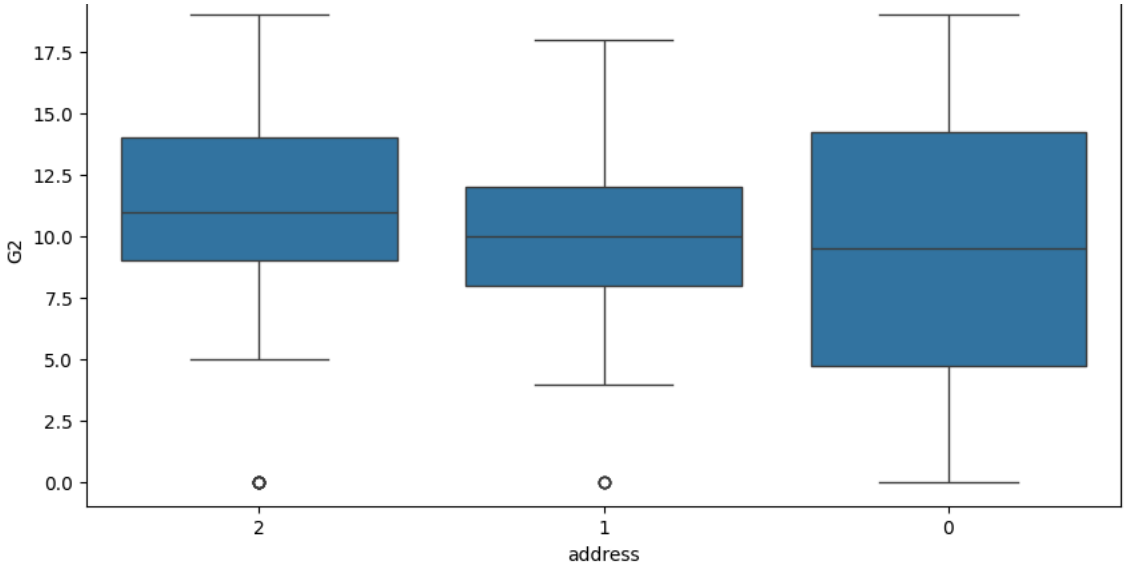

1.5
1.0
2
1
0
address

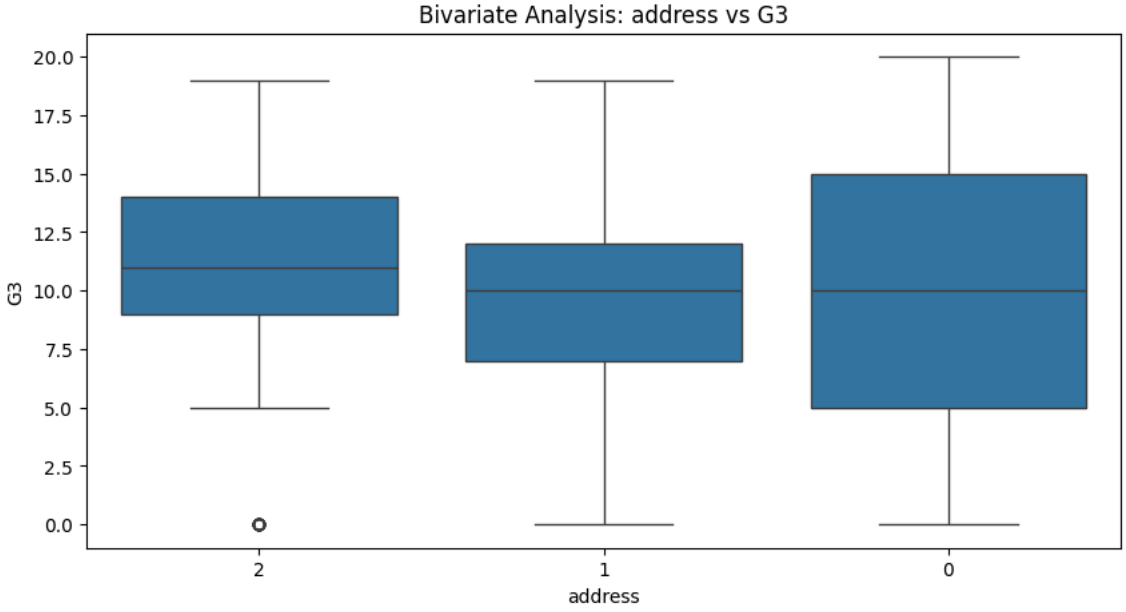

Bivariate Analysis: address vs health
health
address

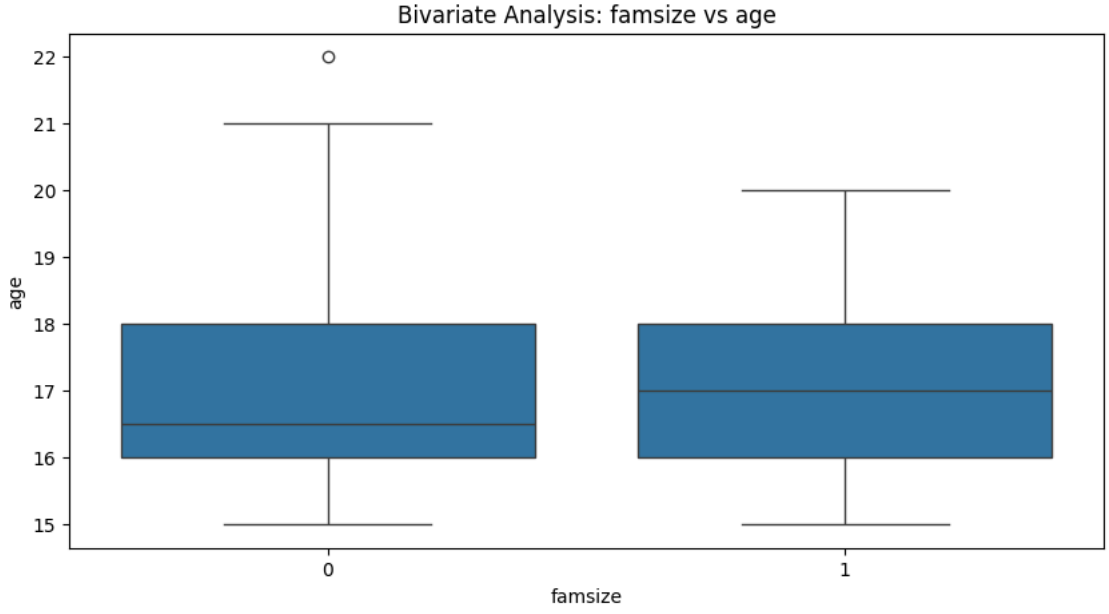

Bivariate Analysis: address vs absences
absences
address

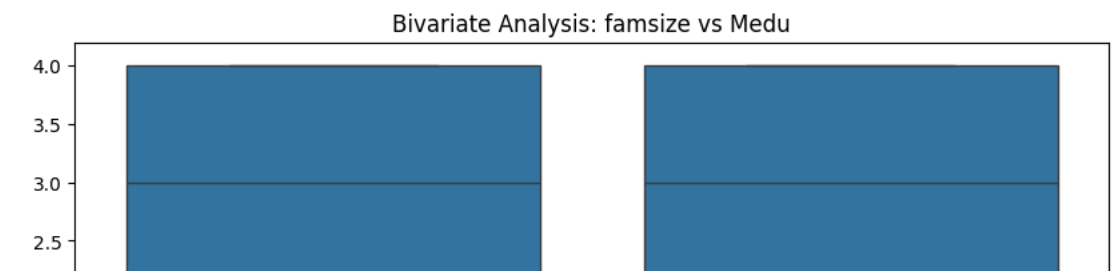

Bivariate Analysis: address vs G1
G1
address

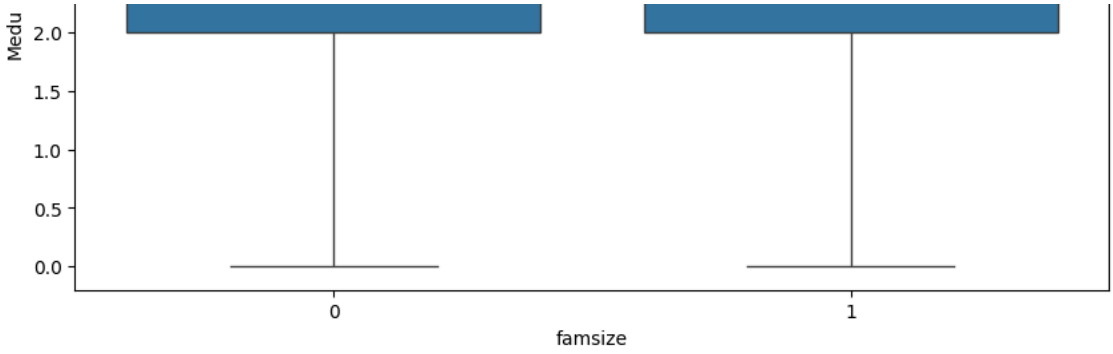

Bivariate Analysis: address vs G2

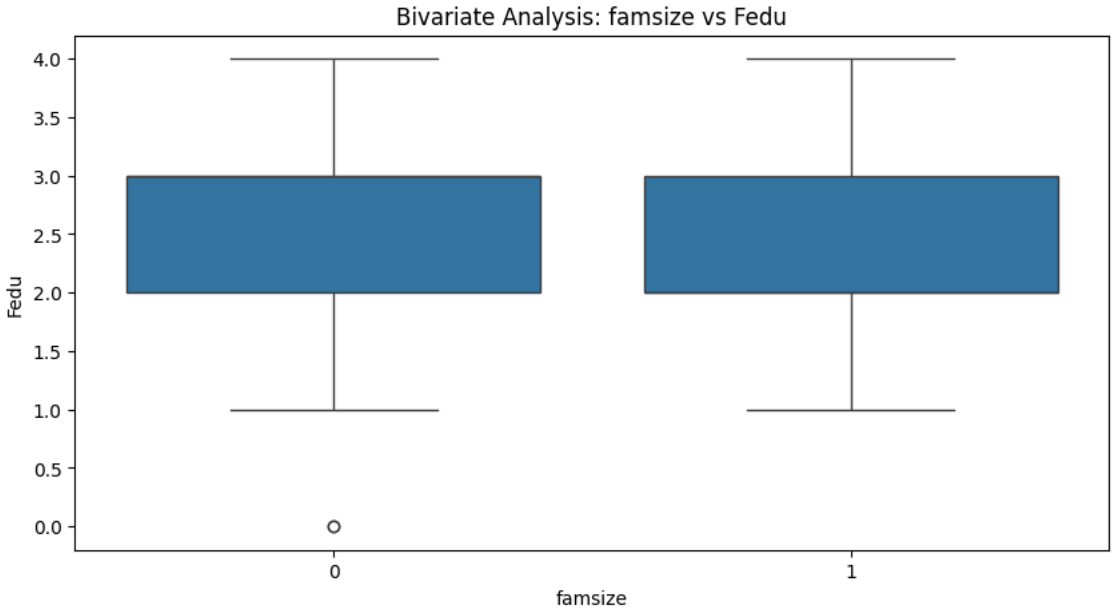

G2
address
2
1
0

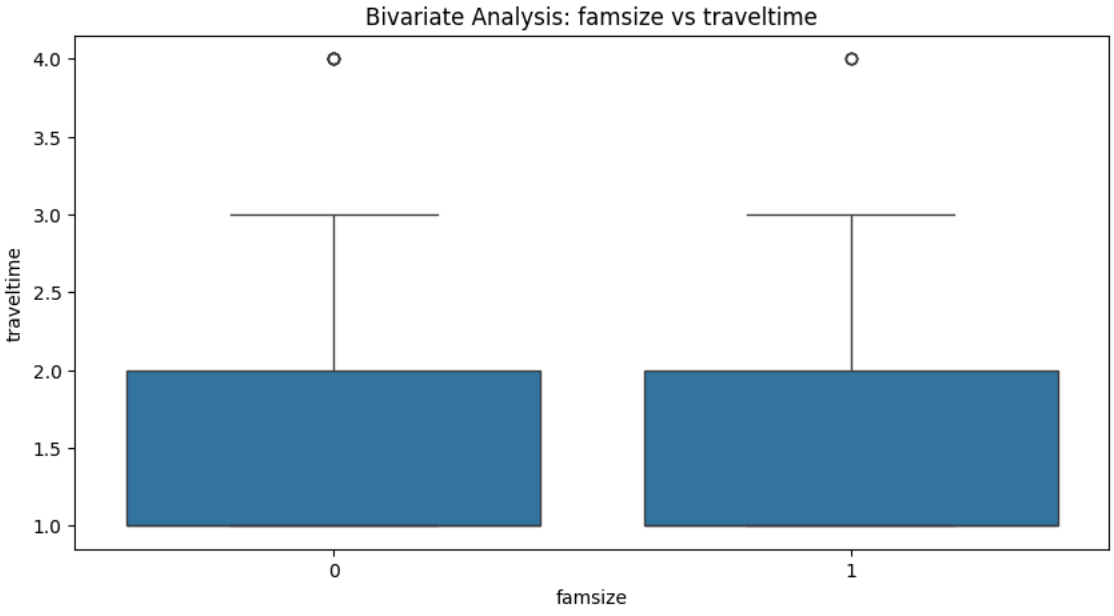

Bivariate Analysis: address vs G3
G3
