project1

May 14, 2025

Project - *"Predicting Student Academic Performance Using Behavioral and Demographic Data"*

Data Card: Student Performance Prediction

Problem Statement

To predict students' final grade (G3) based on various academic, personal, and family-related features. The goal is to identify key factors influencing academic performance and build a model to assist educators in early intervention.

Dataset Overview

Name: Student Performance Dataset

Source: UCI Machine Learning Repository

Rows: 395 students

Columns: 33 (categorical + numerical)

Target Variable: G3 (Final Grade)

Features include: demographic info, school background, parental education, study time, absences, etc.

Step-by-Step Analysis

1. **Data Loading and Cleaning** Loaded data using pandas with sep=';'.

Checked for null values and data types.

2. **Exploratory Data Analysis (EDA)**

Univariate Analysis: Histograms for numeric columns, count plots for categorical ones.

Bivariate Analysis: Compared each feature with G3 using:

Boxplots (for categorical vs G3)

Regression plots (for numerical vs G3)

3. **Feature Engineering** Encoded categorical variables using LabelEncoder.

Analyzed correlation between variables using a heatmap.

4. **Feature Selection** Applied SelectKBest with f_regression to identify top features affecting final grades.

Selected best-performing features for model input.

5. **Model Building** Split data into training and test sets.

Algorithm used to create a model:

Linear Regression

Random Forest Regressor

Support Vector Regressor (SVR)

Loading libraries and Data

[8]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Suppress future warnings
import warnings
warnings.filterwarnings('ignore')
```

[9]:
```
# read data set
data = pd.read_csv(r"C:\Users\hp\Downloads\student-mat.csv", sep=',')

# see top 5 rows
data.head()
```

[9]:
```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  …  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  …
1     GP   F   17       U     GT3       T     1     1  at_home     other  …
2     GP   F   15       U     LE3       T     1     1  at_home     other  …
3     GP   F   15       U     GT3       T     4     2   health  services  …
4     GP   F   16       U     GT3       T     3     3    other     other  …

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4        3      4     1     1      3        6   5   6   6
1      5        3      3     1     1      3        4   5   5   6
2      4        3      2     2     3      3       10   7   8  10
3      3        2      2     1     1      5        2  15  14  15
4      4        3      2     1     2      5        4   6  10  10

[5 rows x 33 columns]
```

Column Disclosure

[ ]:
```
age         -     # Student age
Medu        -     # Mother's education (0-4)
Fedu        -     # Father's education (0-4)
traveltime  -     # Travel time to school (1-4)
studytime   -     # Weekly study time (1-4)
failures    -     # Number of past class failures
famrel      -     # Family relationship quality (1-5)
freetime    -     # Free time after school (1-5)
goout       -     # Going out with friends (1-5)
Dalc        -     # Workday alcohol consumption (1-5)
Walc        -     # Weekend alcohol consumption (1-5)
health      -     # Current health status (1-5)
absences    -     # Number of school absences
G1          -     # First period grade (0-20)
G2          -     # Second period grade (0-20)
G3          -     # Final grade (target variable)
school      -     # School name (GP/MS)
sex         -     # Gender (M/F)
address     -     # Urban/rural (U/R)
famsize     -     # Family size (LE3/GT3)
Pstatus     -     # Parent status (T/A)
Mjob        -     # Mother's job
Fjob        -     # Father's job
reason      -     # Reason to choose this school
guardian    -     # Guardian (mother/father/other)
schoolsup   -     # Extra educational support (yes/no)
famsup      -     # Family educational support (yes/no)
paid        -     # Extra paid classes (yes/no)
activities  -     # Extra-curricular activities (yes/no)
nursery     -     #Attended nursery school (yes/no)
higher      -     # Wants to pursue higher education (yes/no)
internet    -     # Internet access at home (yes/no)
romantic    -     # In a romantic relationship (yes/no)**
```

[7]:
```
# see column data type and some info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
```

```
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

[11]:

```python
# see precentege of missing value in each column

data.isna().sum()
```

[11]:

```
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
```

```
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
```

[13]:
```
data.shape
```

[13]:
```
(395, 33)
```

[15]:
```
# check if duplicated in data

data.duplicated().any()
```

[15]:
```
False
```

[17]:
```
# see quick info of numeric values
data.describe()
```

[17]:
```
              age        Medu        Fedu  traveltime   studytime    failures  \
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000
mean    16.696203    2.749367    2.521519    1.448101    2.035443    0.334177
std      1.276043    1.094735    1.088201    0.697505    0.839240    0.743651
min     15.000000    0.000000    0.000000    1.000000    1.000000    0.000000
25%     16.000000    2.000000    2.000000    1.000000    1.000000    0.000000
50%     17.000000    3.000000    2.000000    1.000000    2.000000    0.000000
75%     18.000000    4.000000    3.000000    2.000000    2.000000    0.000000
max     22.000000    4.000000    4.000000    4.000000    4.000000    3.000000

           famrel    freetime       goout        Dalc        Walc      health  \
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000
```

```
mean     3.944304    3.235443    3.108861    1.481013    2.291139    3.554430
std      0.896659    0.998862    1.113278    0.890741    1.287897    1.390303
min      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
25%      4.000000    3.000000    2.000000    1.000000    1.000000    3.000000
50%      4.000000    3.000000    3.000000    1.000000    2.000000    4.000000
75%      5.000000    4.000000    4.000000    2.000000    3.000000    5.000000
max      5.000000    5.000000    5.000000    5.000000    5.000000    5.000000

         absences          G1          G2          G3
count  395.000000  395.000000  395.000000  395.000000
mean     5.708861   10.908861   10.713924   10.415190
std      8.003096    3.319195    3.761505    4.581443
min      0.000000    3.000000    0.000000    0.000000
25%      0.000000    8.000000    9.000000    8.000000
50%      4.000000   11.000000   11.000000   11.000000
75%      8.000000   13.000000   13.000000   14.000000
max     75.000000   19.000000   19.000000   20.000000
```

[21]:
```python
# see quick info of category values
# include=object parameter ensures only string/object columns are included
# This shows count, unique values, top (most frequent) value, and its frequency

data.describe(include = object)
```

[21]:
```
       school  sex address famsize Pstatus   Mjob   Fjob  reason guardian  \
count     395  395     395     395     395    395    395     395      395
unique      2    2       2       2       2      5      5       4        3
top        GP    F       U     GT3       T  other  other  course   mother
freq      349  208     307     281     354    141    217     145      273

       schoolsup famsup paid activities nursery higher internet romantic
count        395    395  395        395     395    395      395      395
unique         2      2    2          2       2      2        2        2
top           no    yes   no        yes     yes    yes      yes       no
freq         344    242  214        201     314    375      329      263
```

[11]:
```python
# Define the list of numerical columns manually (based on dataset)
numerical_columns =␣
 ↪['age','Medu','Fedu','traveltime','studytime','failures','famrel','freetime','goout','Dalc'
'G3']
# Create a new DataFrame with only numerical columns
df_numeric = data[numerical_columns]

# Display the first few rows
print(df_numeric.head())
```

```
   age  Medu  Fedu  traveltime  studytime  failures  famrel  freetime  goout  \
0   18     4     4           2          2         0       4         3      4
```

```
1   17     1     1           1          2         0       5         3      3
2   15     1     1           1          2         3       4         3      2
3   15     4     2           1          3         0       3         2      2
4   16     3     3           1          2         0       4         3      2

   Dalc  Walc  health  absences  G1  G2  G3
0     1     1       3         6   5   6   6
1     1     1       3         4   5   5   6
2     2     3       3        10   7   8  10
3     1     1       5         2  15  14  15
4     1     2       5         4   6  10  10
```

[13]:
```python
# Define the list of categorical columns manually
categorical_columns = ['school', 'sex', 'address', 'famsize', 'Pstatus',
    'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup',
    'famsup', 'paid', 'activities', 'nursery', 'higher',
    'internet', 'romantic']

# Create a new DataFrame with only categorical columns
df_categorical = data[categorical_columns]

# Display the first few rows
print(df_categorical.head())
```

```
  school sex address famsize Pstatus     Mjob      Fjob  reason guardian  \
0     GP   F       U     GT3       A  at_home   teacher  course   mother
1     GP   F       U     GT3       T  at_home     other  course   father
2     GP   F       U     LE3       T  at_home     other   other   mother
3     GP   F       U     GT3       T   health  services    home   mother
4     GP   F       U     GT3       T    other     other    home   father

  schoolsup famsup paid activities nursery higher internet romantic
0       yes     no   no         no     yes    yes       no       no
1        no    yes   no         no      no    yes      yes       no
2       yes     no  yes         no     yes    yes      yes       no
3        no    yes  yes        yes     yes    yes      yes      yes
4        no    yes  yes         no     yes    yes       no       no
```

Univariate Analysis & Visualizations

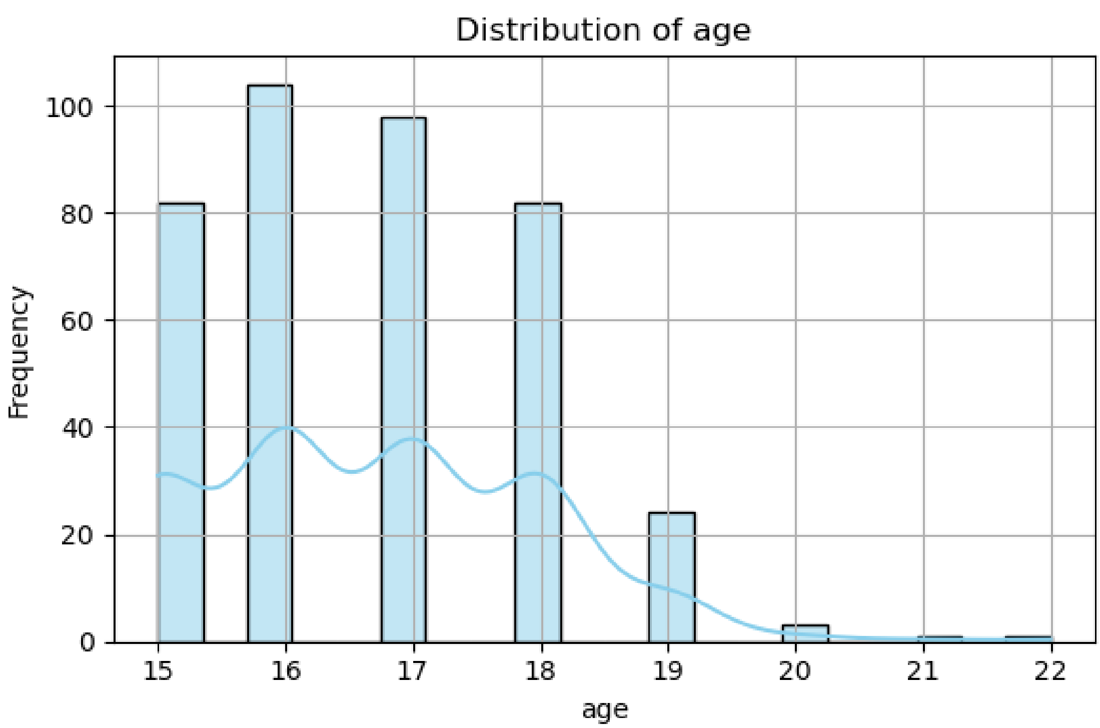
[117]:
```python
# create function that visualized numeric columns using box plot


def hist_plot(column_name, data, bins=20, kde=True, color='skyblue'):
    """
    1) Plots a histogram for a numerical column using seaborn
    2) Inputs:
```

```
        - column_name: string, name of the column to plot
        - data: DataFrame containing the column
        - bins: number of bins for histogram
        - kde: whether to show kernel density estimate
    3) Output:
        - Histogram showing distribution of values
    """
    plt.figure(figsize=(6, 4))  # Create a new figure with specified size
    sns.histplot(data[column_name], kde=kde, bins=bins, color=color)  # Create␣
↪histogram with seaborn
    plt.title(f'Distribution of {column_name}')  # Set plot title
    plt.xlabel(column_name)  # Set x-axis label
    plt.ylabel('Frequency')  # Set y-axis label
    plt.grid(True)  # Add grid lines
    plt.tight_layout()  # Adjust subplot parameters for better fit
    plt.show()  # Display the plot
```

Numerical columns Distribution

Discovering age column

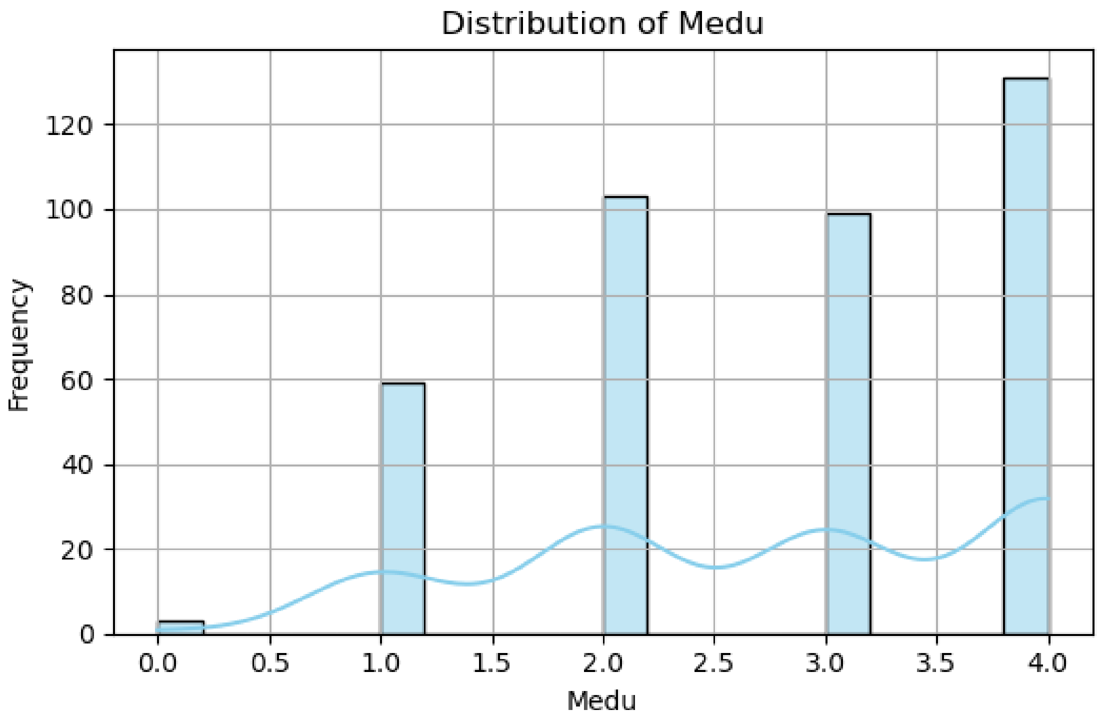
[106]:
```
#call function for numeric columns

hist_plot("age", df_numeric)
```



As we can see in Age Distribution : Insight: The dataset primarily represents a standard high school population, with a minor presence of non-traditional age students.

Discovering Medu column

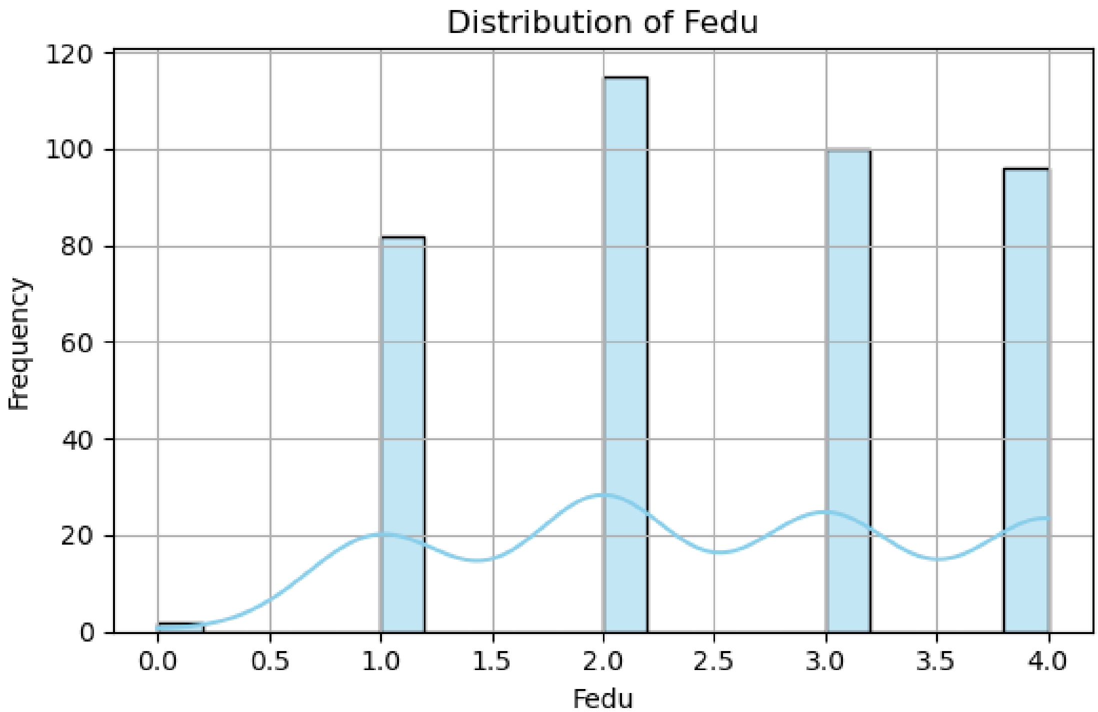
[119]:
```
#call function for numeric columns

hist_plot("Medu", df_numeric)
```



As we can see in Mother's Education Distribution : Insight: Educational attainment among mothers tends to be moderate, potentially impacting students' academic support at home.

Discovering Fedu column

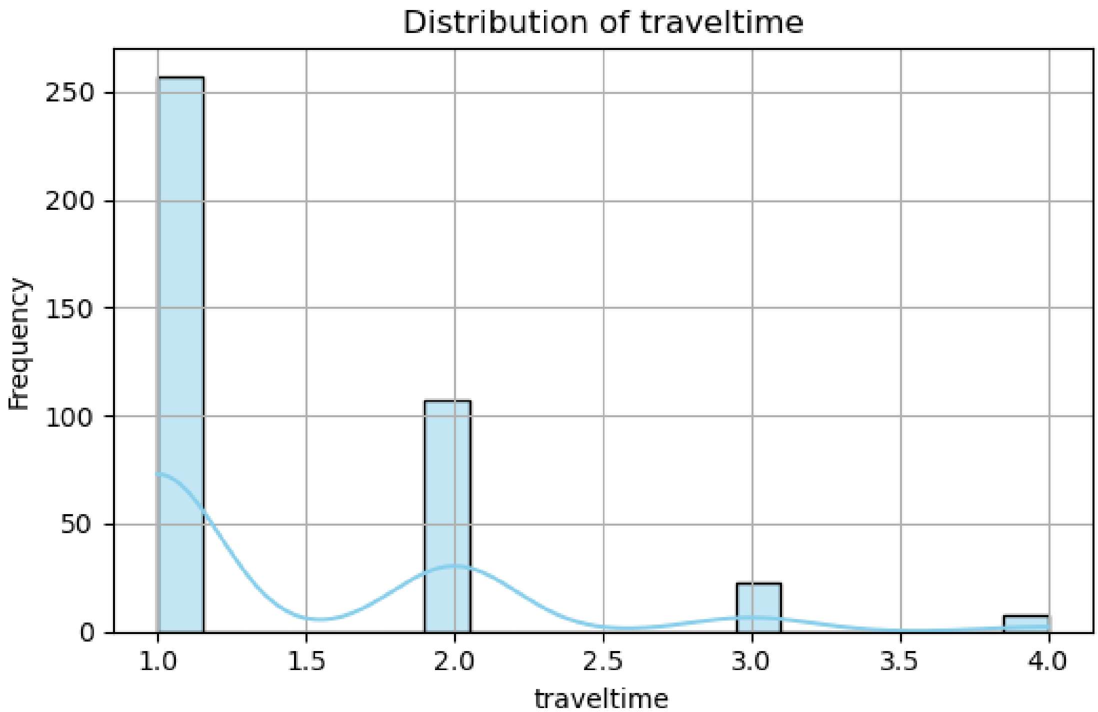
[121]:
```
#call function for numeric columns

hist_plot("Fedu", df_numeric)
```



As we can see Father's Education Distribution: Insight: Both parents generally have a moderate level of education, which may play a role in shaping the academic environment at home.