address
2
1
0

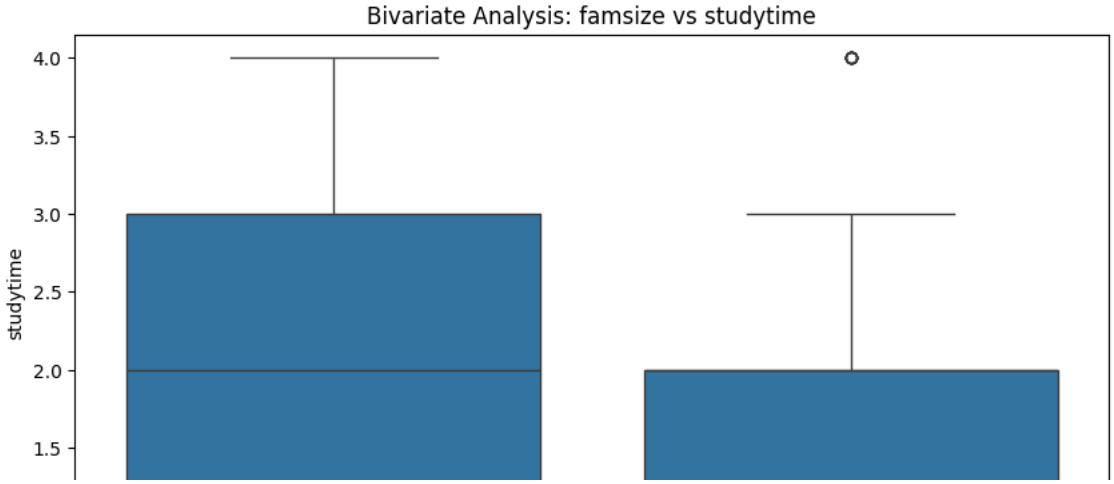

Bivariate Analysis: famsize vs age
age
famsize
0
1

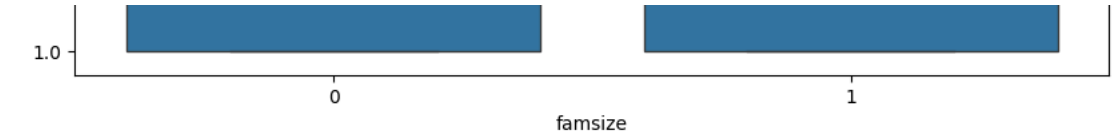

Bivariate Analysis: famsize vs Medu

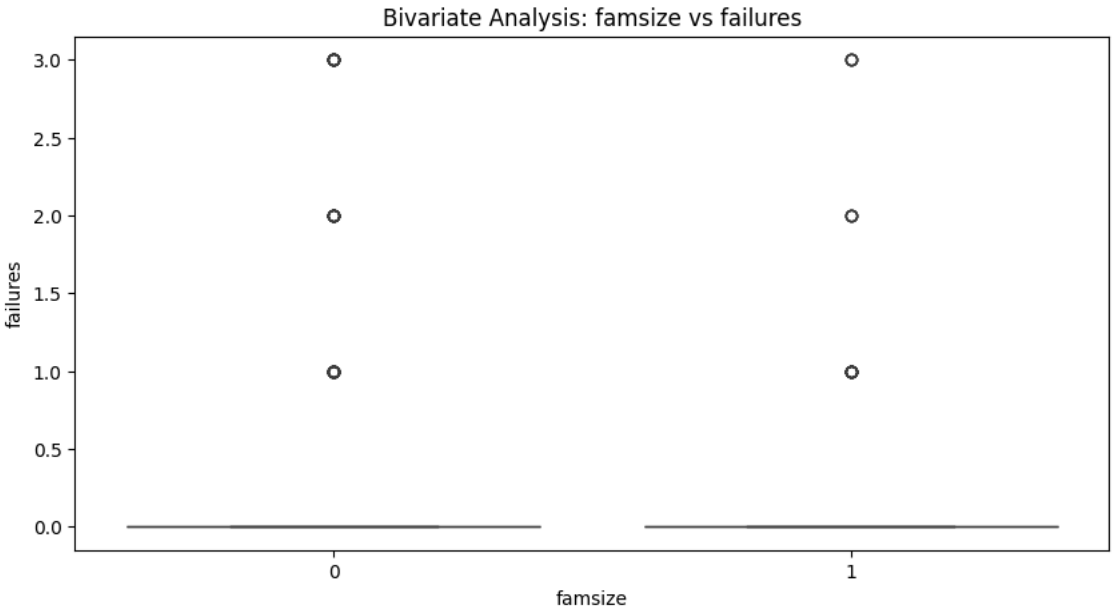

Medu
2.0
1.5
1.0
0.5
0.0
0
1
famsize

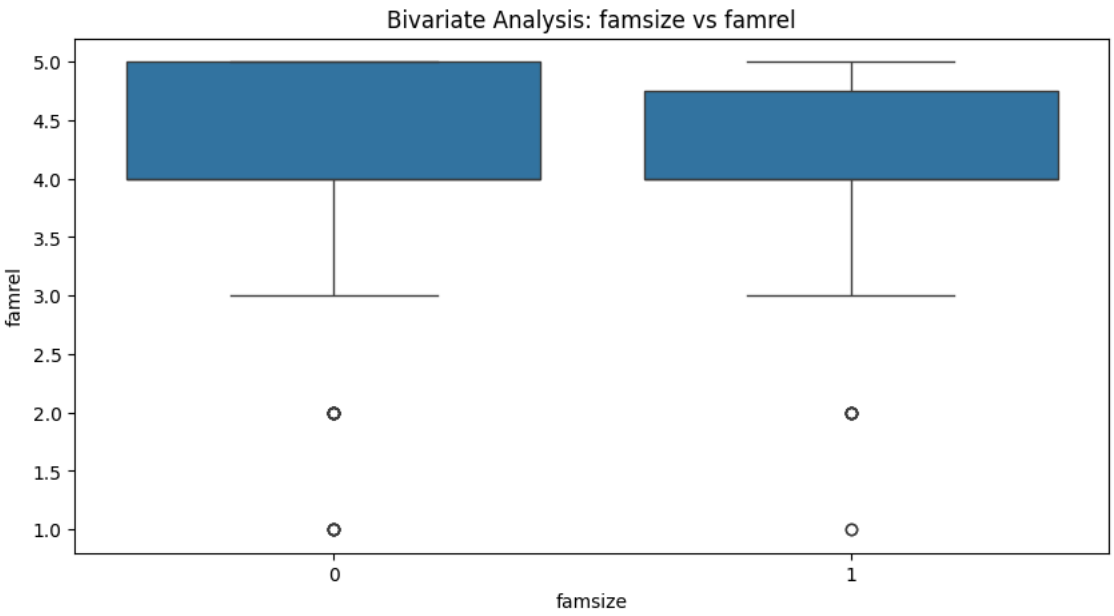

Bivariate Analysis: famsize vs Fedu
Fedu
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
famsize

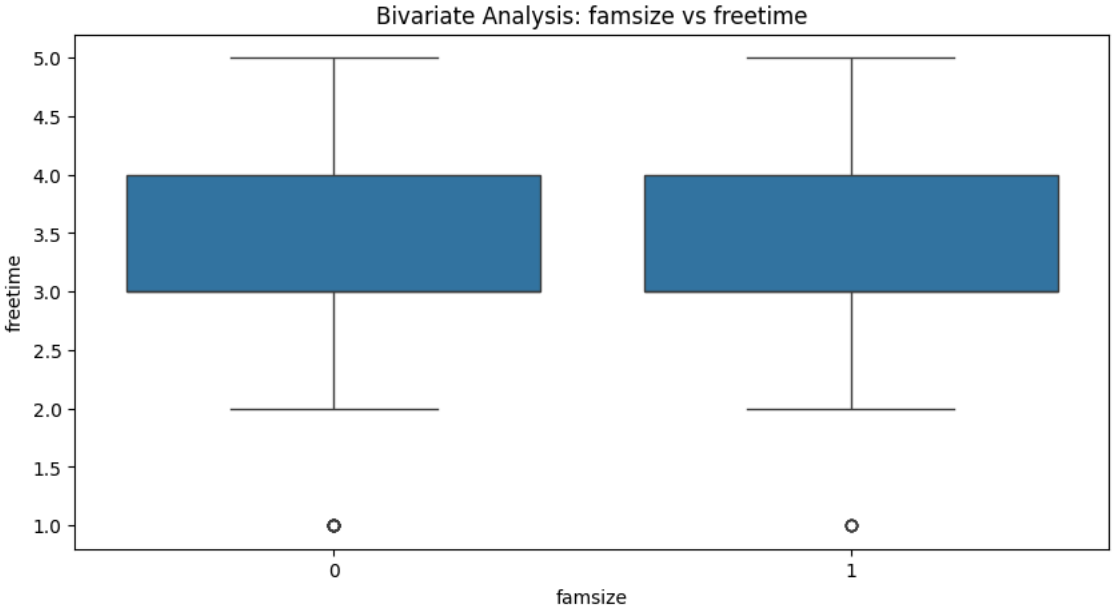

Bivariate Analysis: famsize vs traveltime
traveltime
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
famsize

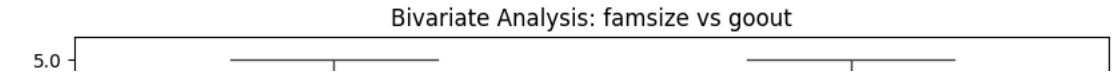

Bivariate Analysis: famsize vs studytime
studytime
4.0
3.5
3.0
2.5
2.0
1.5

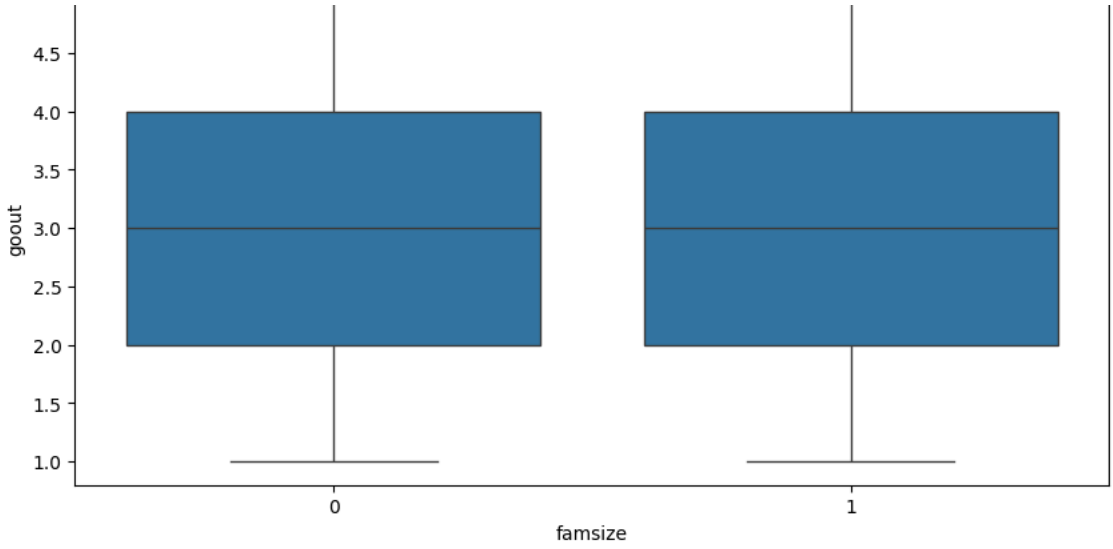

1.0
0
1
famsize


Bivariate Analysis: famsize vs failures
failures
famsize

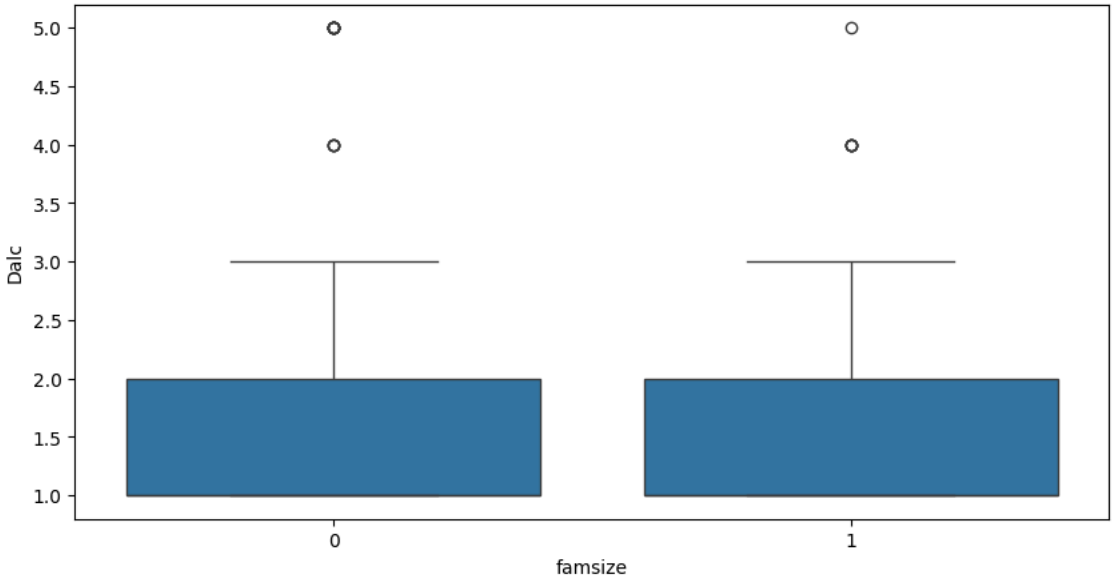

Bivariate Analysis: famsize vs famrel
famrel
famsize


Bivariate Analysis: famsize vs freetime
freetime
famsize

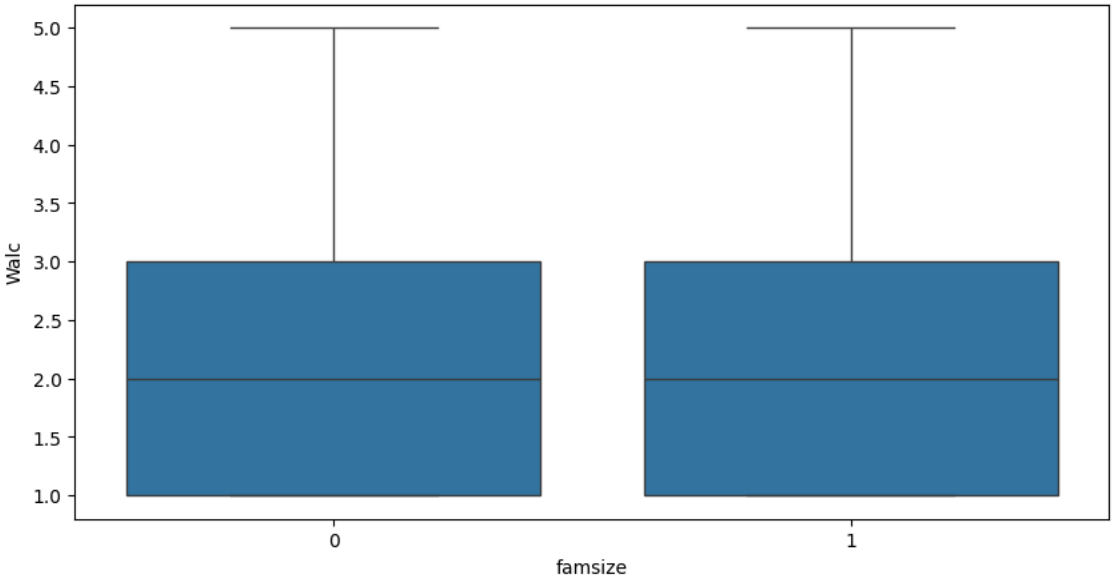

Bivariate Analysis: famsize vs goout
5.0



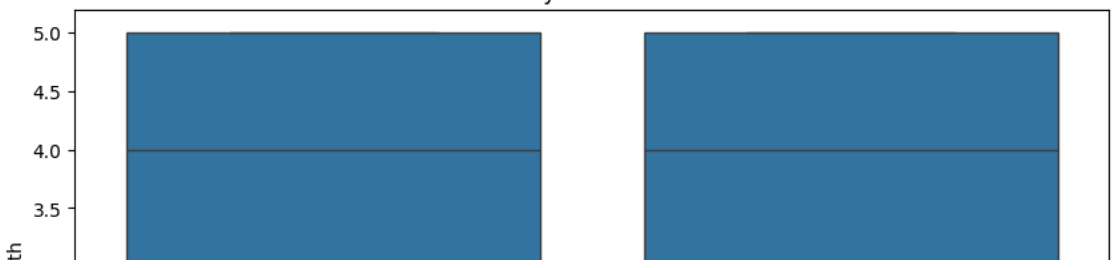