Discovering traveltime column

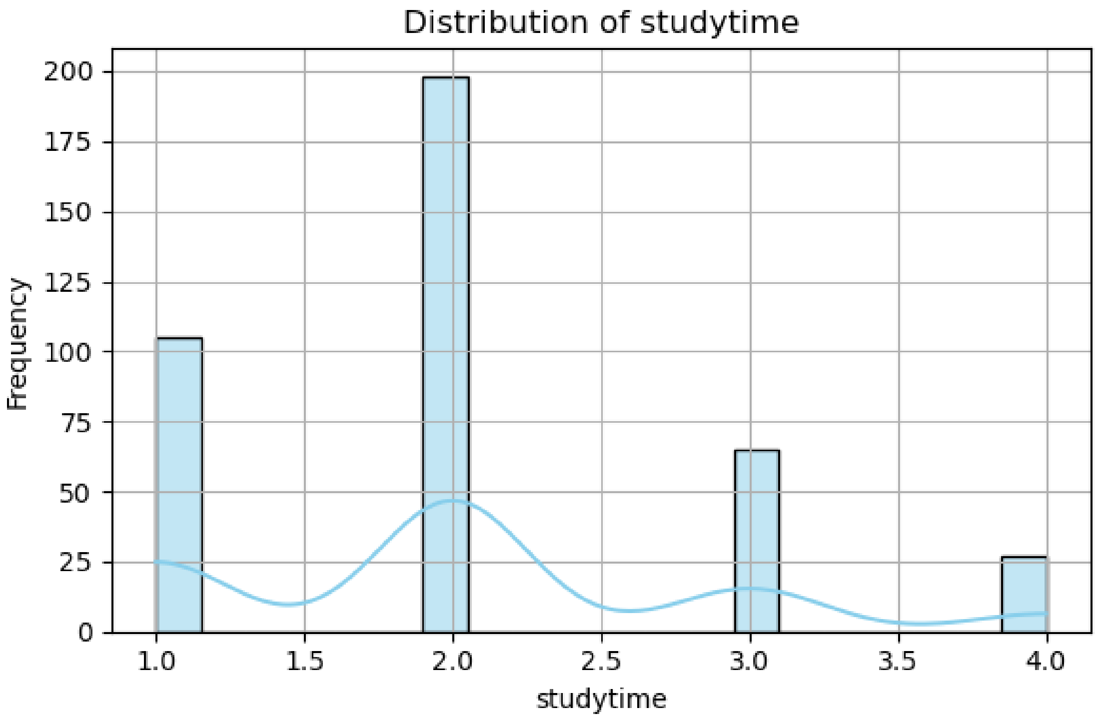
[134]:
```
#call function for numeric columns

hist_plot("traveltime", df_numeric)
```



As we can see Travel Time to School Distribution : Insight: The dataset reflects a student population that generally lives close to school, potentially contributing to better attendance and punctuality.

Discovering studytime column

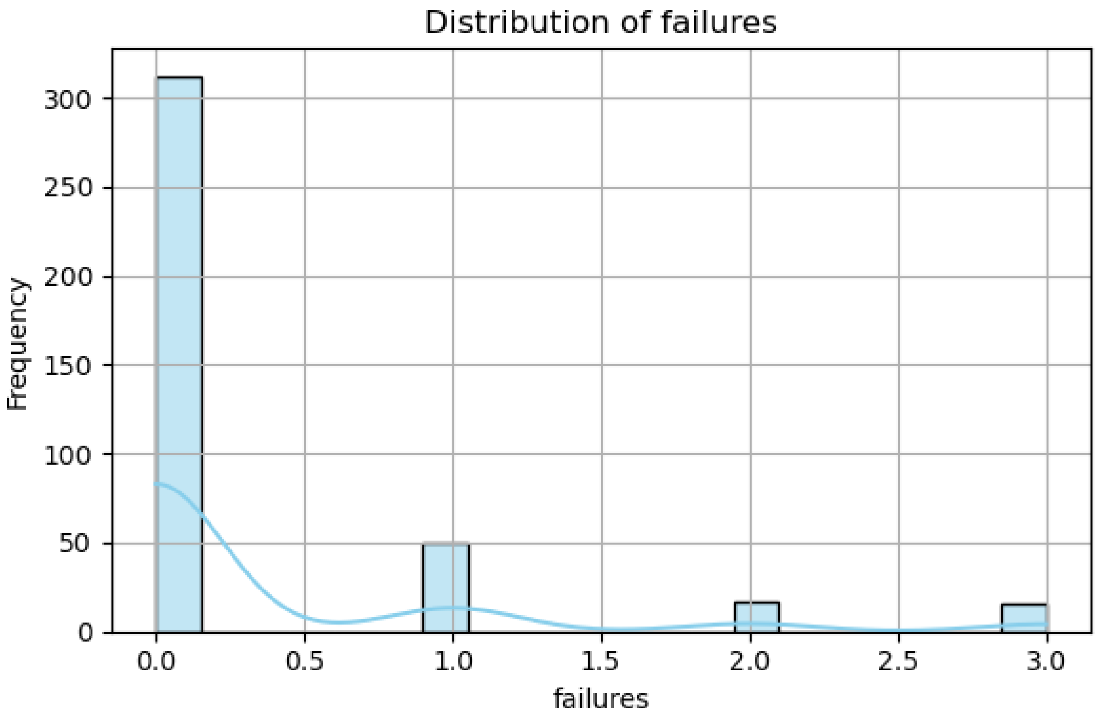
[138]:
```
#call function for numeric columns

hist_plot("studytime", df_numeric)
```



As we can see Weekly Study Time Distribution : Insight: While most students are engaged in some out-of-class study, very few show high study commitment, which may reflect either confidence, workload, or lack of academic pressure.

Discovering failures column

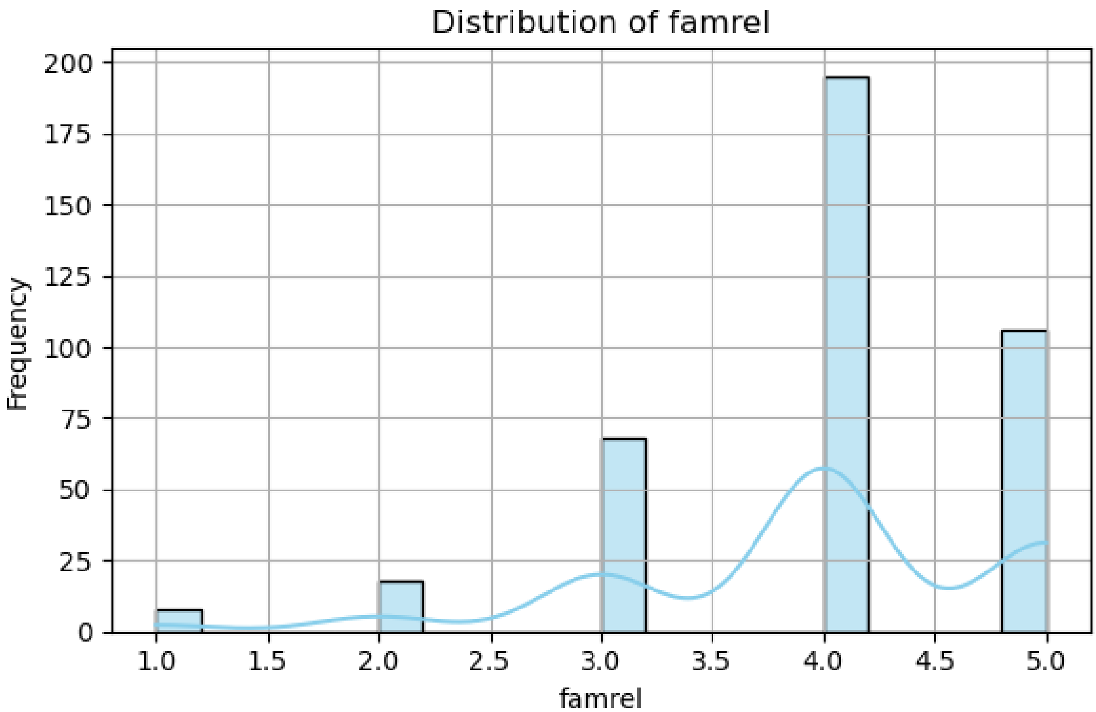
[146]:
```
#call function for numeric columns

hist_plot("failures", df_numeric)
```



As we can see Past Class Failures Distribution: Insight: Most students have a clean academic history, but a small group may require academic support or intervention due to repeated failures.

Discovering famrel column

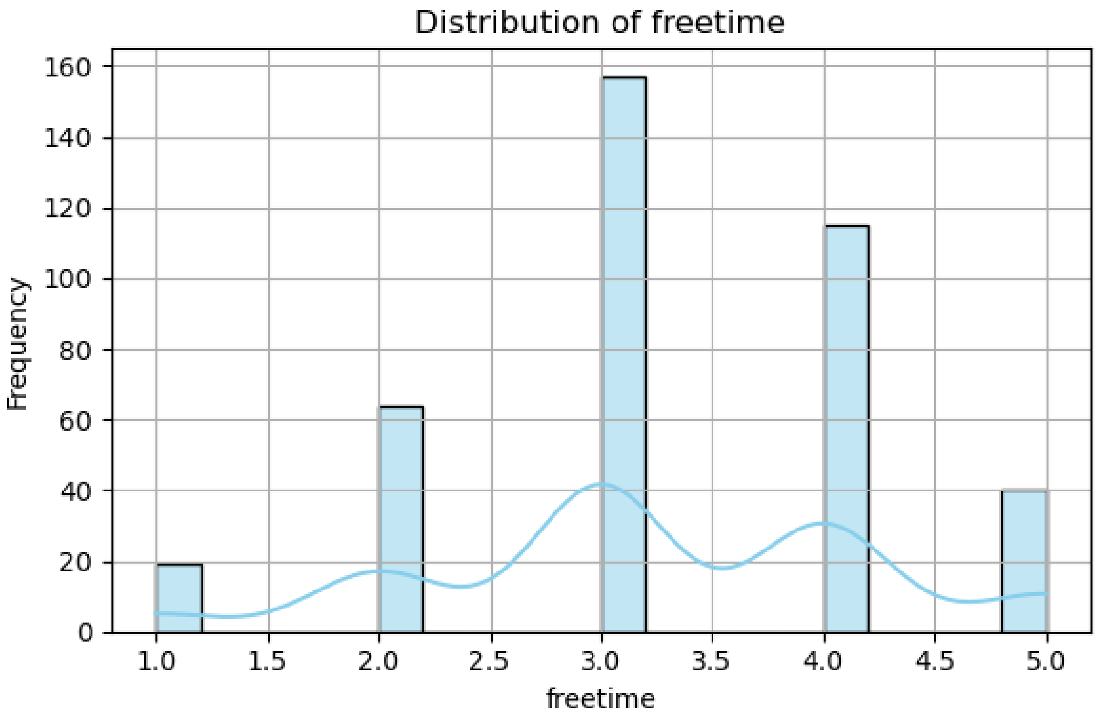
[148]:
```
#call function for numeric columns

hist_plot("famrel", df_numeric)
```



As we can see Family Relationship Quality Distribution : Insight: Most students report having supportive family environments, which can be a protective factor in academic and emotional well-being.

Discovering freetime column

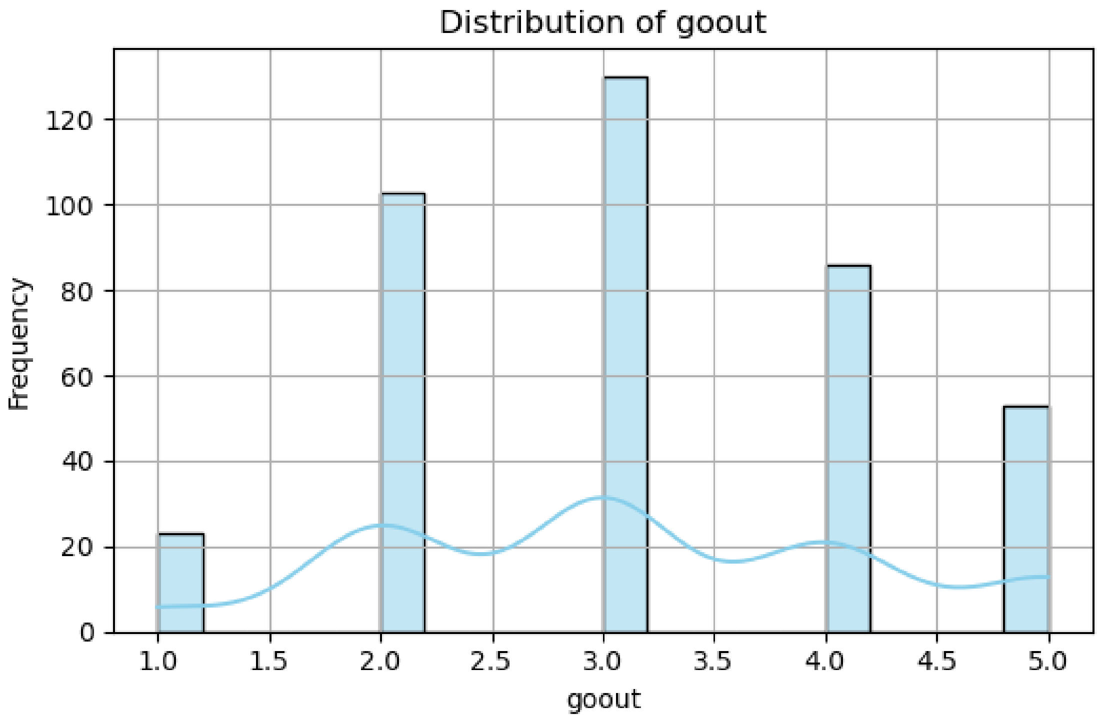
[156]:
```
#call function for numeric columns

hist_plot("freetime", df_numeric)
```



Free Time After School Analysis : Insight: The typical student seems to maintain a healthy balance between academic responsibilities and leisure time, which may positively affect stress levels and performance.

Discovering goout column

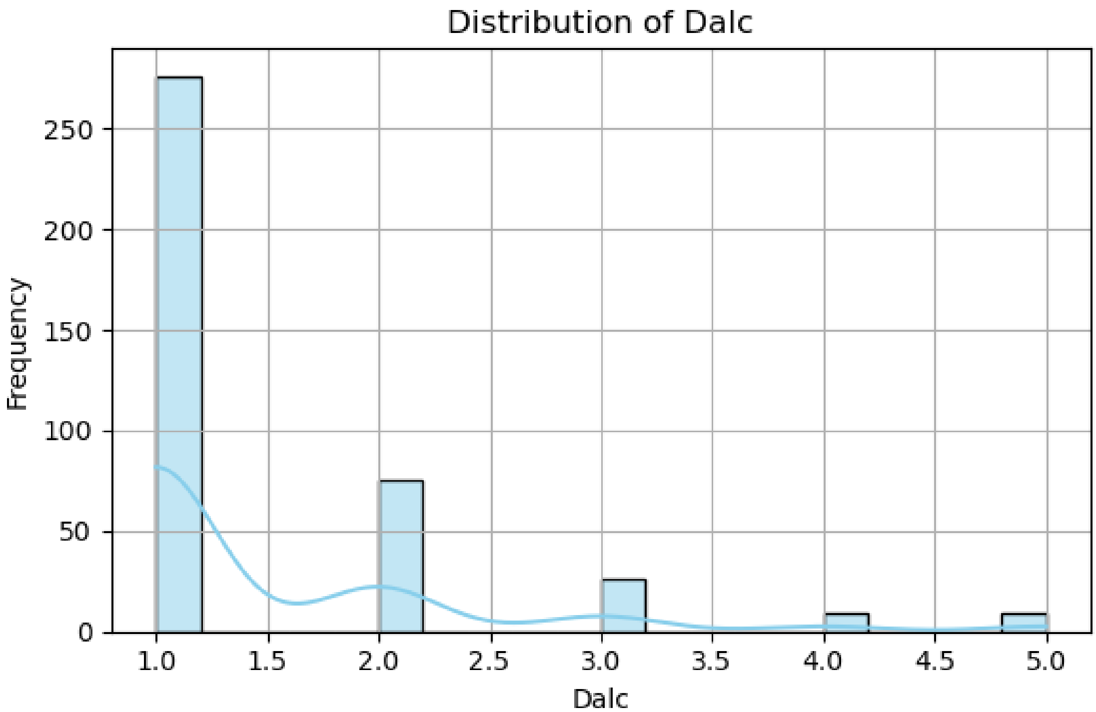
[161]:
```
#call function for numeric columns

hist_plot("goout", df_numeric)
```



As we can see Social Outings (Time Spent with Friends) Distribution : Insight: Most students balance social life with academic responsibilities, though high social activity could potentially influence study time or academic focus for some.

Discovering Dalc column

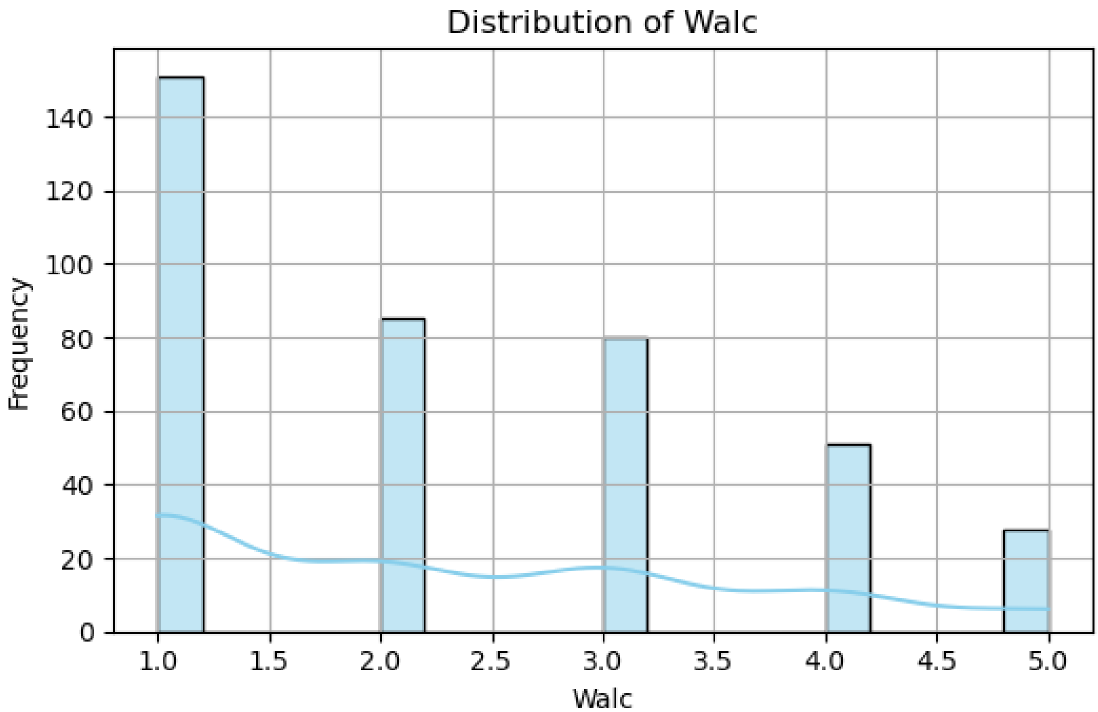
[169]:
```
#call function for numeric columns

hist_plot("Dalc", df_numeric)
```



As we can say Workday Alcohol Consumption Distribution : Insight: Weekday alcohol use is minimal for most students, suggesting that substance use does not commonly interfere with academic responsibilities during the week.