Bivariate Analysis: famsize vs Dalc

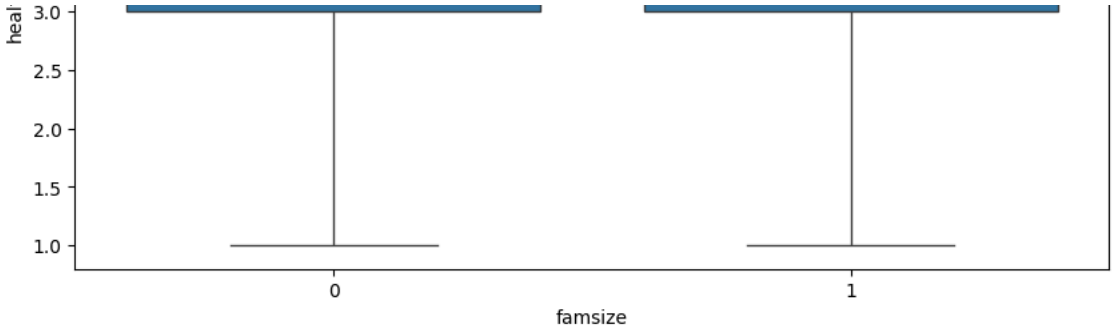


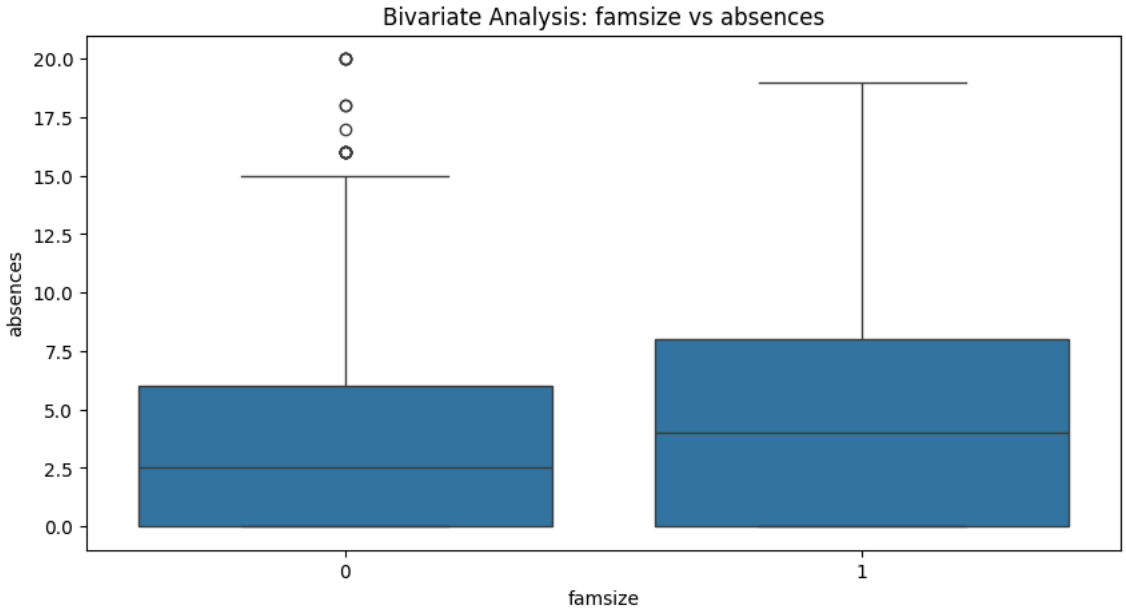
Bivariate Analysis: famsize vs Walc

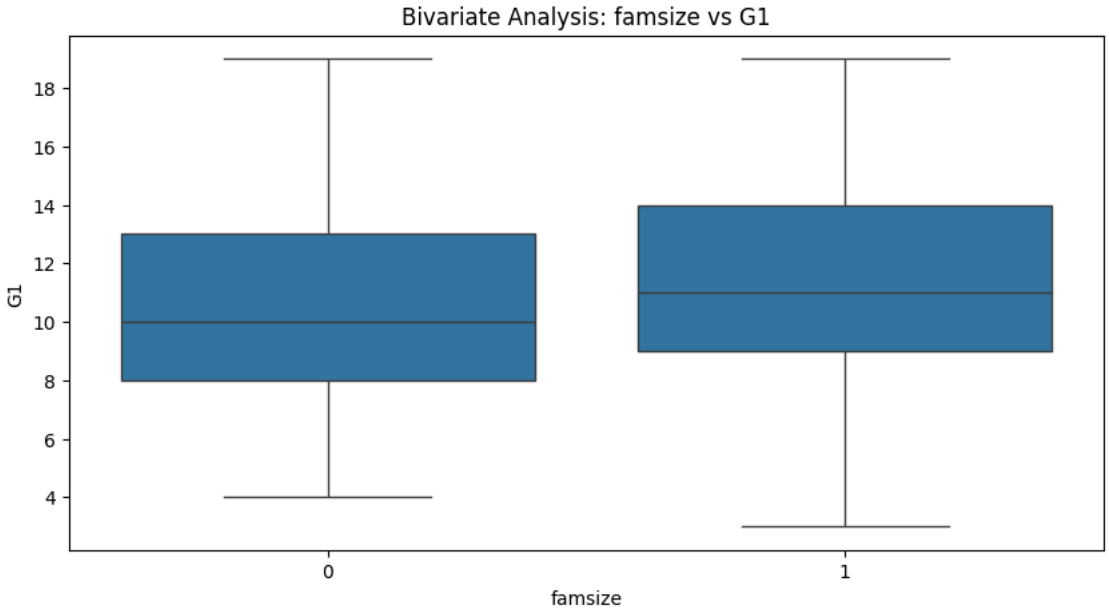


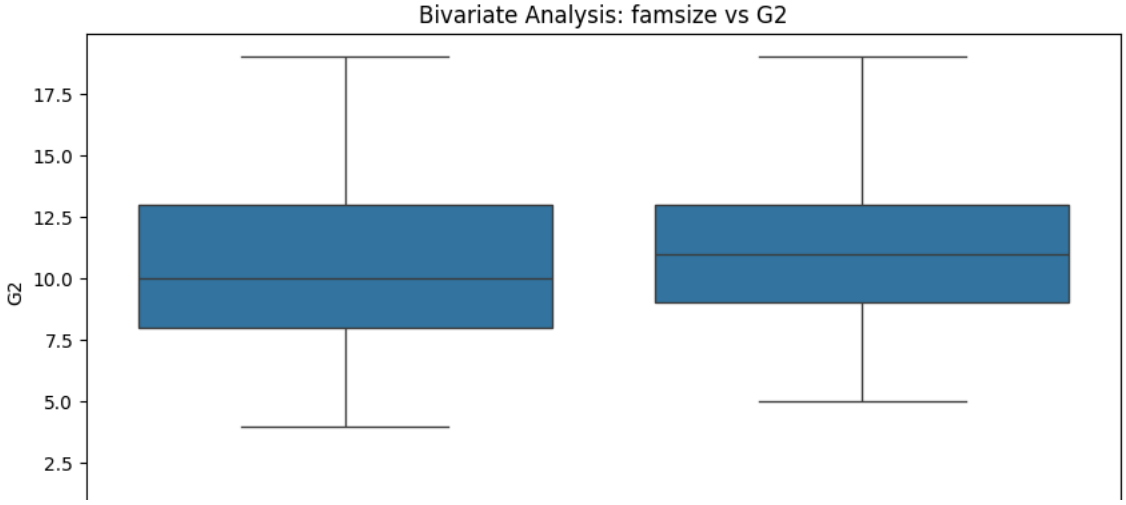
Bivariate Analysis: famsize vs health



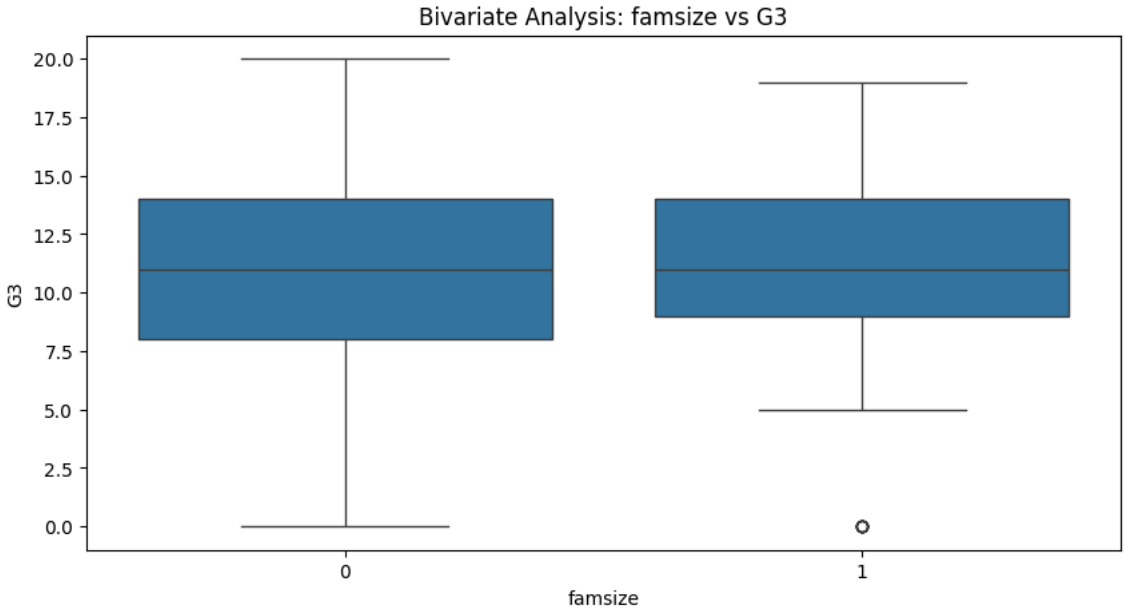

heal
3.0
2.5
2.0
1.5
1.0
0
1
famsize

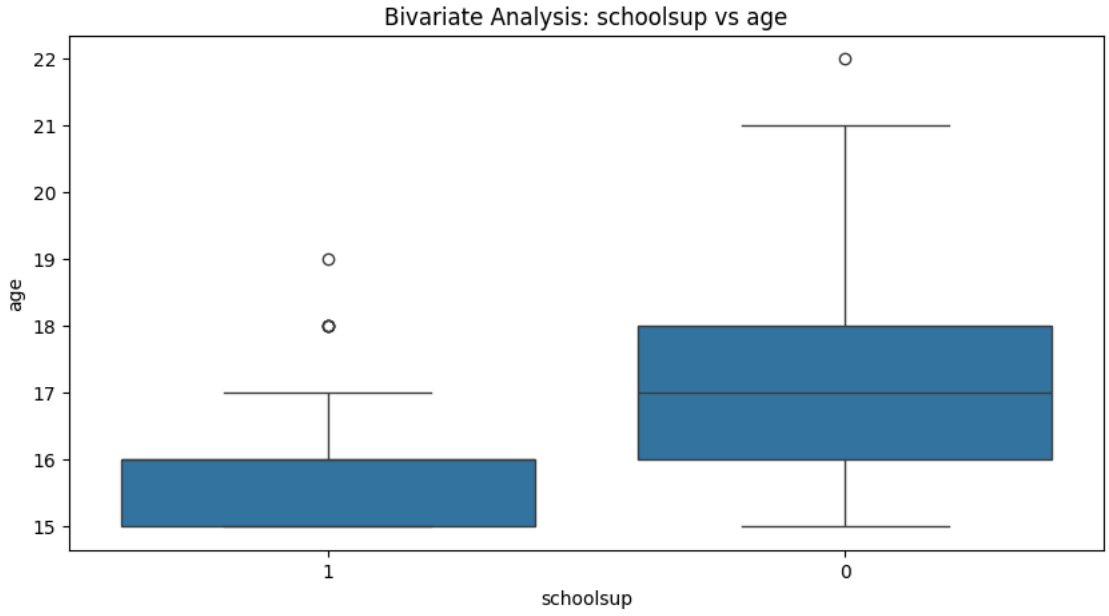

Bivariate Analysis: famsize vs absences
absences
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
famsize

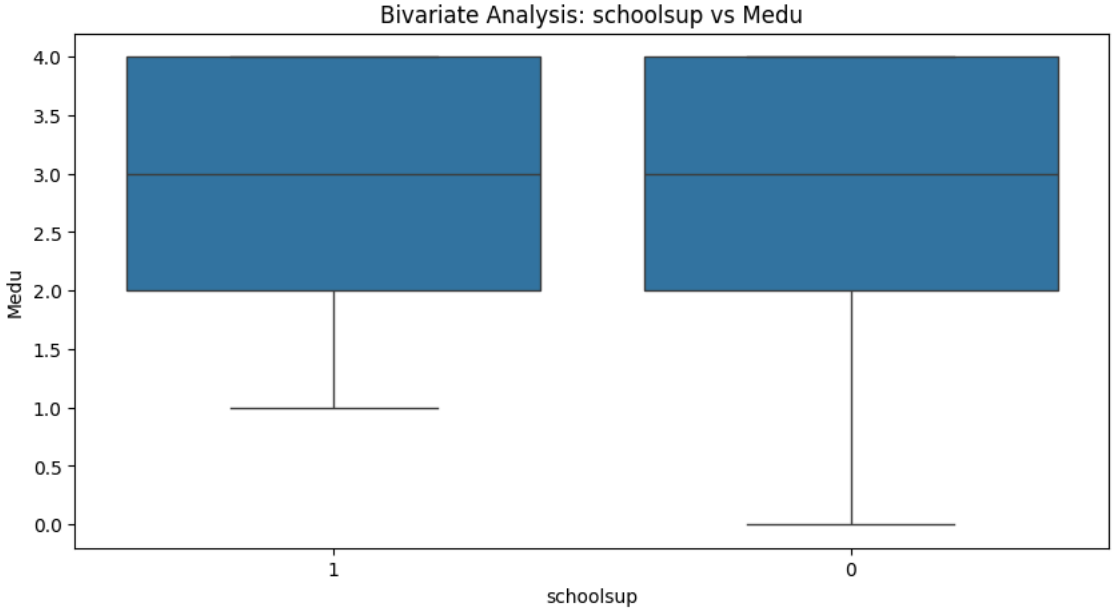

Bivariate Analysis: famsize vs G1
G1
18
16
14
12
10
8
6
4
0
1
famsize

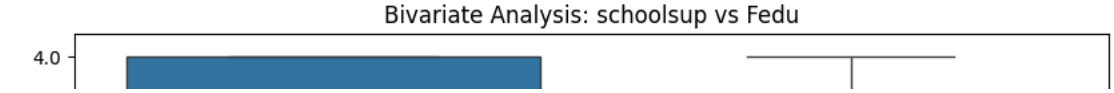

Bivariate Analysis: famsize vs G2
G2
17.5
15.0
12.5
10.0
7.5
5.0
2.5

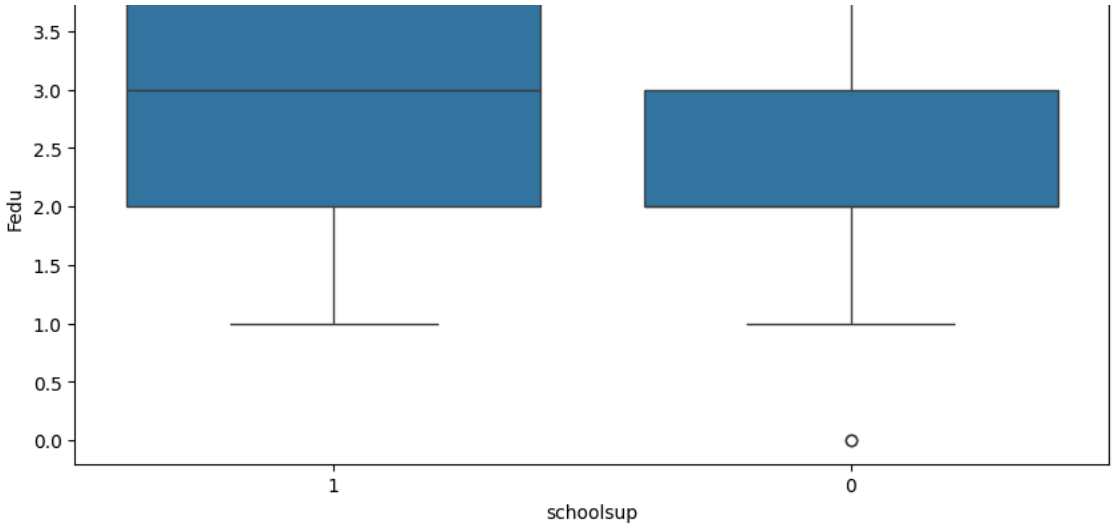

0.0
0
1
famsize

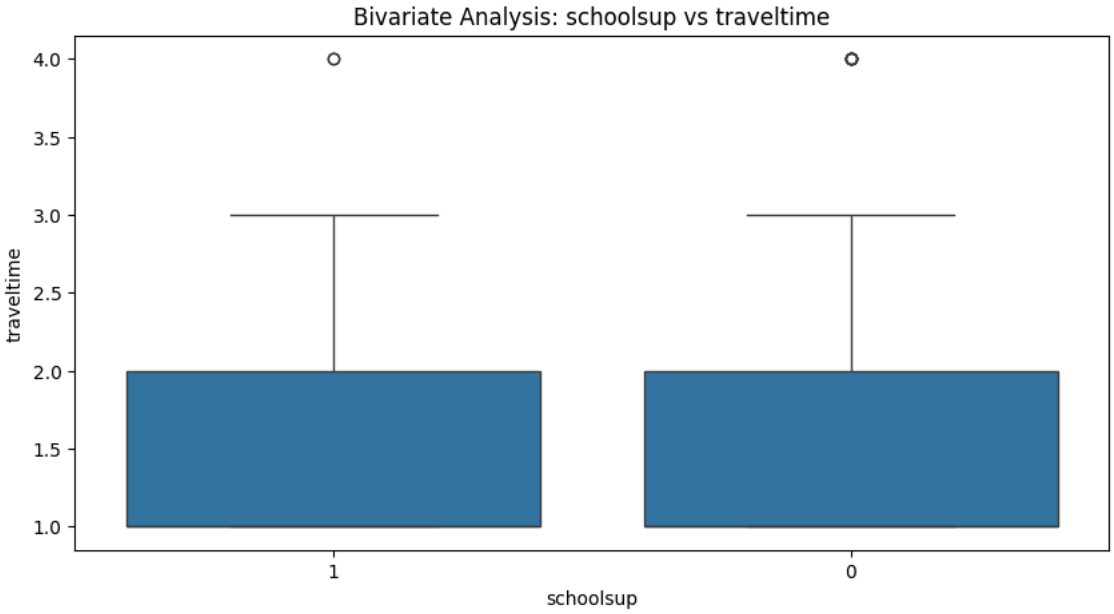

Bivariate Analysis: famsize vs G3
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
G3
0
1
famsize