Discovering Walc column

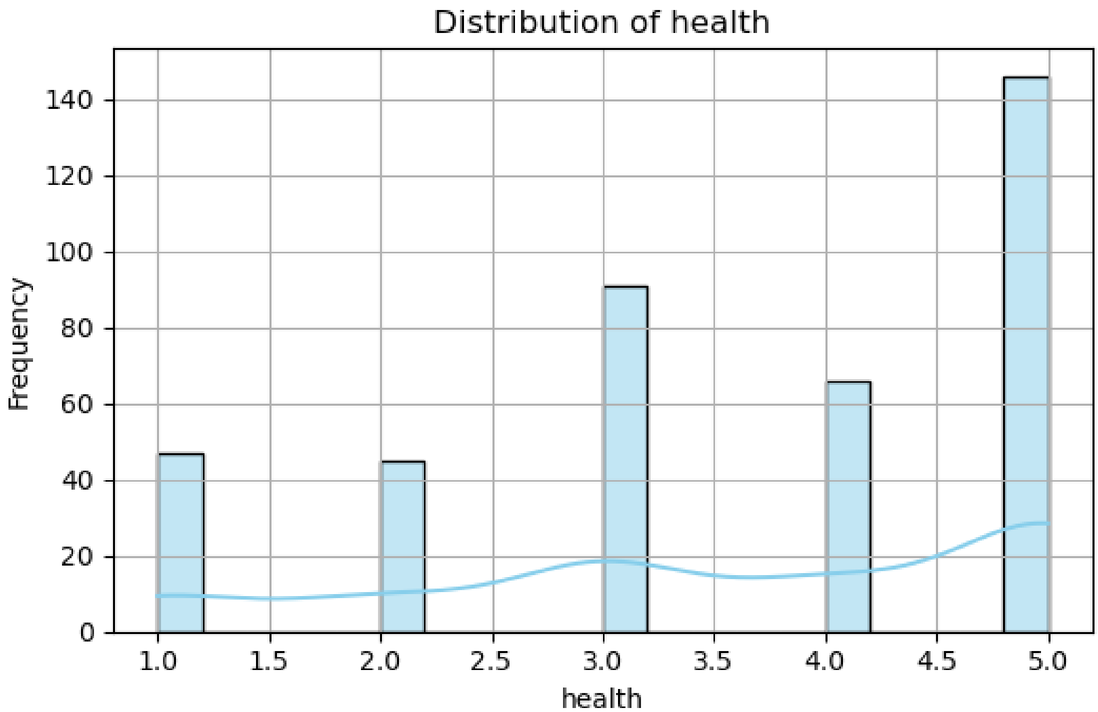
[178]:
```
#call function for numeric columns

hist_plot("Walc", df_numeric)
```



As we can see Weekend Alcohol Consumption Distribution : Insight: Students tend to reserve alcohol consumption for weekends, which could be tied to social gatherings or reduced academic pressure. However, moderate-to-high weekend use may still impact overall health and performance.

Discovering health column

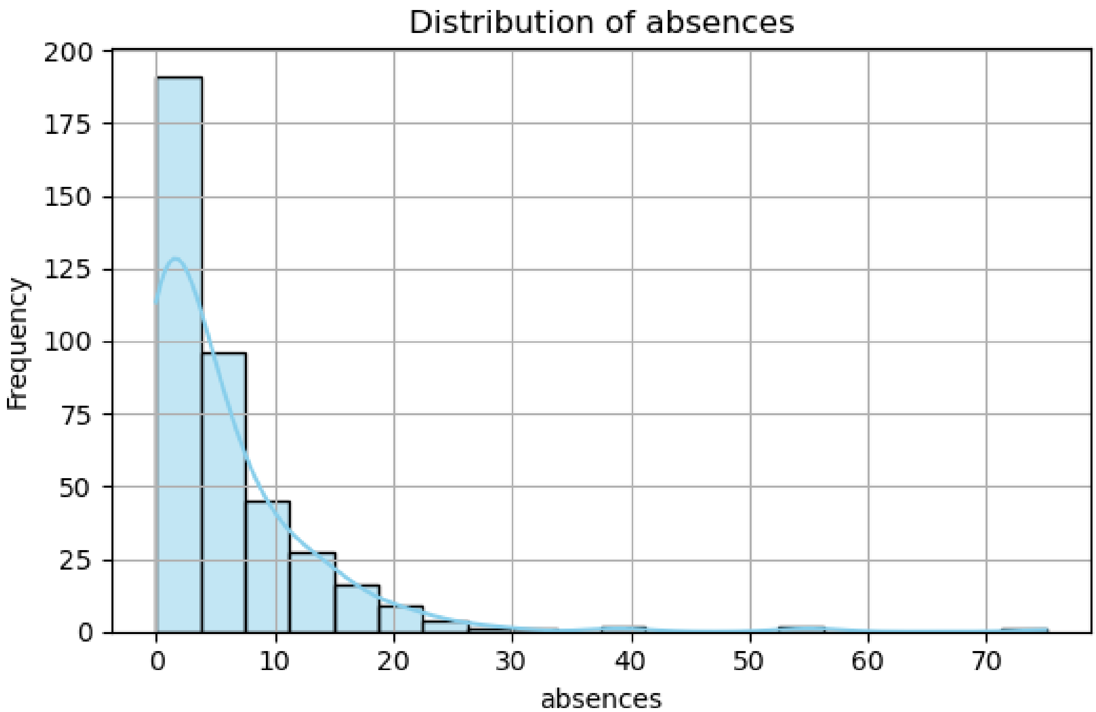
[191]:
```
#call function for numeric columns

hist_plot("health", df_numeric)
```



As we can see Self-Reported Health Status Distribution : Insight: Most students consider their health to be good to very good, which can contribute positively to academic engagement and overall well-being.

Discovering absences column

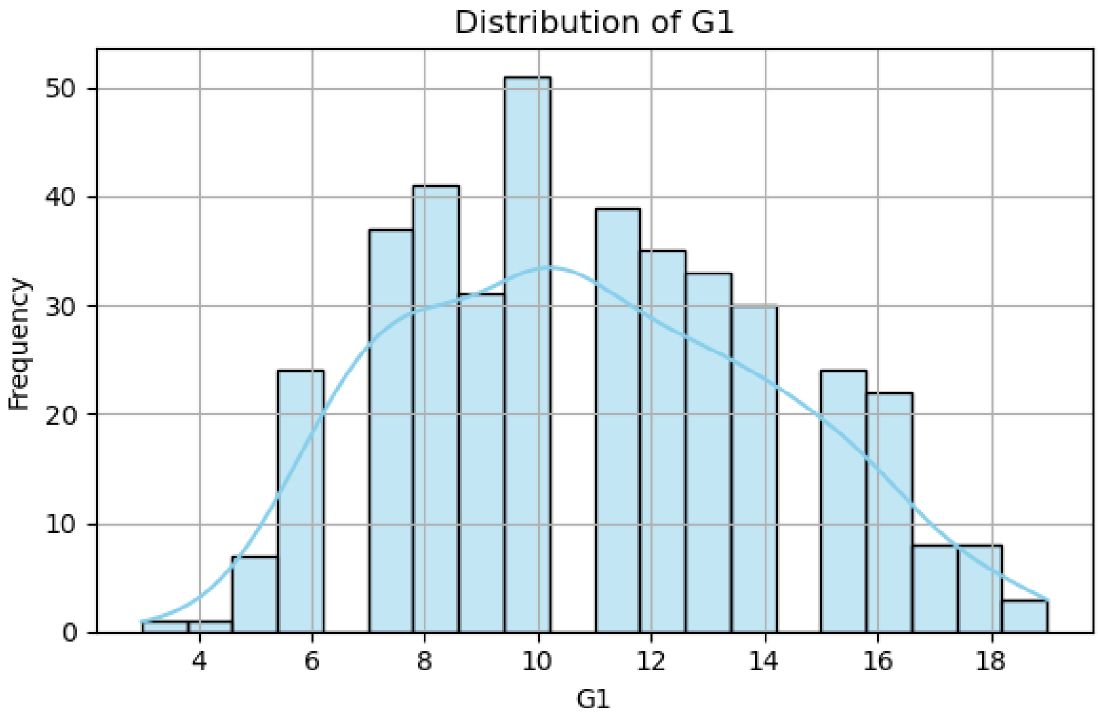
[194]:
```
#call function for numeric columns

hist_plot("absences", df_numeric)
```

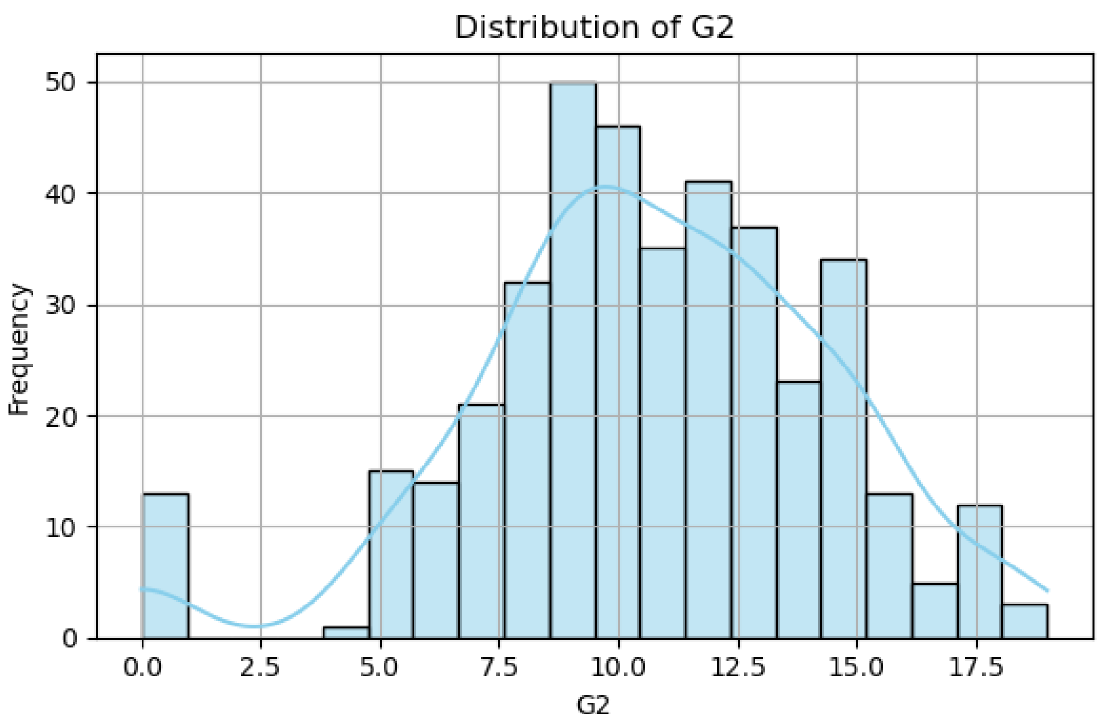


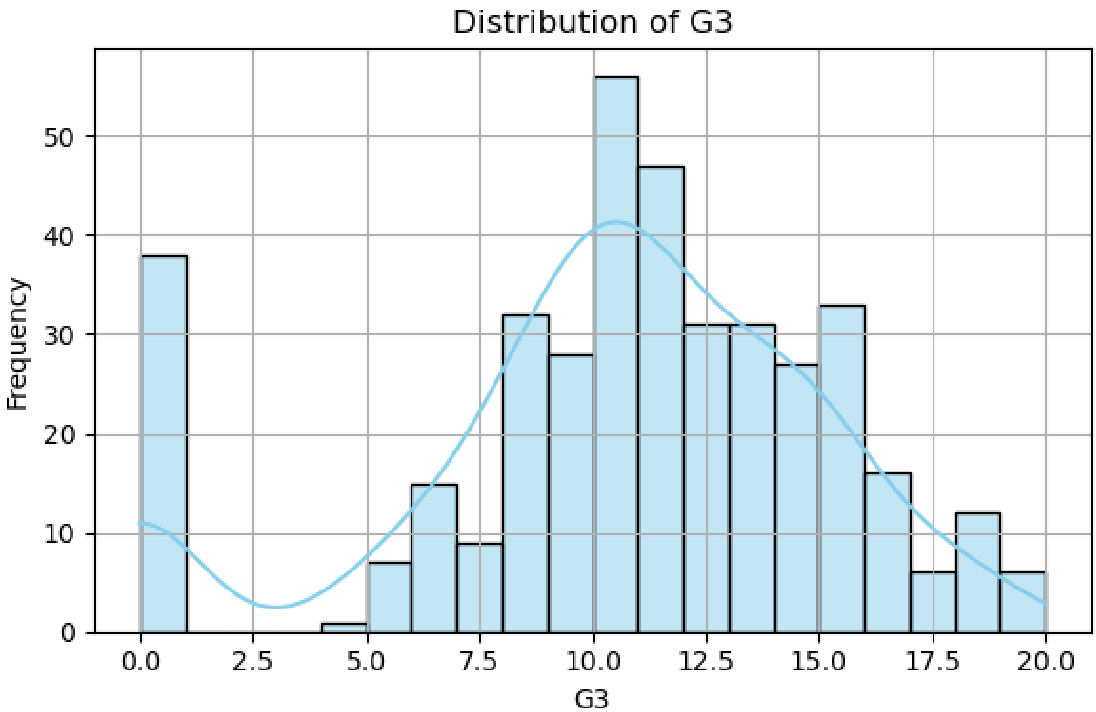
As we can see School Absences Distribution : Insight: While most students attend school regularly, a small subset with high absence rates may be at risk academically and could benefit from targeted support or counseling.

Discovering Grade Progression Analysis (G1, G2, G3)

[212]:
```
# Loop through each grade column (G1, G2, G3)
for col in ["G1", "G2", "G3"]:
    # For each grade column, create a histogram plot using the hist_plot␣
 ↪function
    # This visualizes the distribution of student grades across the three␣
 ↪periods
    hist_plot(col, df_numeric)
```


Distribution of G1
Frequency
G1
0
10
20
30
40
50
4
6
8
10
12
14
16
18


Distribution of G2
Frequency
G2
0
10
20
30
40
50
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



Grade Progression Analysis (G1, G2, G3) : Insight: Grades tend to improve over time for most students, possibly due to accumulated learning and adjustment to academic expectations. However, notable drops in G3 for some students suggest potential end-term stress, burnout, or external factors.

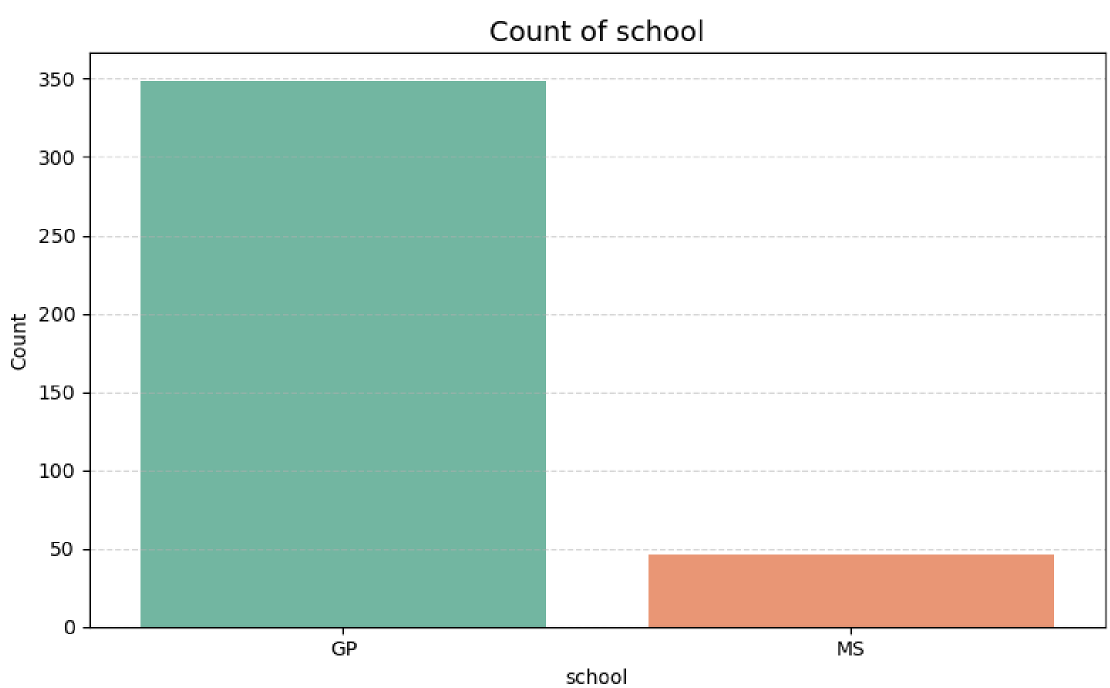
Categorical Column Distribution

[16]:
```
# create function to visualized categorical column using countplot

def count_plot(column_name, data, hue=None, rotation=0, palette='pastel'):
    """  Parameters:
    - column_name: str, name of the categorical column
    - data: pd.DataFrame, your dataset
    - hue: str, optional column for subgrouping (e.g., 'sex')
    - rotation: int, degree of x-axis label rotation
    - palette: str or list, color palette for bars

    Output:
    - Count plot with frequency labels above bars
    """

    plt.figure(figsize=(8, 5))  # Create a figure with specified size
```

```python
    # Generate the countplot using seaborn sns.countplot(x='your_column',␣
↪data=df, hue='your_column', palette='Set2', legend=False)
    sns.countplot(
        x=column_name,          # Column to count and display on x-axis
        data=data,              # DataFrame containing the data
        hue=column_name,                # Optional column for color grouping
        order=data[column_name].value_counts().index, # Sort bars by frequency
        palette='Set2')         # Color scheme for the plot

    plt.title(f'Count of {column_name}', fontsize=14)  # Set plot title with␣
↪custom font size
    plt.xticks(rotation=rotation)  # Rotate x-axis labels by specified degrees
    plt.xlabel(column_name)  # Set x-axis label to the column name
    plt.ylabel('Count')  # Set y-axis label to 'Count'
    plt.tight_layout()  # Adjust subplot params to give specified padding
    plt.grid(axis='y', linestyle='--', alpha=0.5)  # Add horizontal grid lines␣
↪with dashed style and transparency
    plt.show()  # Display the plot
```

Discovering school column

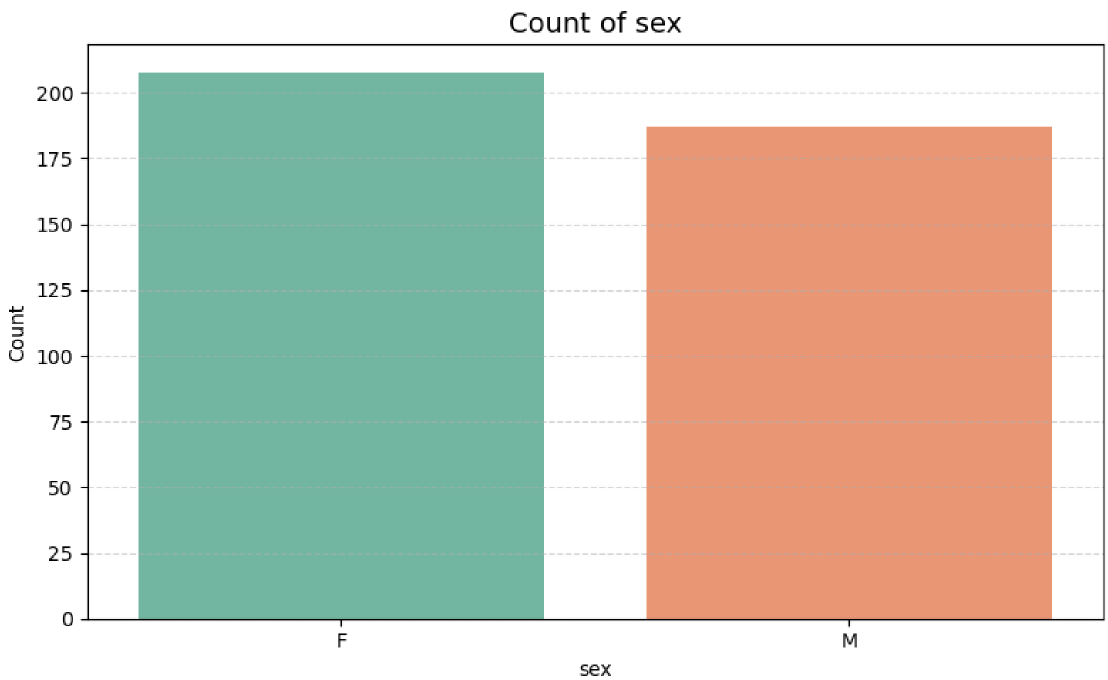
[254]:
```python
#call function for categoric columns

count_plot("school", df_categorical)
```



As we can say School Distribution : Insight: GP appears to be the dominant school in the dataset. Any findings could be more representative of GP students.

Discovering sex column

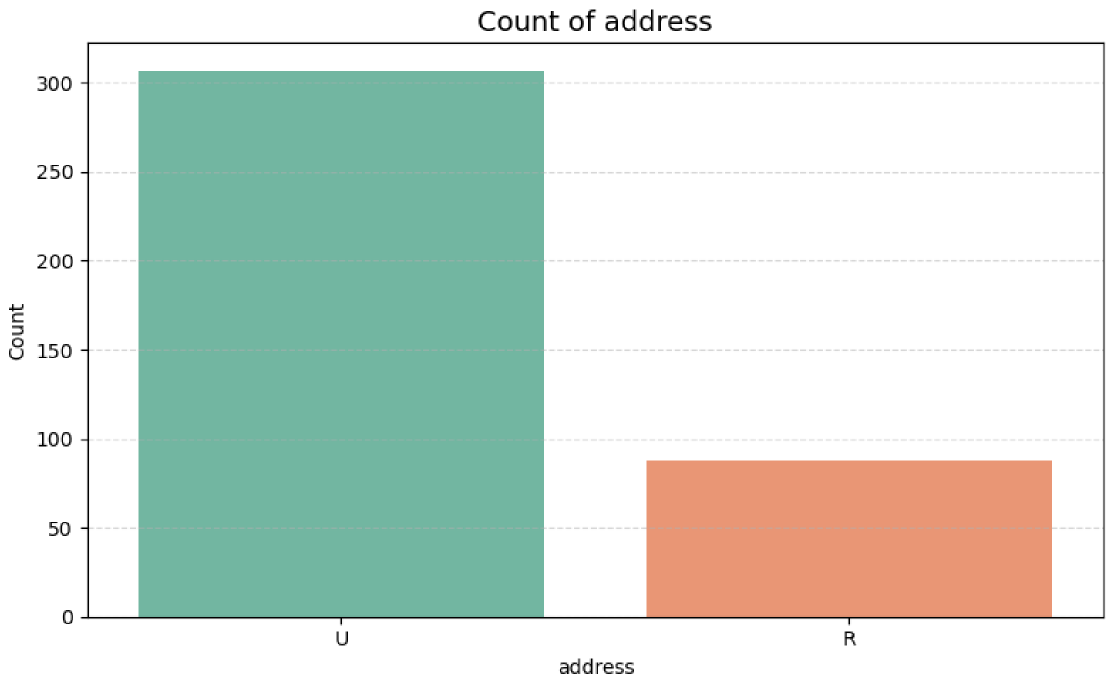
[256]:
```
#call function for categorical columns

count_plot("sex", df_categorical)
```



As we can say Sex Distribution : Insight: This balanced split ensures that gender-related analysis is statistically sound and unbiased.

Discovering address column

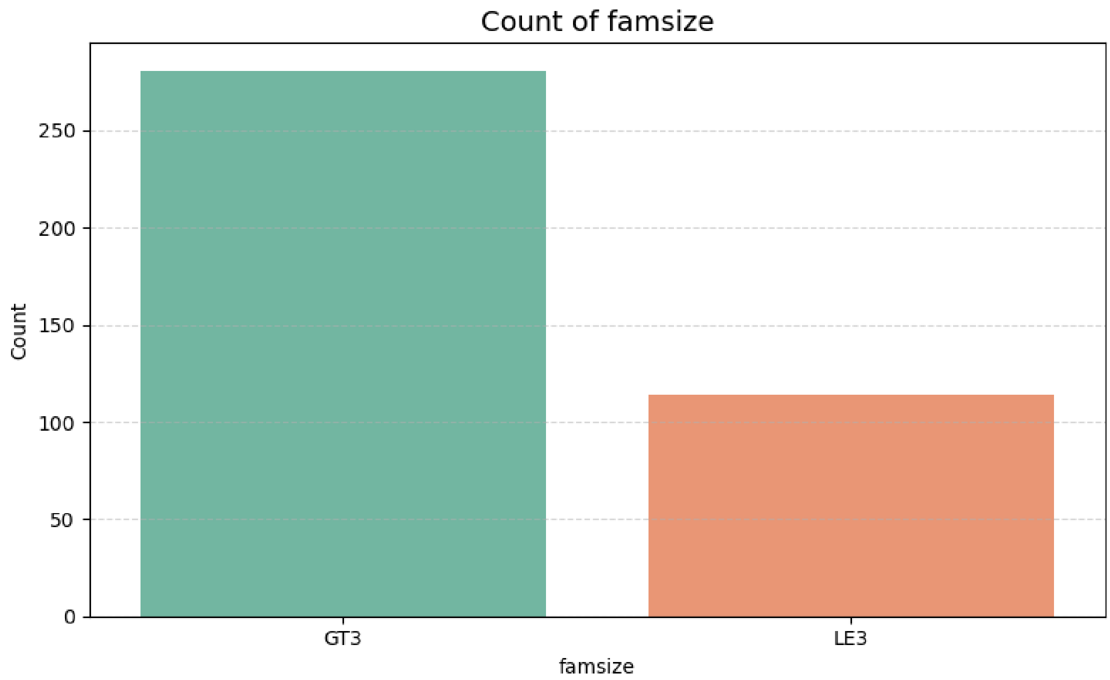
[261]:
```
#call function for categorical columns

count_plot("address", df_categorical)
```



As we can say Address Distribution : Insight: Students from urban settings might benefit from better infrastructure, internet access, or academic support.

Discovering famsize column

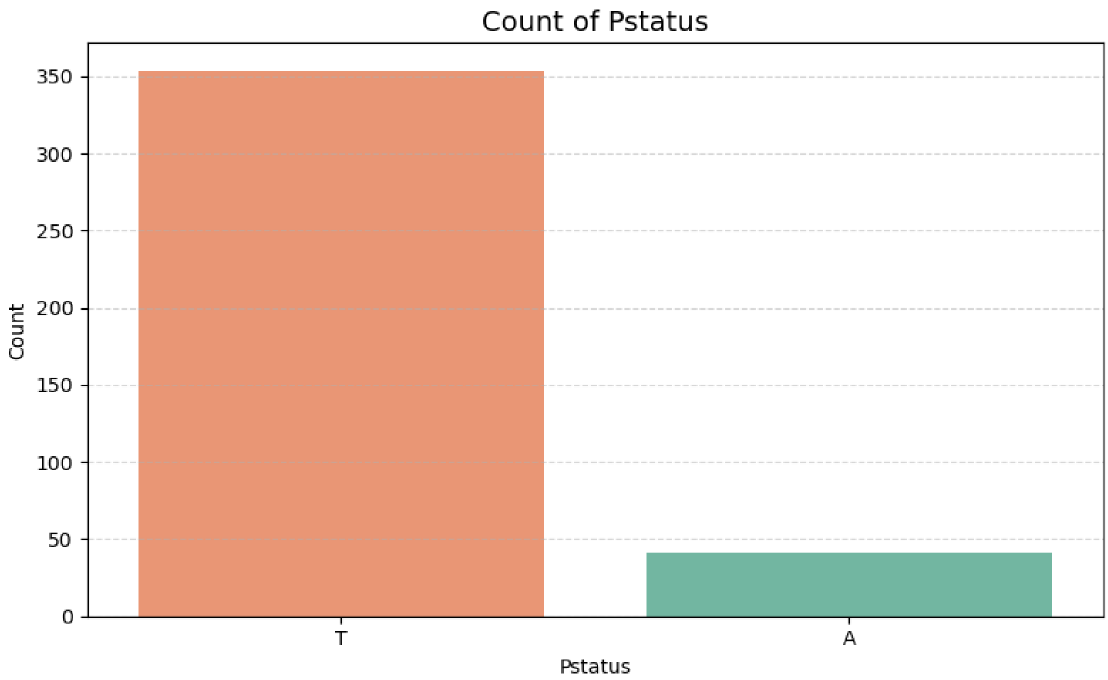
[268]:
```
#call function for categorical columns

count_plot("famsize", df_categorical)
```



As we can say famsize Distribution : Insight: Family size could influence time and attention available for individual study or support.

Discovering Pstatus column

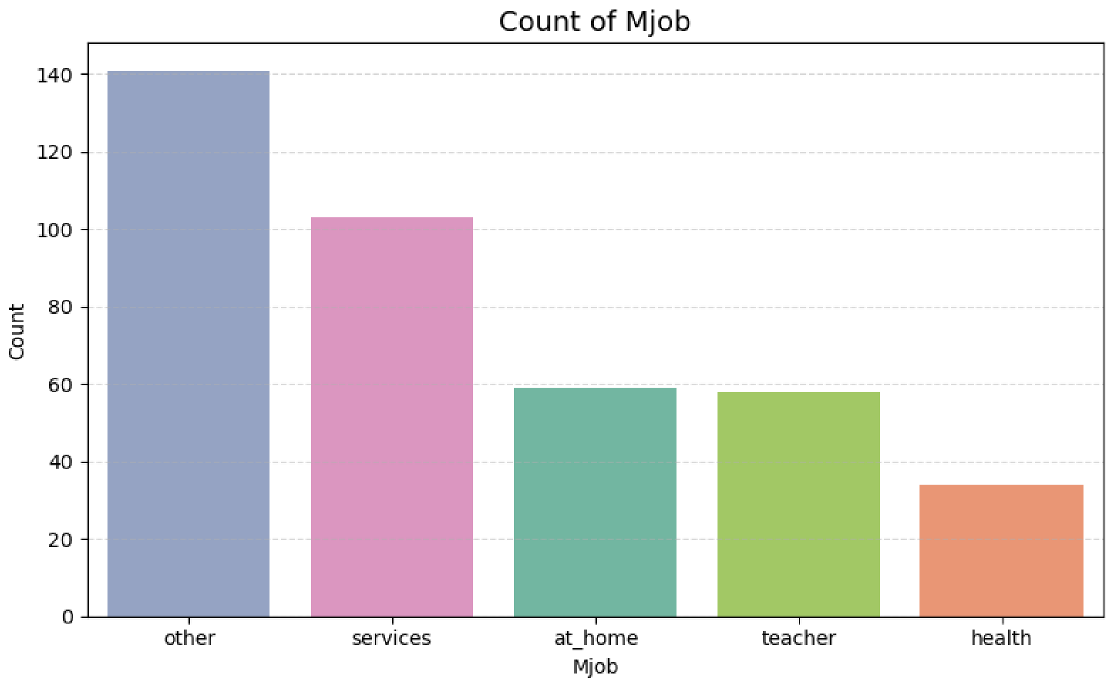
[271]:
```
#call function for categorical columns

count_plot("Pstatus", df_categorical)
```



As we can say Pstatus Distribution : Insight: Family structure could influence emotional and academic stability.

Discovering Mjob (Mother's Job) column

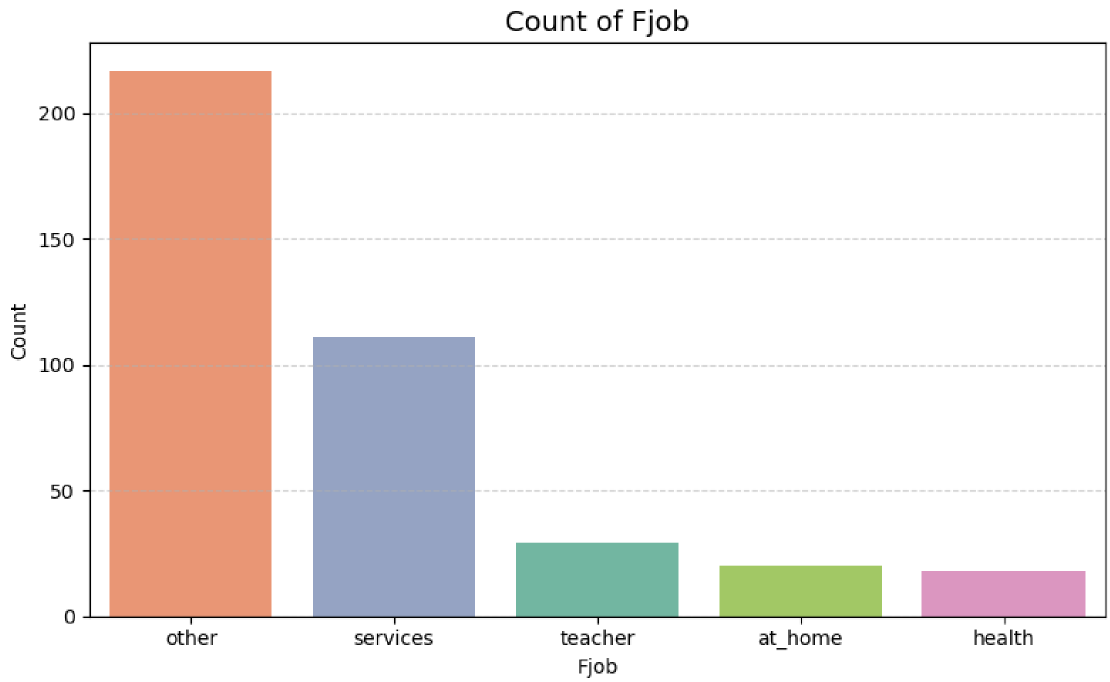
[276]:
```
#call function for categorical columns

count_plot("Mjob", df_categorical)
```



As we can say Mjob Distribution : Insight:Mothers in education might positively impact academic motivation.

Discovering Fjob column

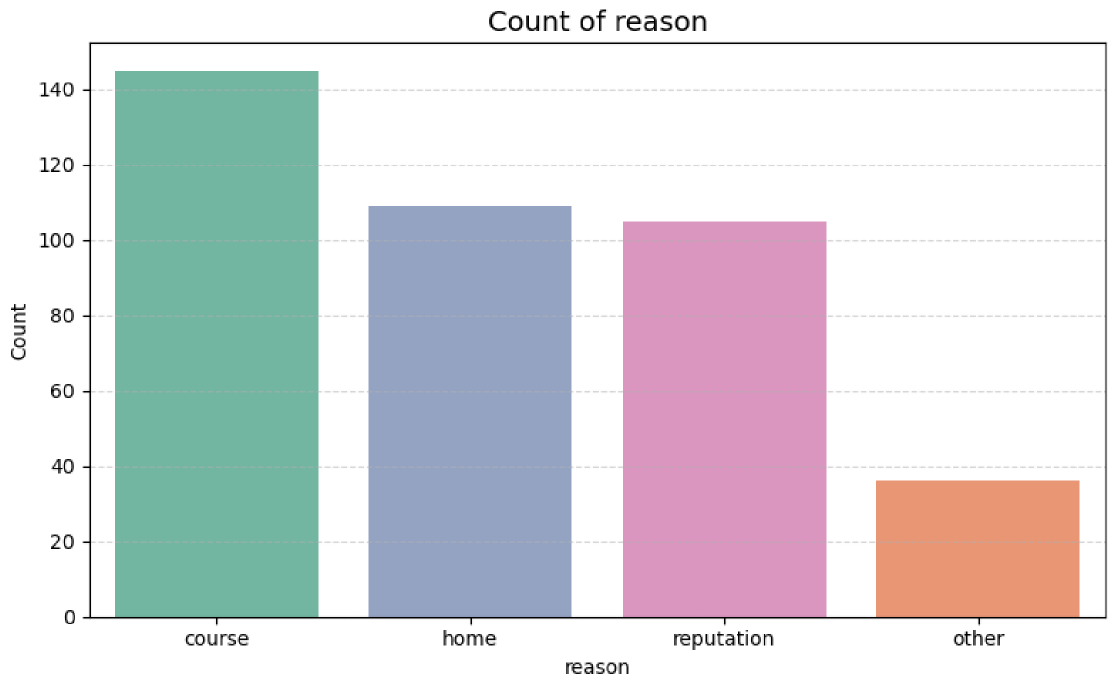
[279]:
```
#call function for categorical columns

count_plot("Fjob", df_categorical)
```



As we can say Fjob Distribution : Insight: The father's professional background varies more, which may affect family income and priorities.