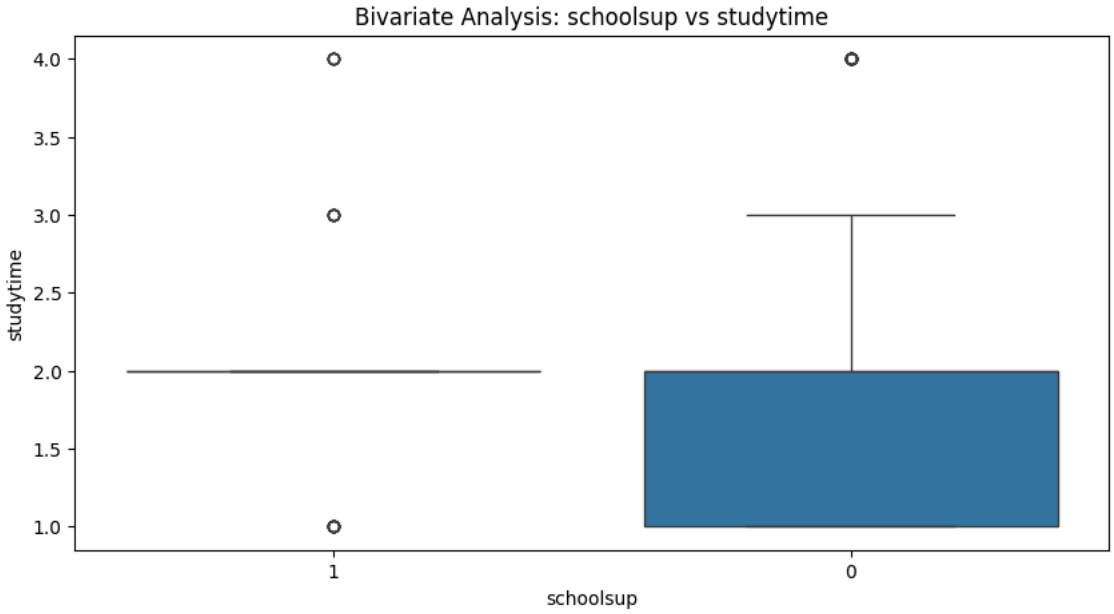

Bivariate Analysis: schoolsup vs age
22
21
20
19
18
17
16
15
age
1
0
schoolsup

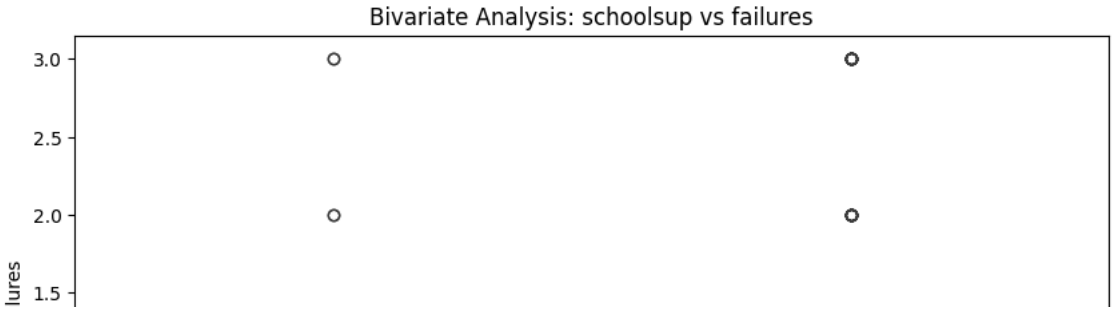

Bivariate Analysis: schoolsup vs Medu
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Medu
1
0
schoolsup

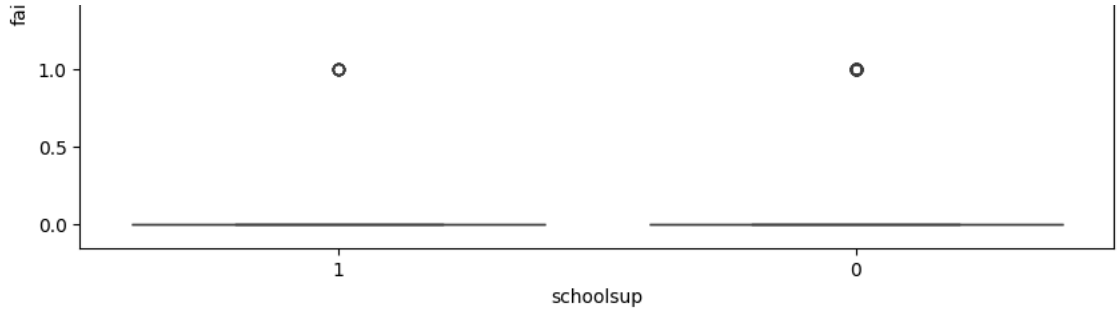

Bivariate Analysis: schoolsup vs Fedu
4.0

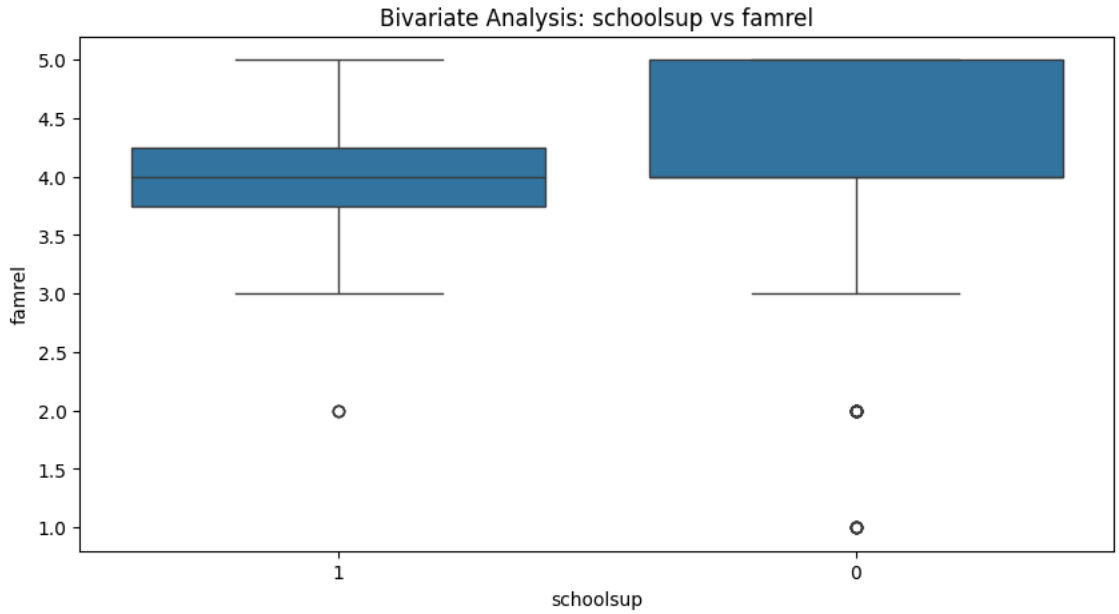

Fedu
schoolsup
1
0

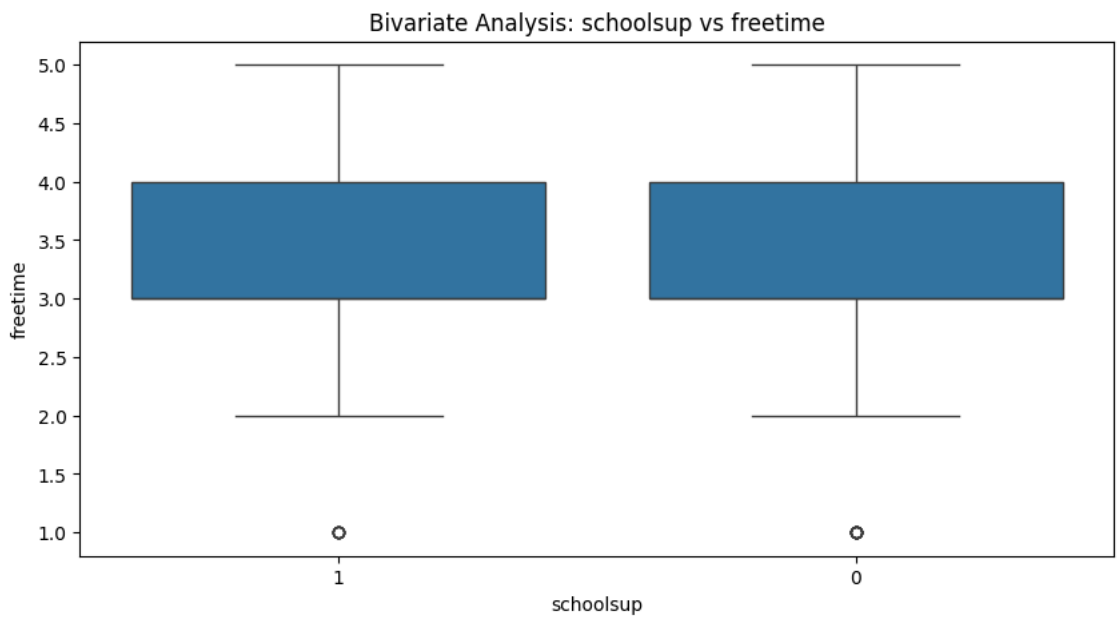

Bivariate Analysis: schoolsup vs traveltime
traveltime
schoolsup
1
0

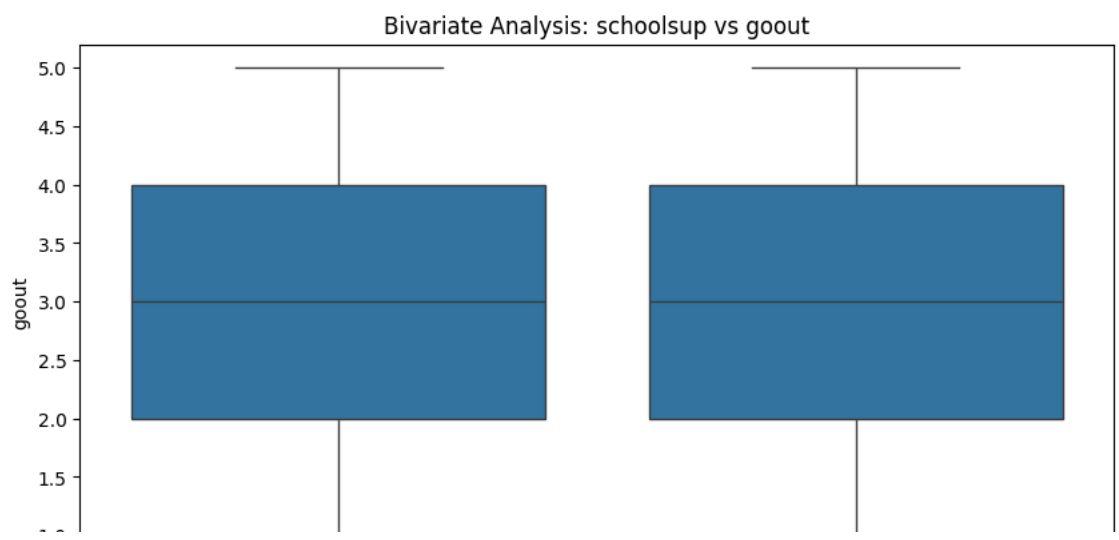

Bivariate Analysis: schoolsup vs studytime
studytime
schoolsup
1
0


Bivariate Analysis: schoolsup vs failures

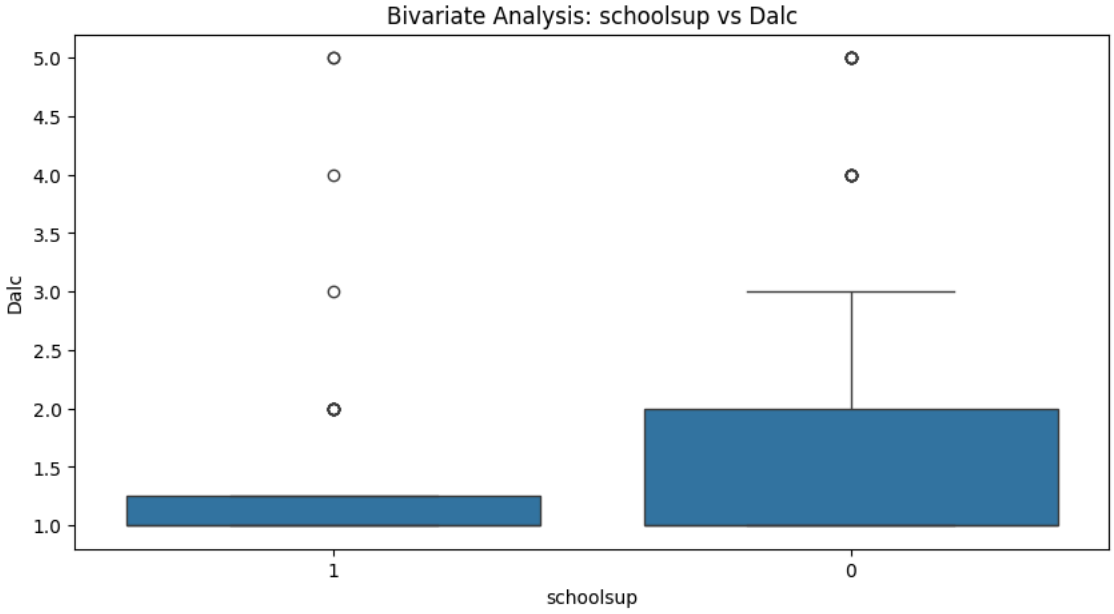

fai
1.0
0.5
0.0
1
0
schoolsup

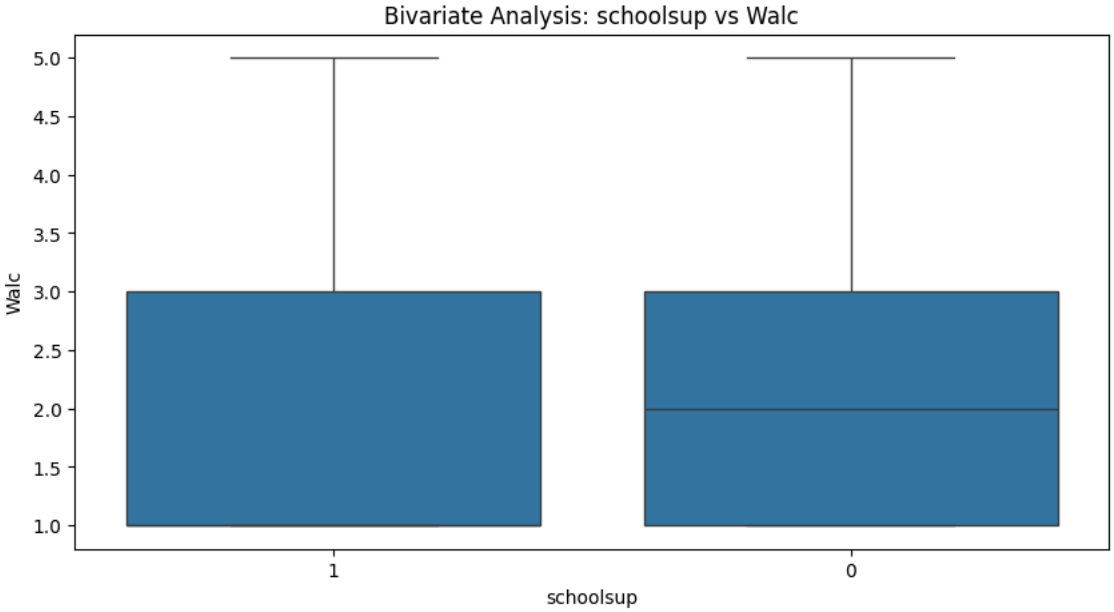

Bivariate Analysis: schoolsup vs famrel
famrel
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