Discovering reason column

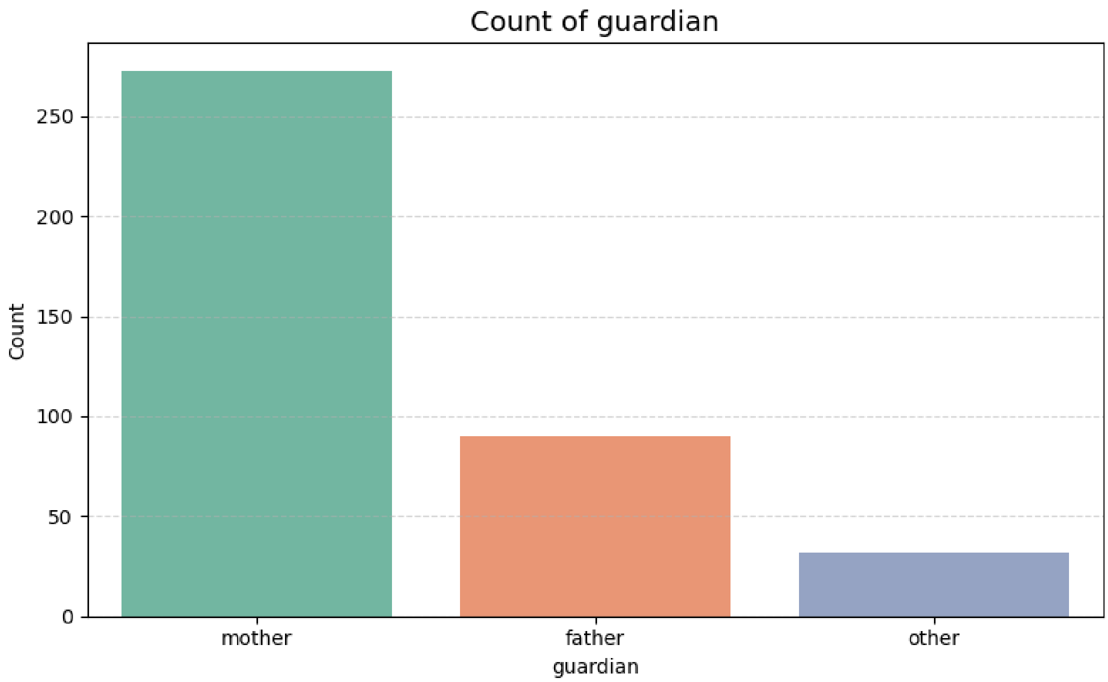
[284]:
```
#call function for categorical columns

count_plot("reason", df_categorical)
```



As we can say reason Distribution : Insight: Indicates a goal-oriented student group driven by academic or career planning.

Discovering guardian column

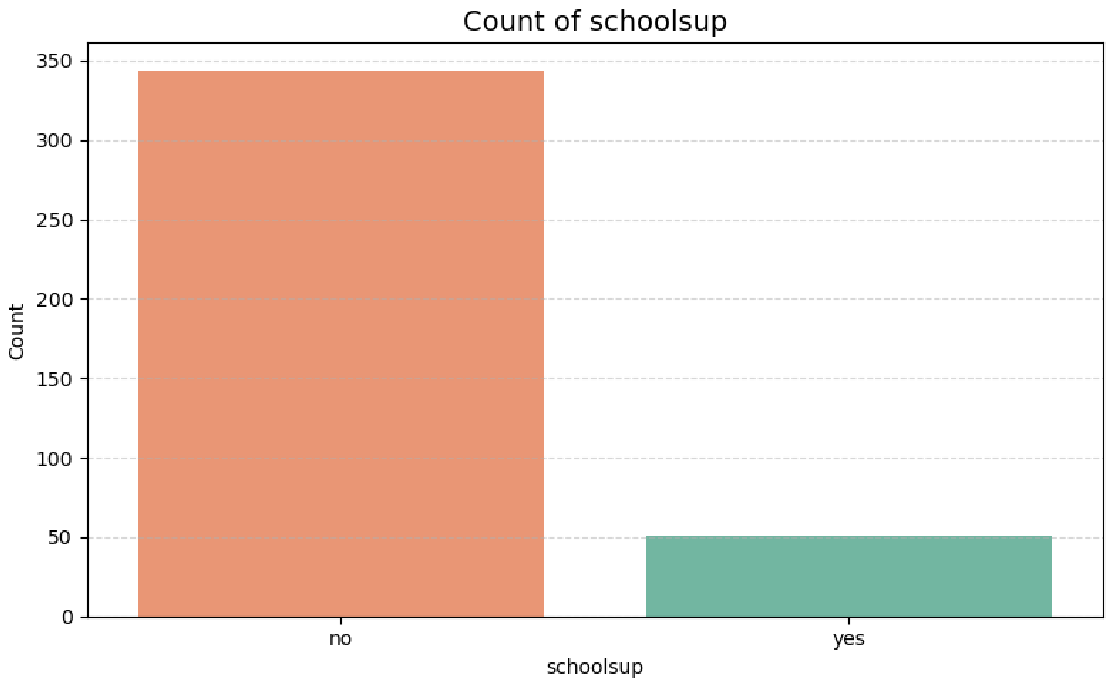
[288]:
```
#call function for categorical columns

count_plot("guardian", df_categorical)
```



As we can say guardian Distribution : Insight: Suggests mothers are the primary caregivers, potentially influencing emotional and academic support systems.

Discovering schoolsup (School Support) column

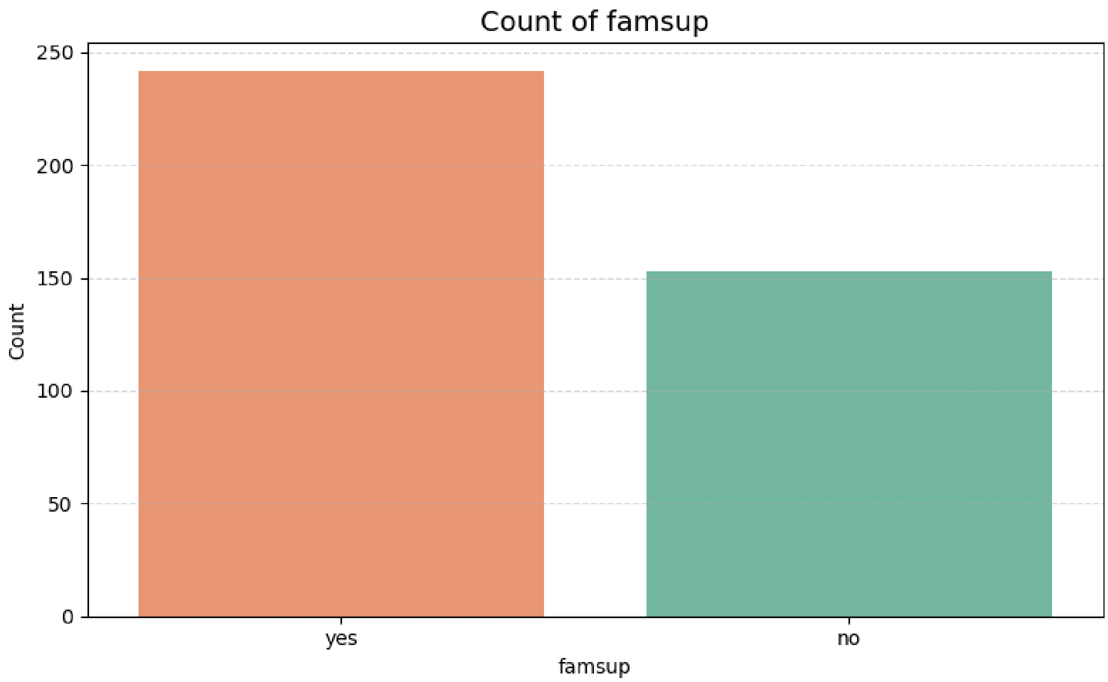
[293]:
```
#call function for categorical columns

count_plot("schoolsup", df_categorical)
```



As we can say schoolsup Distribution : Insight: This might indicate gaps in resource access or support for students who struggle.

Discovering famsup (Family Support) column

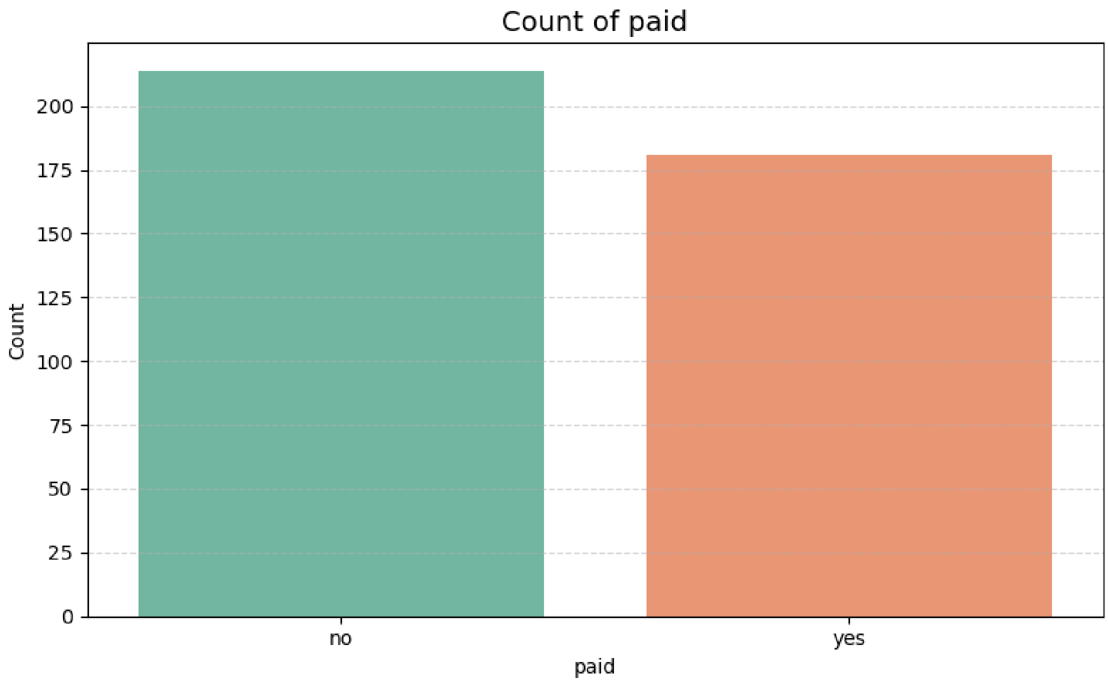
[331]:
```
#call function for categorical columns

count_plot("famsup", df_categorical)
```



As we can say famsup Distribution : Insight: Highlights variation in parental involvement and its potential effect on performance.

Discovering paid (Extra Paid Classes) column

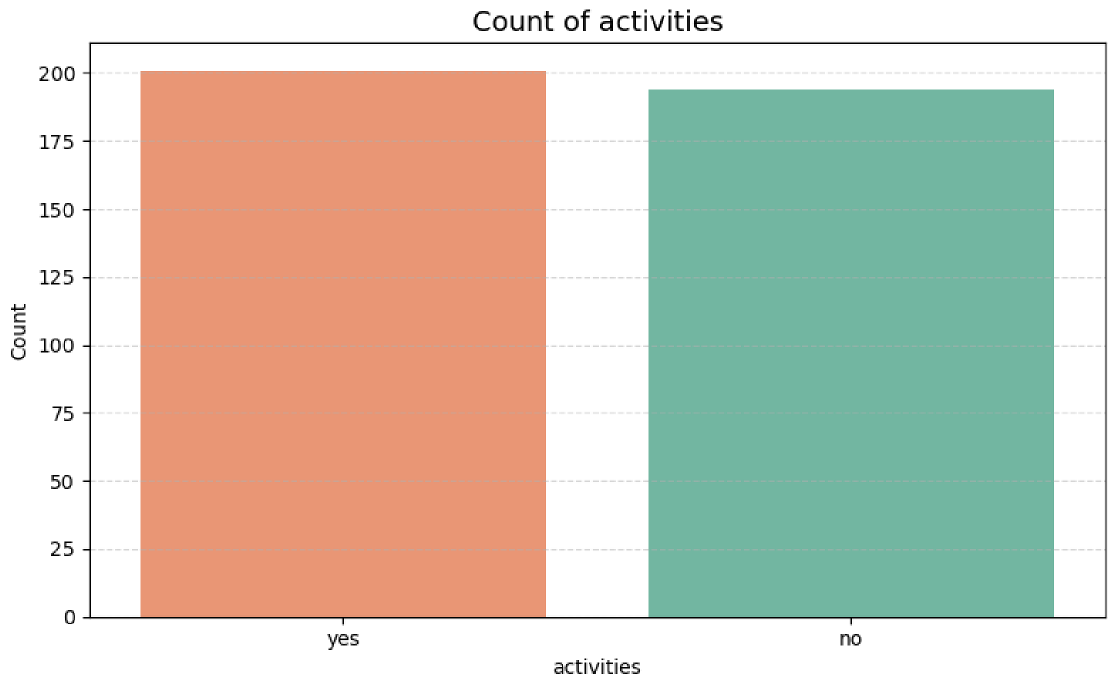
[333]:
```
#call function for categorical columns

count_plot("paid", df_categorical)
```



As we can say paid Distribution: Insight: May reflect financial constraints or reliance on in-school education.

Discovering activities column

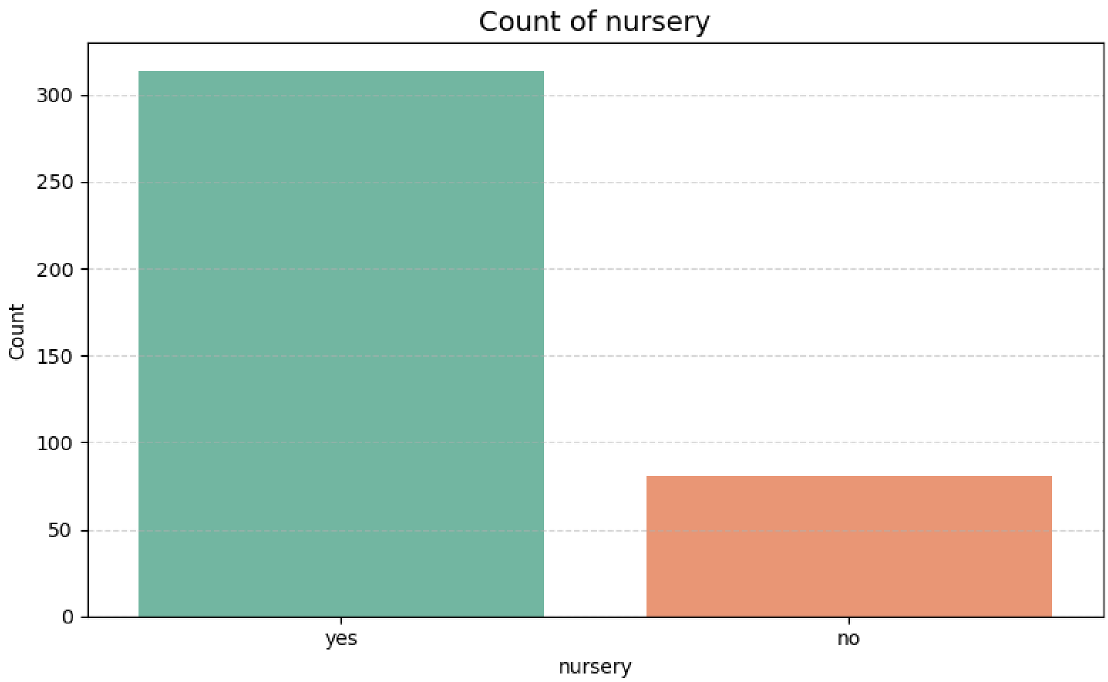
[335]:
```
#call function for categorical columns

count_plot("activities", df_categorical)
```



As we can say activities Distribution : Insight: Indicates a well-rounded student group with non-academic engagements.

Discovering nursery column

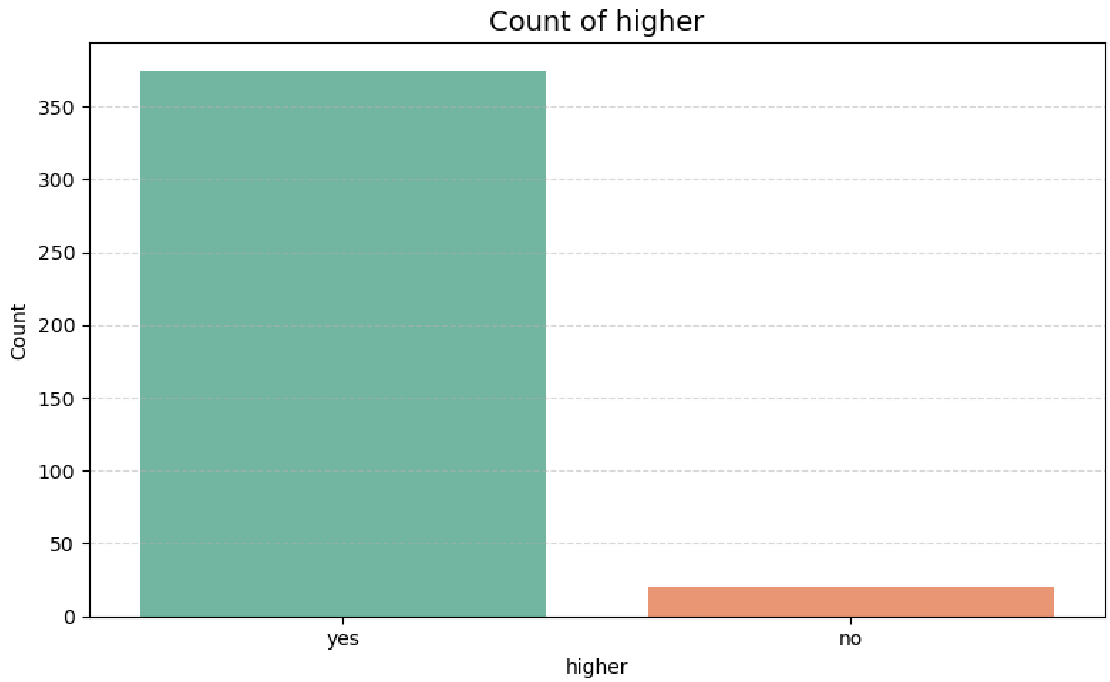
[337]:
```
#call function for categorical columns

count_plot("nursery", df_categorical)
```



As we can say nursery Distribution : Insight: Early education might provide foundational advantages in learning.

Discovering higher (Wants Higher Education) column

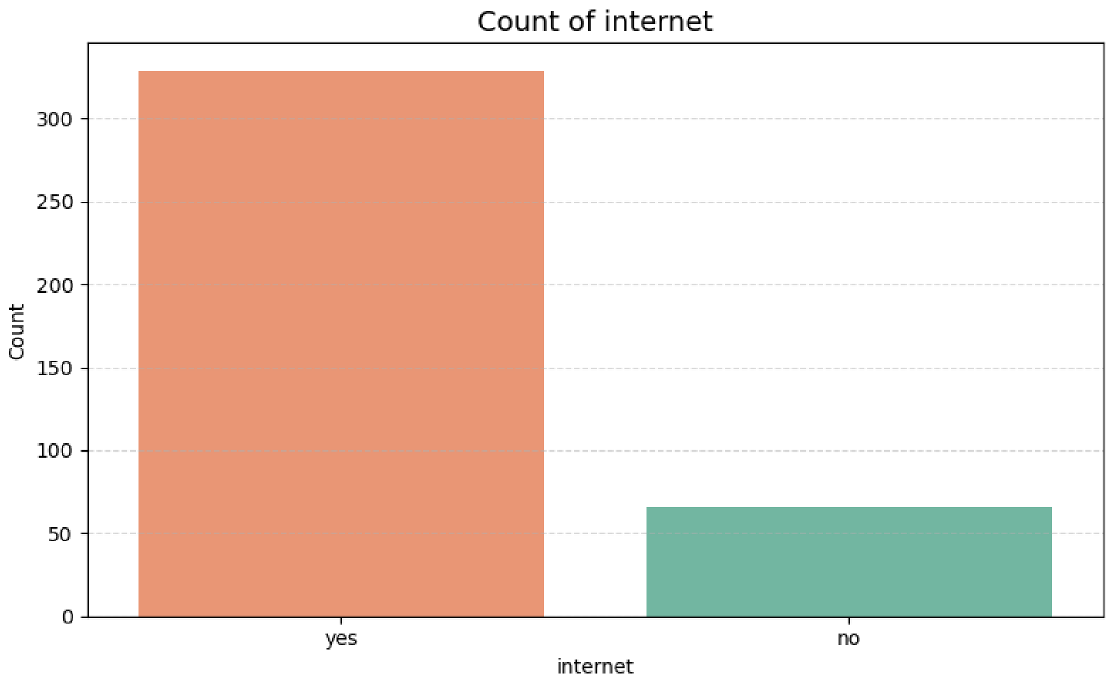
[339]:
```
#call function for categorical columns

count_plot("higher", df_categorical)
```



As we can say higher Distribution : Insight :Reflects a highly motivated academic population.