0
schoolsup

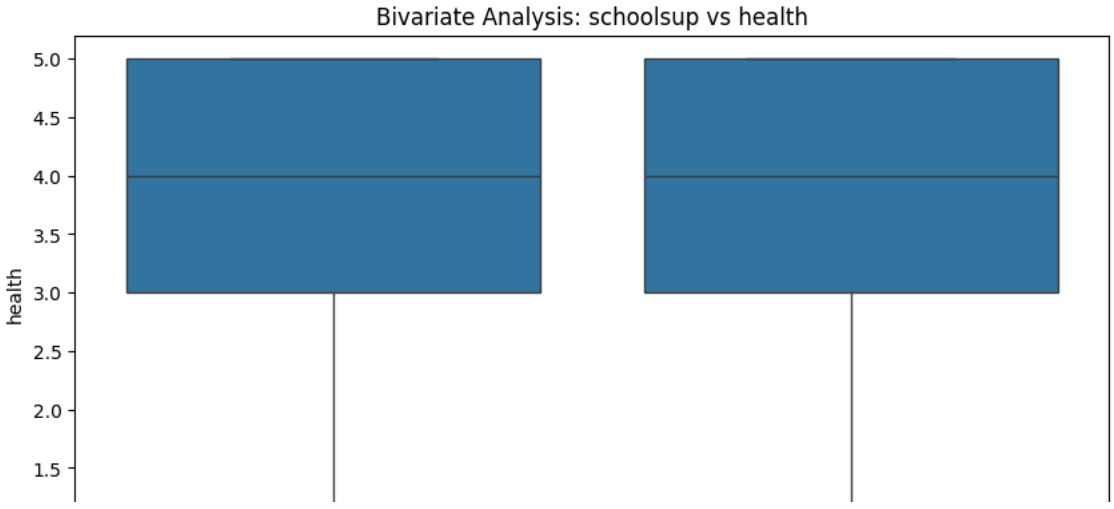

Bivariate Analysis: schoolsup vs freetime
freetime
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
0
schoolsup


Bivariate Analysis: schoolsup vs goout
goout
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5


1.0
1
0
schoolsup


Bivariate Analysis: schoolsup vs Dalc
Dalc
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
0
schoolsup


Bivariate Analysis: schoolsup vs Walc
Walc
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
0
schoolsup


Bivariate Analysis: schoolsup vs health
health
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5