Discovering internet column

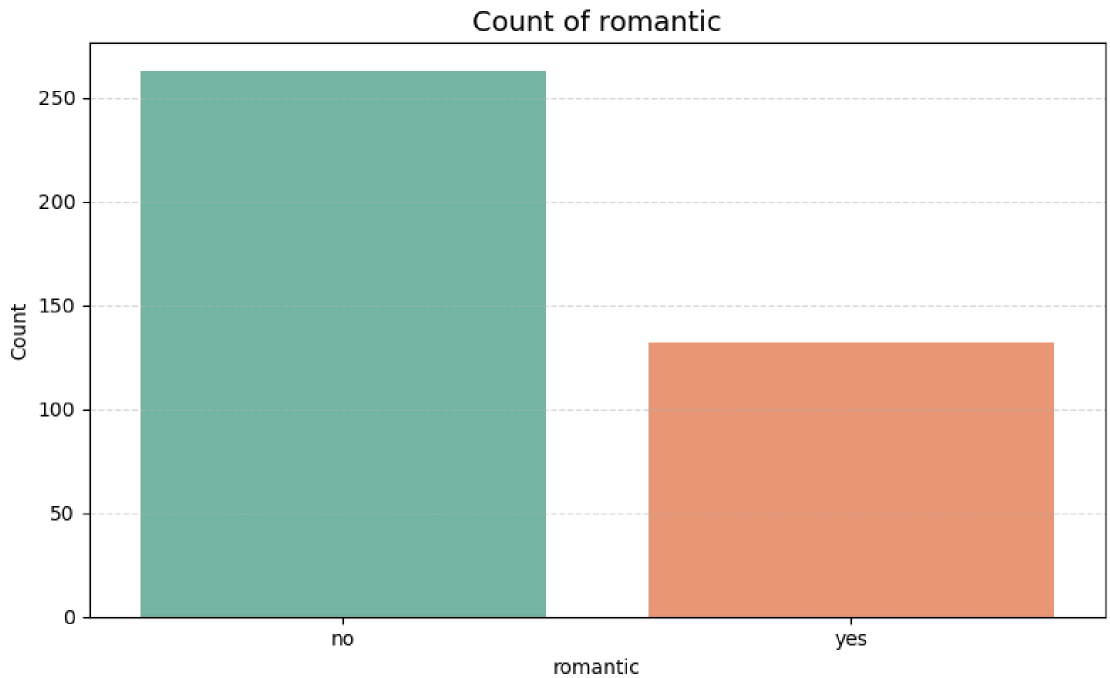
[341]:
```
#call function for categorical columns

count_plot("internet", df_categorical)
```



As we can say internet Distribution : Insight : This accessibility supports digital learning and research opportunities.

Discovering romantic column

[325]:
```
#call function for categorical columns

count_plot("romantic", df_categorical)
```

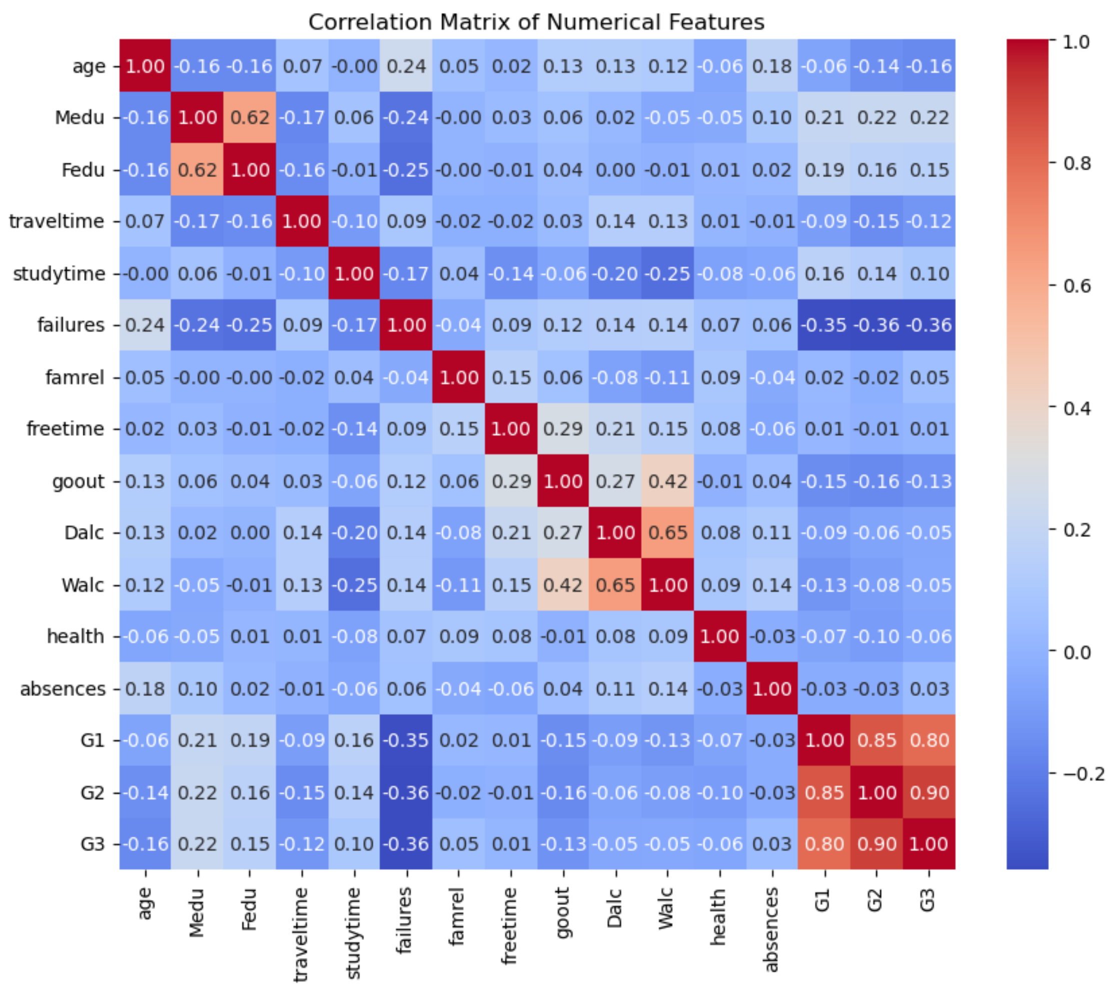


As we can say romantic Distribution : Insight : Students might be more focused on academics, although a few could experience distraction or emotional stress.

Bivariate Analysis & Visualizations

Correlation Matrix (Numerical Columns Only)

[27]:
```
# Create a figure with specified size (10x8 inches)
plt.figure(figsize=(10, 8))
# Generate a heatmap of correlation matrix for numeric columns
# annot=True shows correlation values in each cell
# cmap='coolwarm' sets the color scheme (blue for negative, red for positive␣
↪correlations)
# fmt=".2f" formats the numbers to show 2 decimal places
sns.heatmap(df_numeric.corr(), annot=True, cmap='coolwarm', fmt=".2f")
# Add a title to the heatmap
plt.title("Correlation Matrix of Numerical Features")
# Display the plot
plt.show()
```



[35]:
```python
correlations = df_numeric.corr()['G3'].sort_values(ascending=False)
print(correlations)
```

```
G3            1.000000
G2            0.904868
G1            0.801468
Medu          0.217147
Fedu          0.152457
studytime     0.097820
famrel        0.051363
absences      0.034247
freetime      0.011307
Walc         -0.051939
Dalc         -0.054660
health       -0.061335
traveltime   -0.117142
goout        -0.132791
```

```
age          -0.161579
failures     -0.360415
Name: G3, dtype: float64
```

Note: Correlation coefficients range from -1 to 1. Values closer to 1 or -1 indicate stronger positive or negative relationships, respectively.

Numerical vs G3 (Regression Plots)

[39]:
```python
#Make a regression plot(function) comparing a numerical column with the final␣
↪grade G3.

def reg_plot(column_name, data, color='skyblue'):
    """
    Parameters:
    - column_name: str, name of the numerical column
    - data: pd.DataFrame, your dataset
    - color: str, color of the scatter/regression line

    Output:
    - Regression plot with x = column_name and y = G3
    """
    plt.figure(figsize=(8, 5))

    sns.regplot(
        x=column_name,
        y='G3',
        data=data,
        scatter_kws={'alpha': 0.6},
        line_kws={'color': 'red'},
        color=color
    )

    plt.title(f'Regression Plot: {column_name} vs G3', fontsize=14)
    plt.xlabel(column_name)
    plt.ylabel('Final Grade (G3)')
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.tight_layout()
    plt.show()
```

Age vs G3 distribution

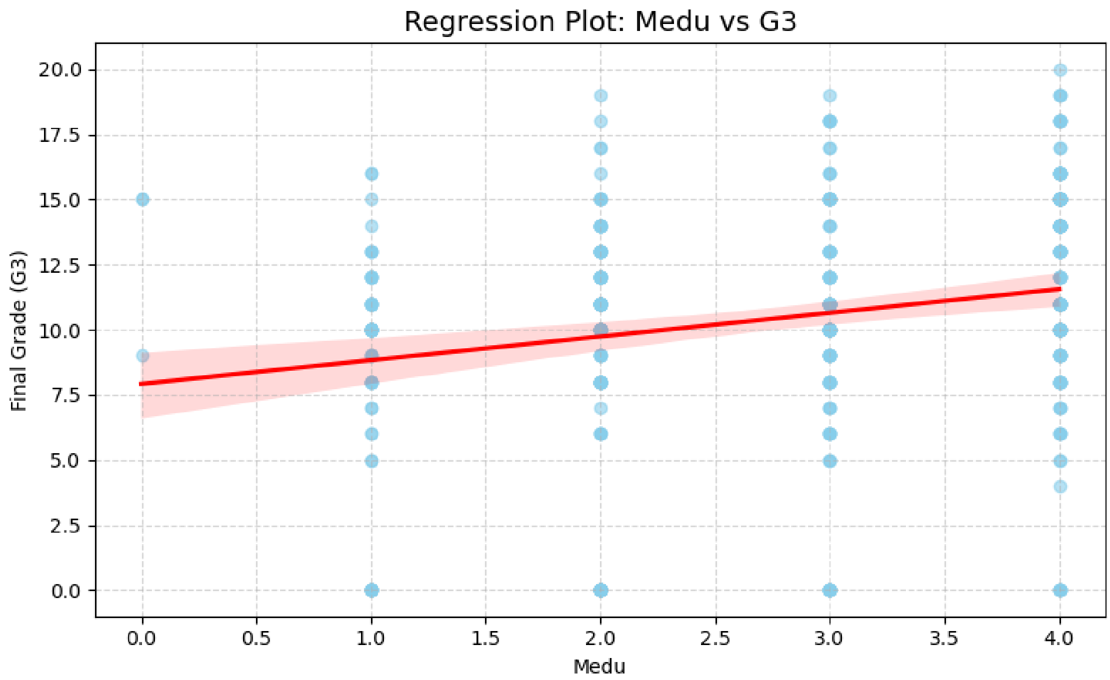
[45]:
```python
#call function comparing a numerical column with G3

reg_plot("age",df_numeric)
```

As we can see : Insight: Younger students (15–17 years) tend to score slightly higher on average.

Older students show more varied and lower performance, possibly due to grade repetition or external responsibilities .

Medu(Mother's education) vs G3 distribution

[54]:
```
#call function comparing a numerical column with G3

reg_plot("Medu",df_numeric)
```

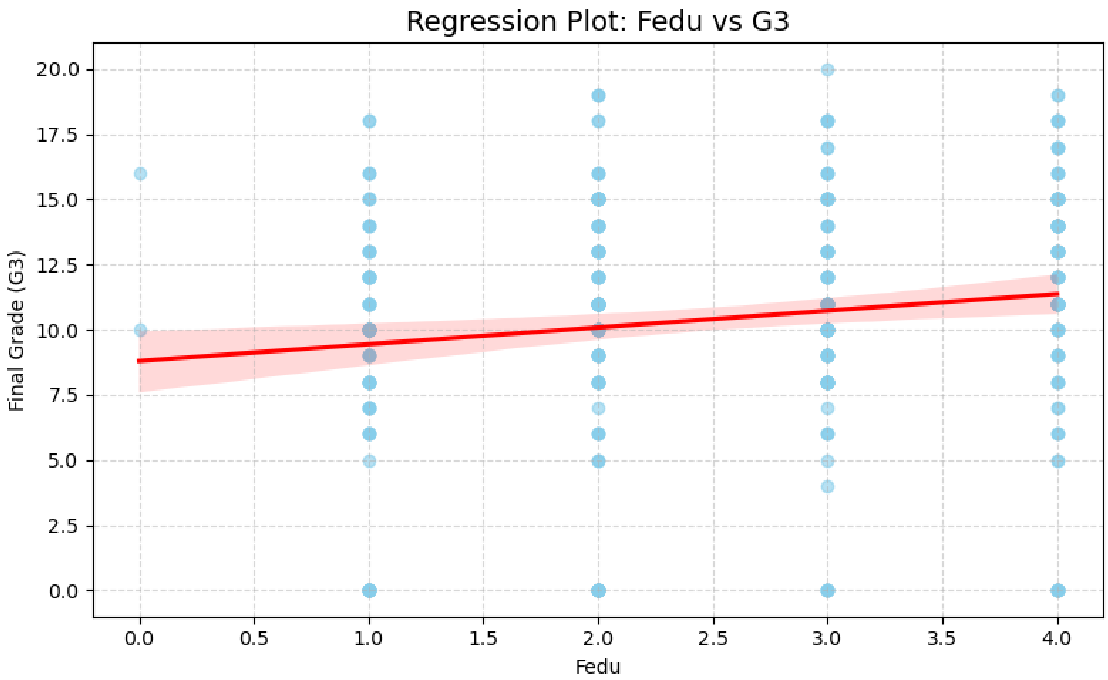


As we can see : Insight: There's a positive trend — students whose mothers have higher education levels generally score better.

Indicates parental education plays a role in academic success.

Fedu (Father's education) vs G3

[63]:
```
#call function comparing a numerical column with G3

reg_plot("Fedu",df_numeric)
```

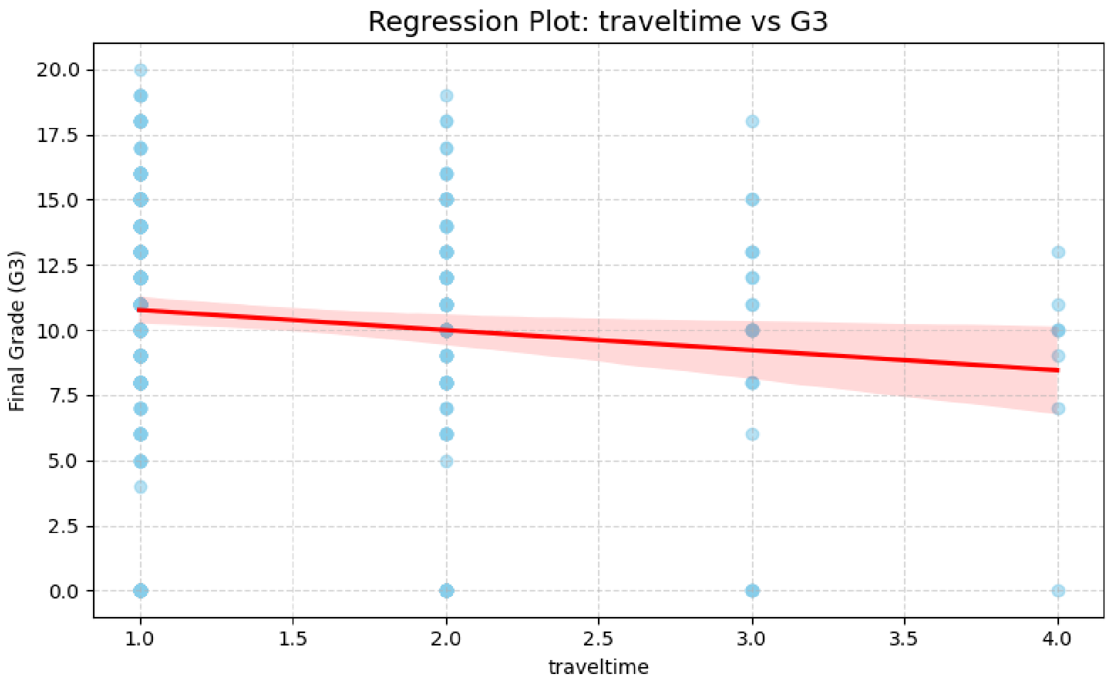


As we can see : Insight: A mild positive correlation is observed.

Father's education does impact performance but not as strongly as mother's education.

Traveltime vs G3

[75]:
```
#call function comparing a numerical column with G3

reg_plot("traveltime",df_numeric)
```

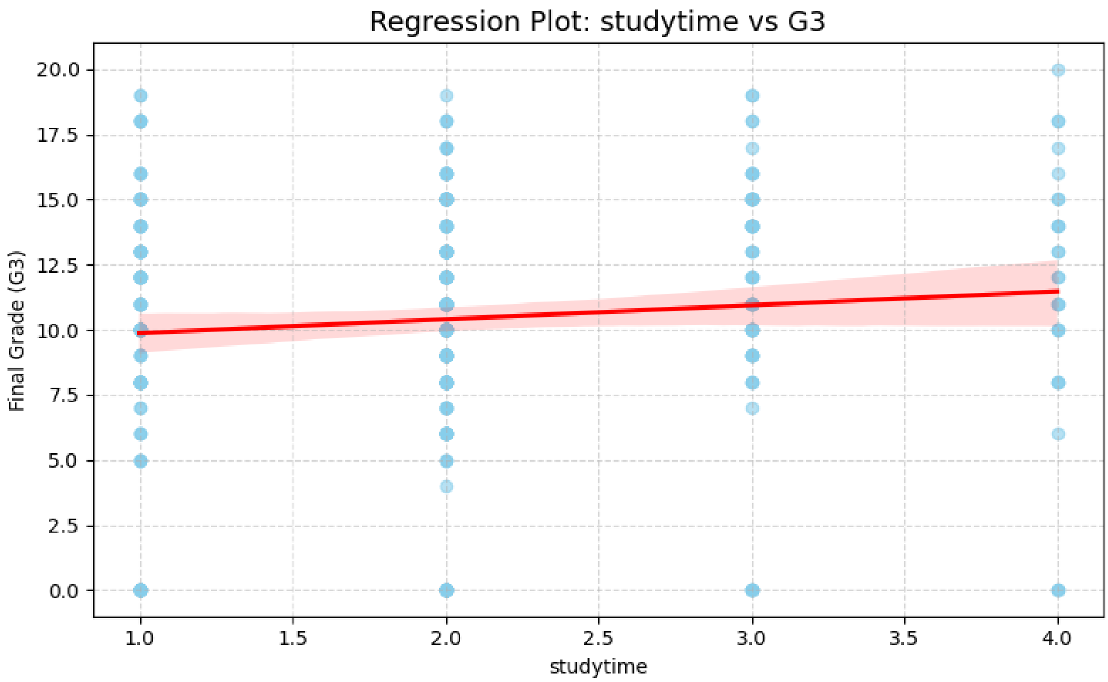


As we can see : Insight: Students with shorter travel times tend to perform better.

Long commutes may reduce time and energy for studies.

Studytime vs G3

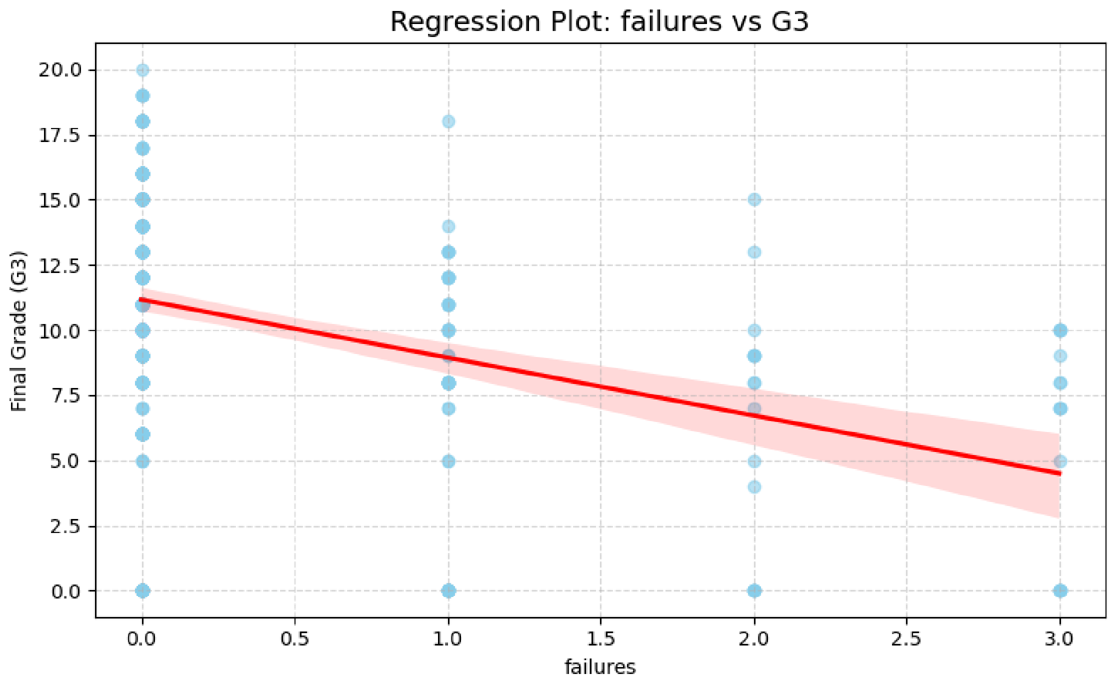
[77]:
```
#call function comparing a numerical column with G3

reg_plot("studytime",df_numeric)
```



As we can see : Insight:Clear positive correlation — more study time leads to higher grades. Emphasizes the importance of consistent study habits.

Failures vs G3

[82]:
```
#call function comparing a numerical column with G3

reg_plot("failures",df_numeric)
```

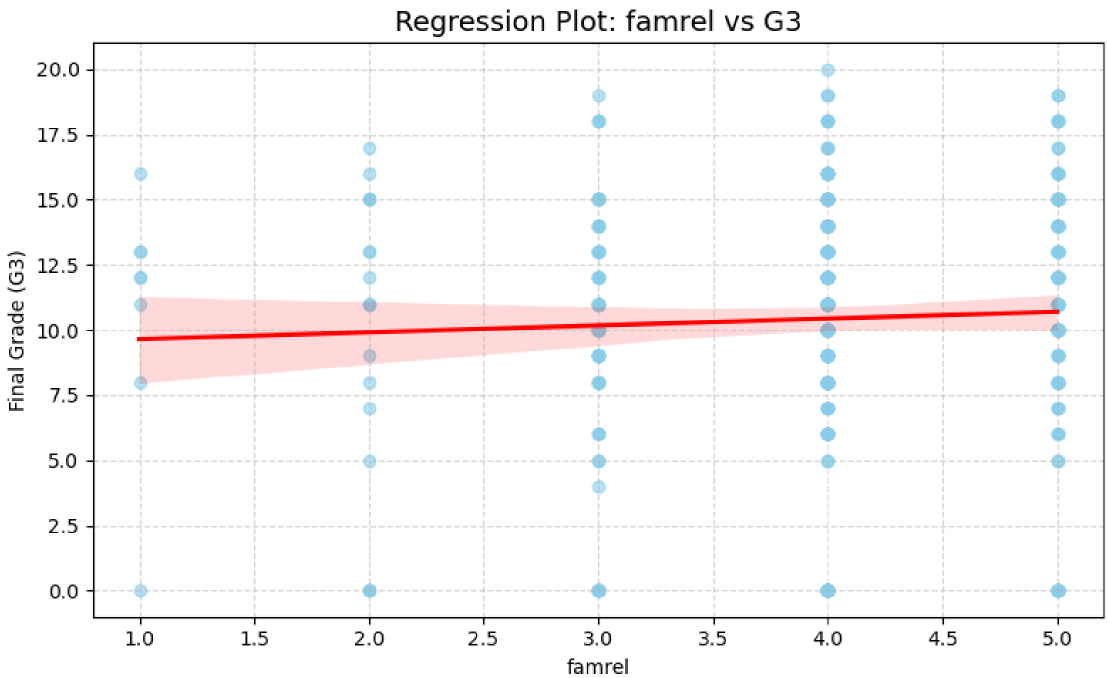


As we can say: Insight: Strong negative correlation — students with more past failures perform worse in G3.

Academic history is a strong predictor of future performance.

Famrel (Family relationship quality) vs G3

[88]:
```
#call function comparing a numerical column with G3

reg_plot("famrel",df_numeric)
```

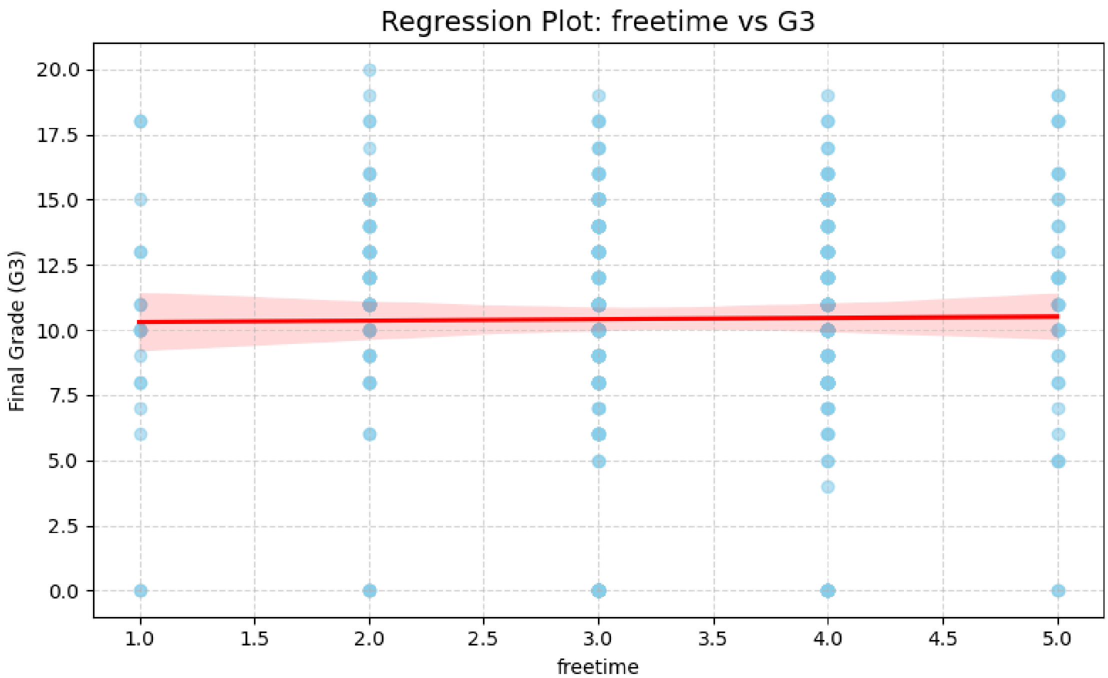


As we can say: Insight: Slight positive trend — better family relationships may support better academic focus and results.

Freetime vs G3

[94]:

```
#call function comparing a numerical column with G3

reg_plot("freetime",df_numeric)
```



As we can say: Insight: No strong trend; however, extremely high free time might correlate with lower grades, suggesting imbalance

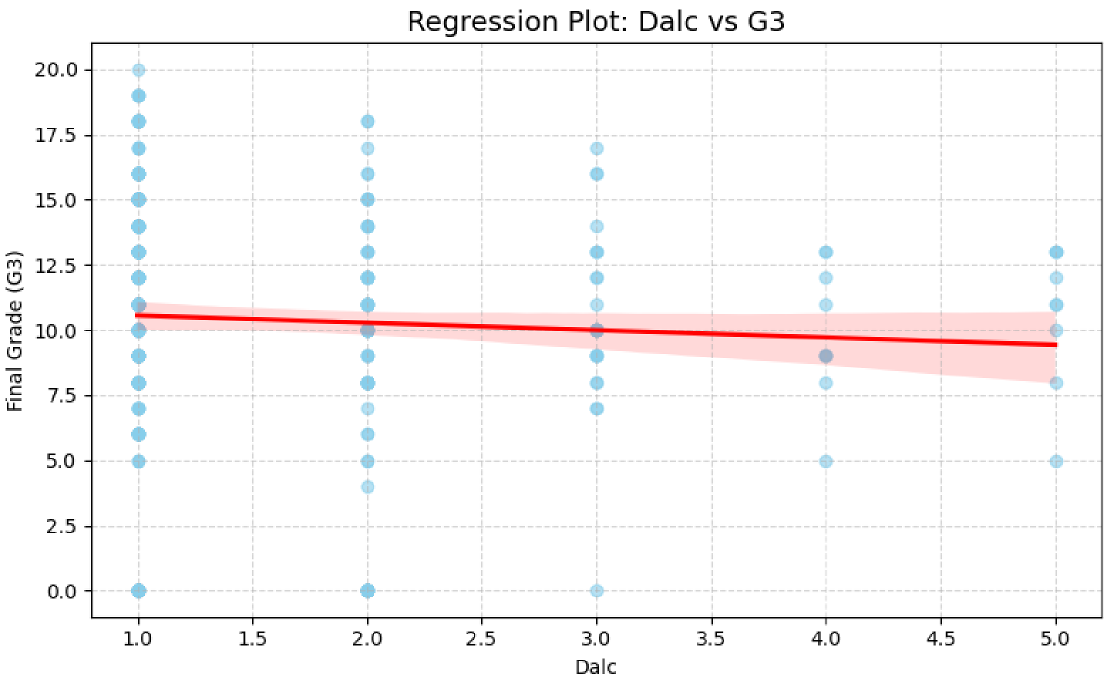
Goout (Going out with friends) vs G3

[102]:
```
#call function comparing a numerical column with G3

reg_plot("goout",df_numeric)
```

As we can say: Insight:Students who frequently go out tend to have slightly lower grades. Suggests time management may be a factor.

Dalc (Workday alcohol consumption) vs G3

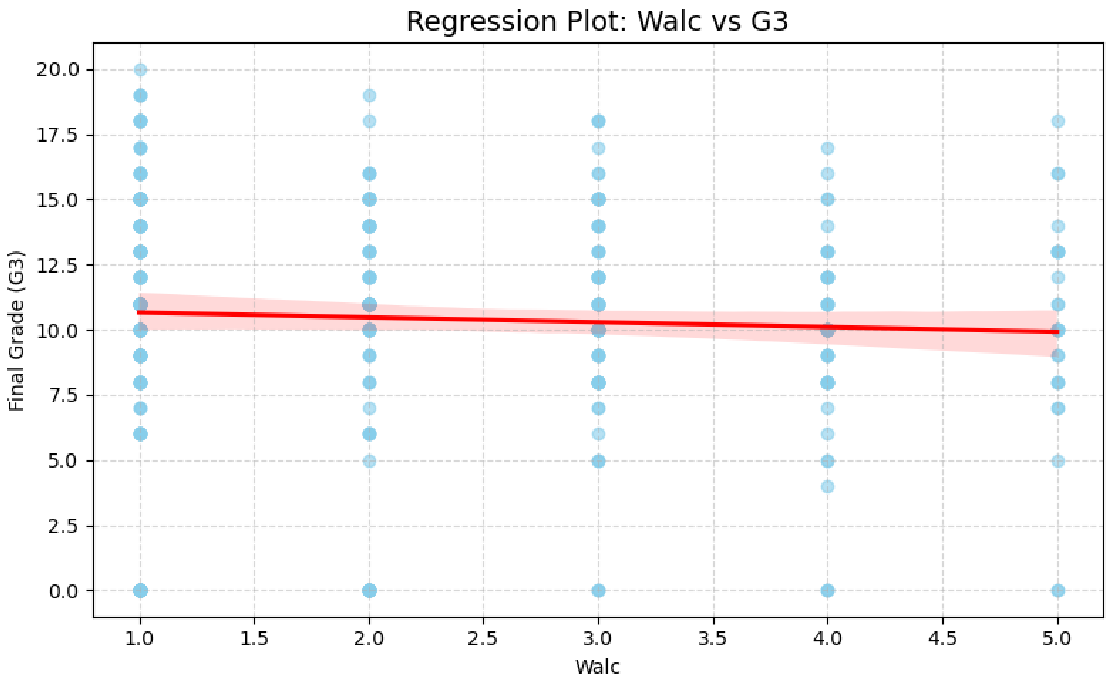
[104]:
```python
#call function comparing a numerical column with G3

reg_plot("Dalc",df_numeric)
```



As we can say: Insight: Clear negative correlation — higher alcohol consumption during weekdays is linked to lower grades.

Walc (Weekend alcohol consumption) vs G3

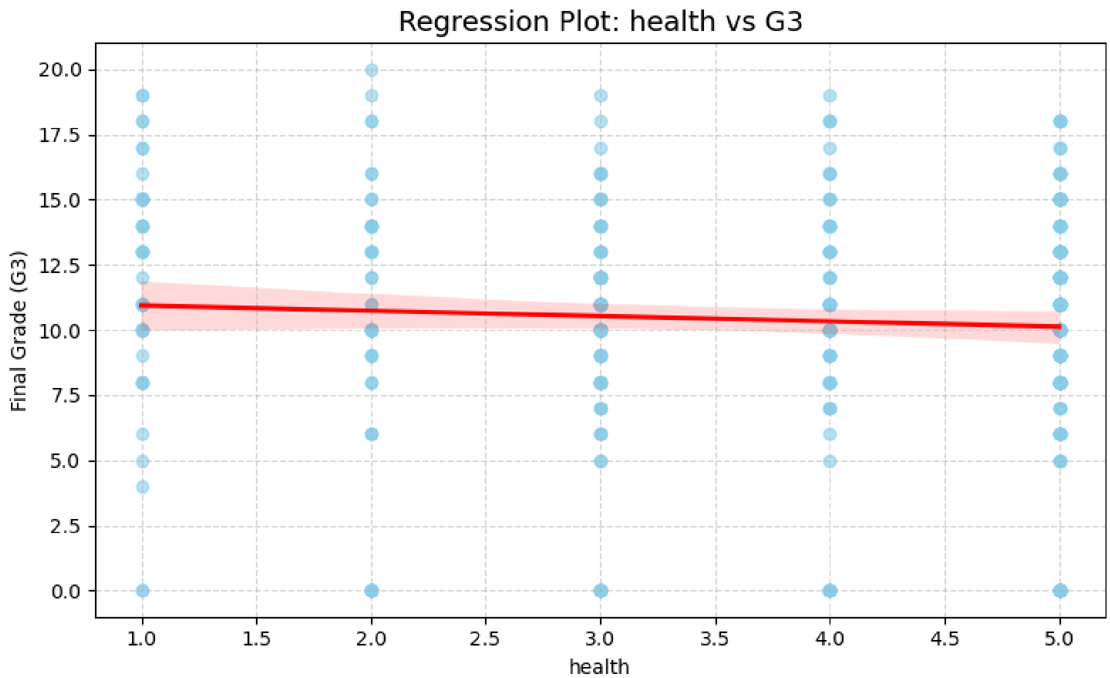
[108]:
```
#call function comparing a numerical column with G3

reg_plot("Walc",df_numeric)
```



As we can say: Insight: Similar to Dalc but less steep; still, heavy weekend drinking correlates with lower performance.

Health vs G3

[113]:
```
#call function comparing a numerical column with G3

reg_plot("health",df_numeric)
```



As we can say: Insight: No strong correlation observed.

Indicates health status (self-reported) doesn't directly impact grades in this dataset.

Absences vs G3

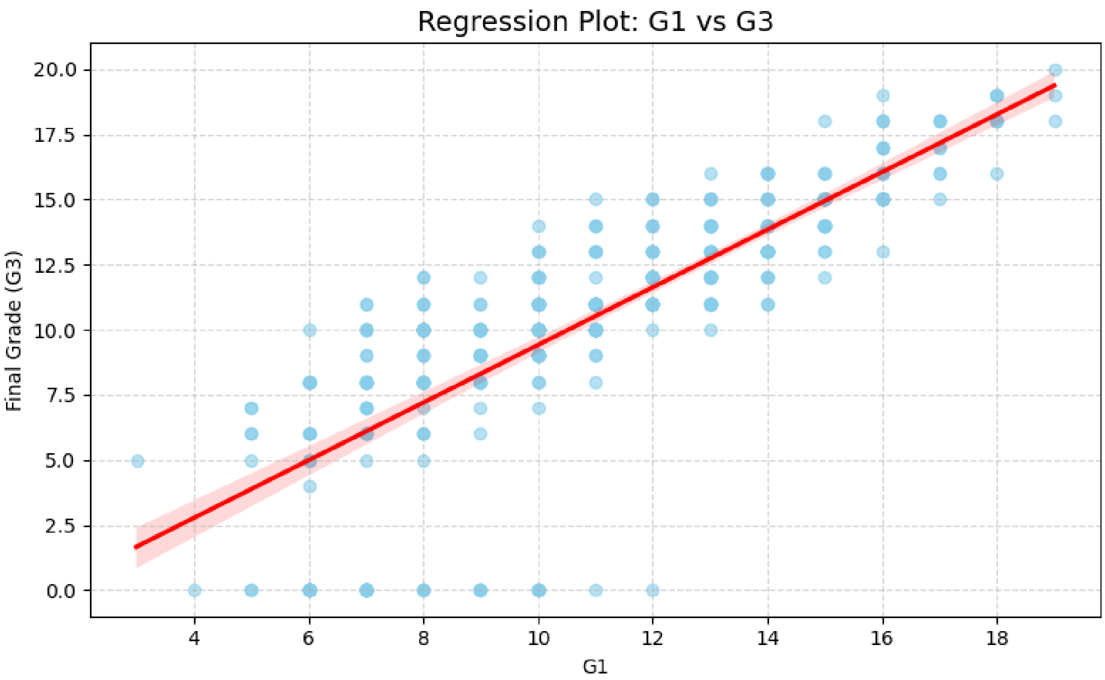
[ ]:
```
#call function comparing a numerical column with G3

reg_plot("Absences",df_numeric)
```

As we can say: Insight: Negative correlation — students with more absences tend to have lower grades.

Attendance is important for academic performance.

G1 vs G3

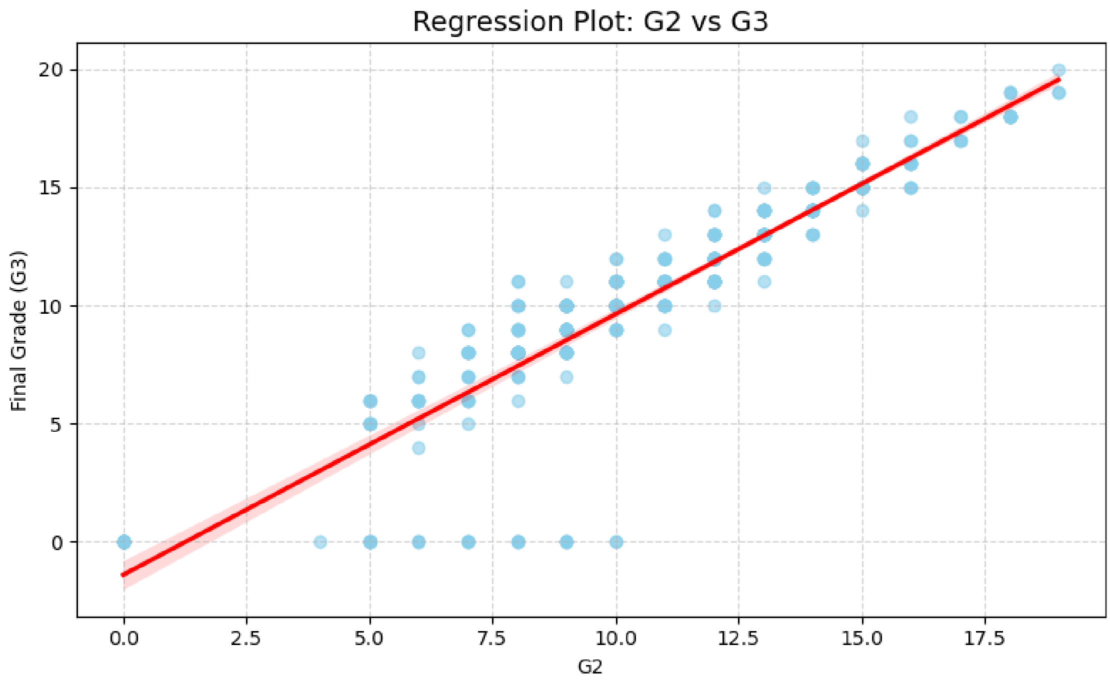
[124]:
```
#call function comparing a numerical column with G3

reg_plot("G1",df_numeric)
```



As we can say: Insight: Strong positive correlation — initial grades are good predictors of final grades.

G2 vs G3

[120]:
```
#call function comparing a numerical column with G3

reg_plot("G2",df_numeric)
```



As we can say: Insight: Very strong positive correlation — second-period grades almost directly align with final grades.

Indicates consistent performance across the year.

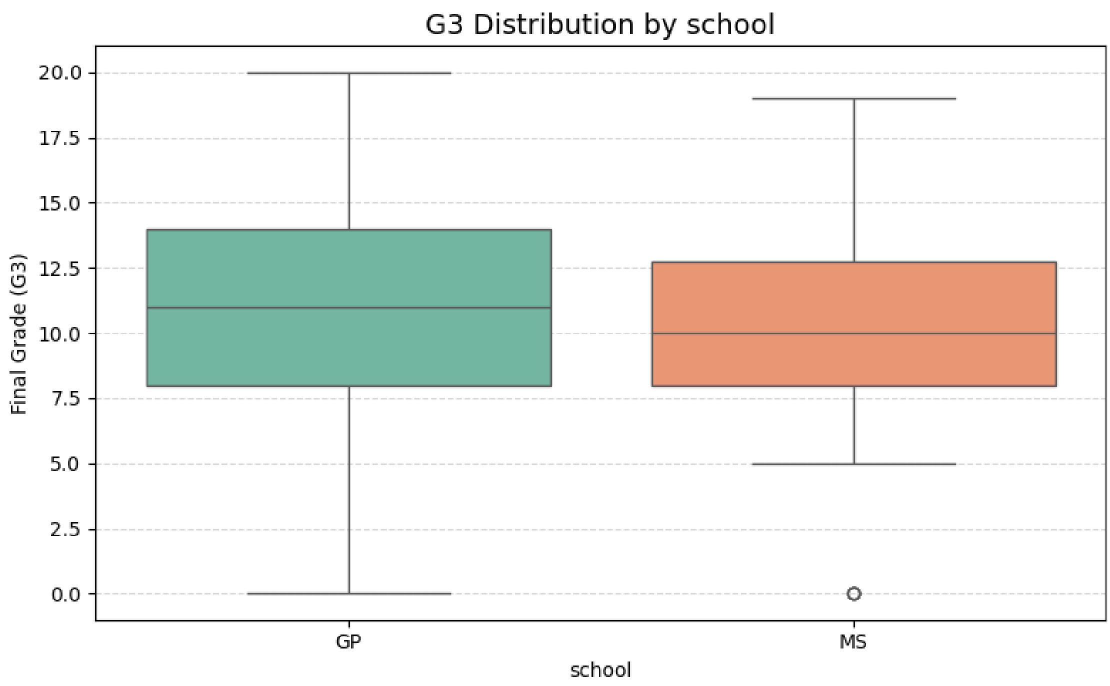
Categorical vs G3 (Regression Plots)

[4]:
```
#Make a Box plot(function) comparing a categorical column with the final grade␣
↪G3

def box_plot(column_name, data, rotation=0, palette='Set2'):
    """
    Parameters:
    - column_name: str, name of the categorical column
    - data: pd.DataFrame, your dataset
    - rotation: int, degree of x-axis label rotation
    - palette: str or list, color palette for boxes

    Output:
    - Boxplot comparing G3 distribution across categories
    """
    plt.figure(figsize=(8, 5))
    sns.boxplot(
        x=column_name,
```

```
        y='G3',
        data=data,
        palette=palette
    )
    plt.title(f'G3 Distribution by {column_name}', fontsize=14)
    plt.xticks(rotation=rotation)
    plt.xlabel(column_name)
    plt.ylabel('Final Grade (G3)')
    plt.grid(axis='y', linestyle='--', alpha=0.5)
    plt.tight_layout()
    plt.legend([],[], frameon=False)  # no legend
    plt.show()
```

School vs G3

[19]:
```
#call function comparing a Categorical column with G3

box_plot("school",data)
```

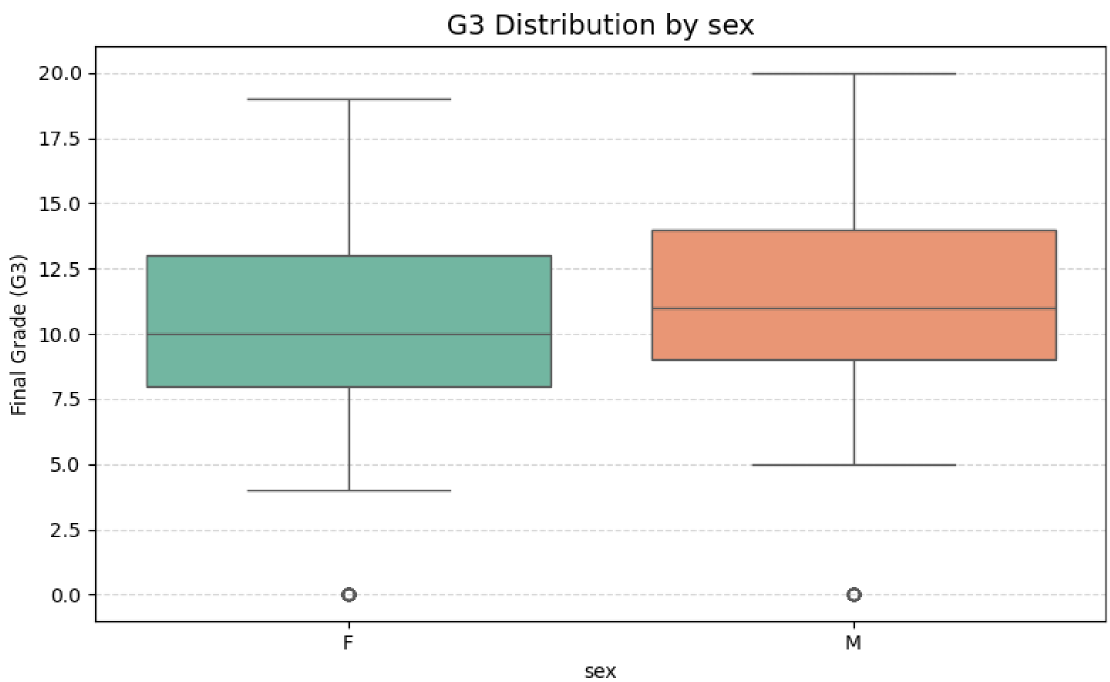


As we can say: Insight: Students from the 'GP' school tend to have slightly higher average G3 scores compared to those from 'MS'.

Sex vs G3

[17]:
```
#call function comparing a Categorical column with G3
```

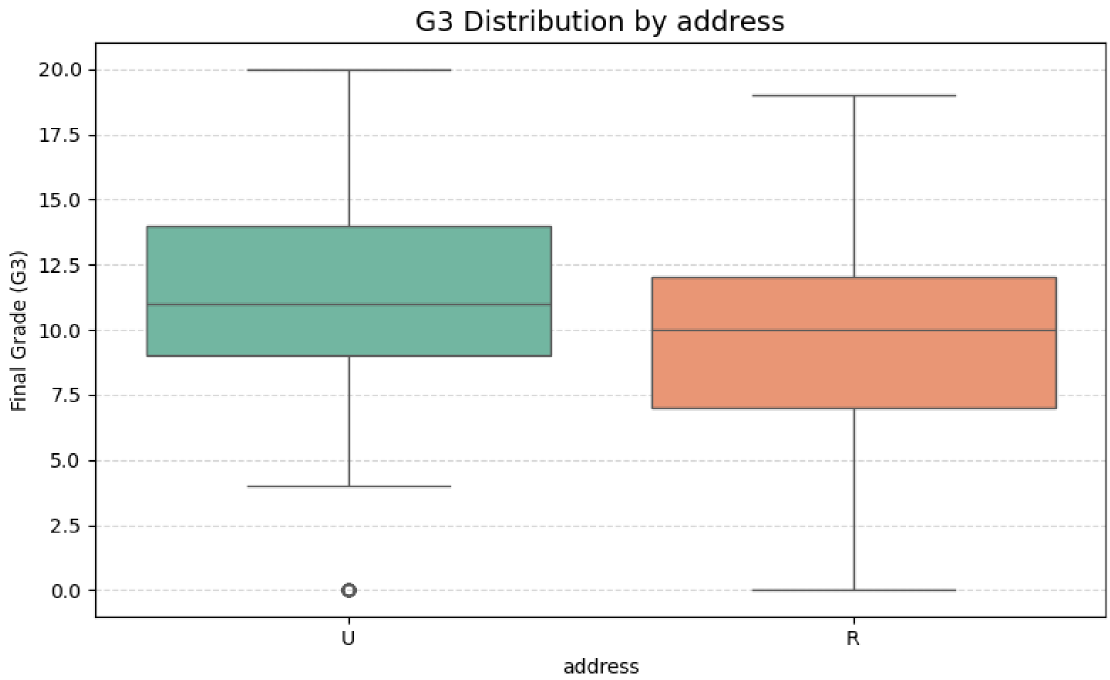
```
box_plot("sex",data)
```



As we can say: Insight: Students from the 'GP' school tend to have slightly higher average G3 scores compared to those from 'MS'.

** Address (address) vs G3**

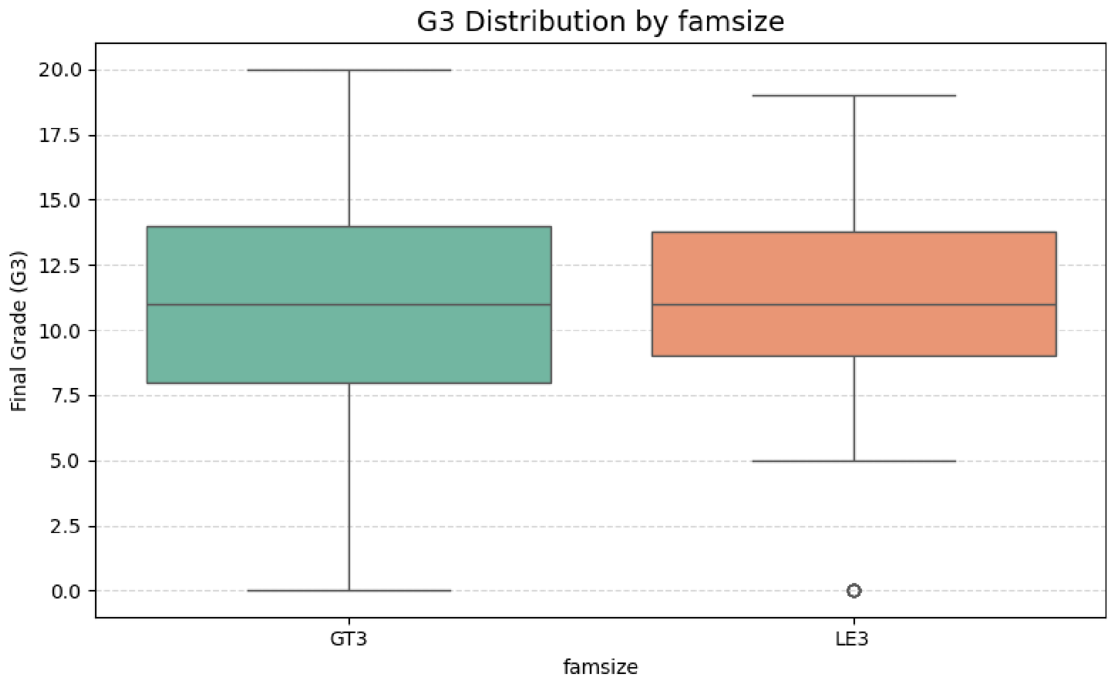
[25]:
```
#call function comparing a Categorical column with G3

box_plot("address",data)
```



As we can say: Insight: Students living in urban areas ('U') tend to perform better than those in rural areas ('R').

Family Size (famsize) vs G3

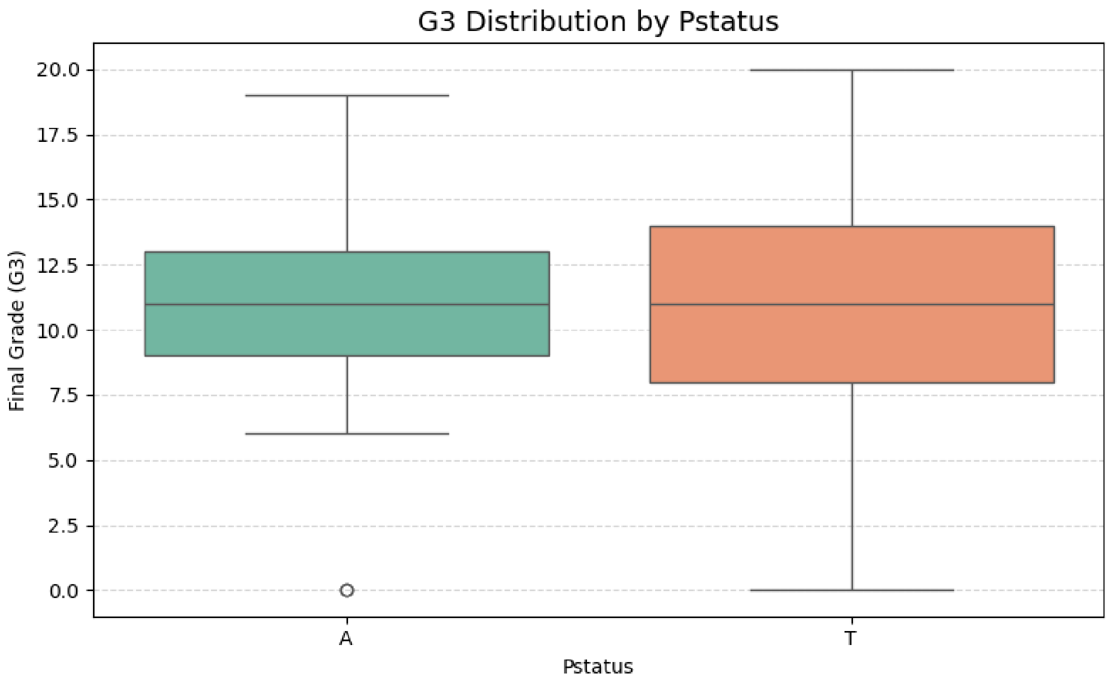
[29]:
```
#call function comparing a Categorical column with G3

box_plot("famsize",data)
```



As we can say: Insight: Students from smaller families ('LE3') show slightly higher G3 scores.

Parental Cohabitation Status (Pstatus) vs G3

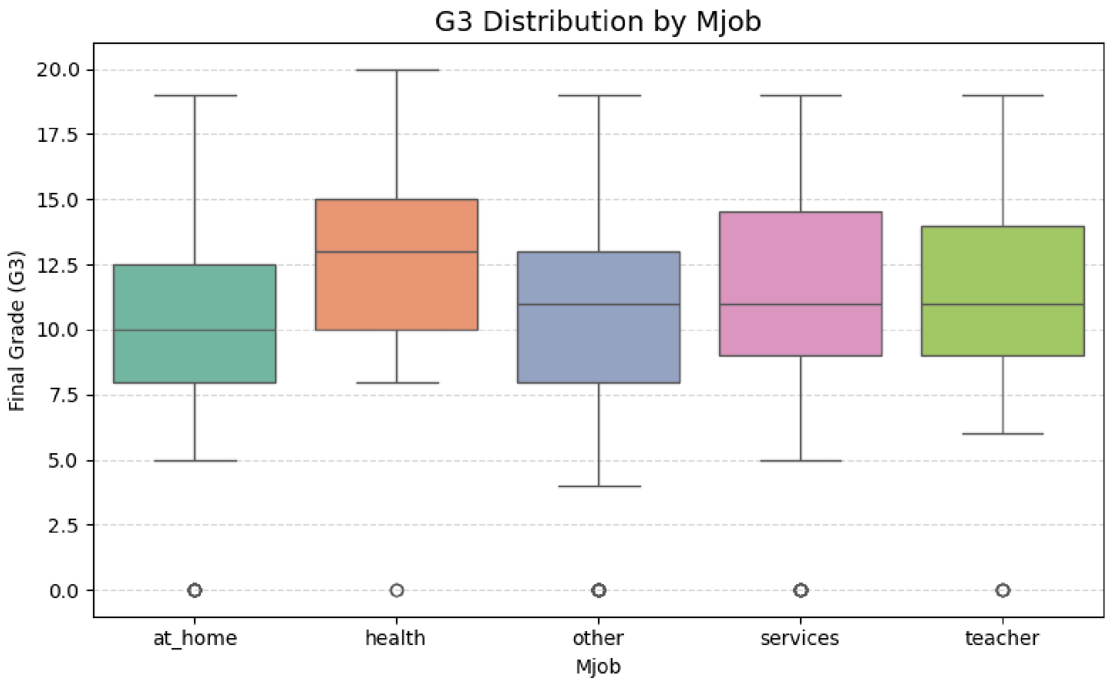
[33]:
```
#call function comparing a Categorical column with G3

box_plot("Pstatus",data)
```



As we can say: Insight: Students whose parents live together ('T') have marginally higher G3 scores.

Mother's Job (Mjob) vs G3

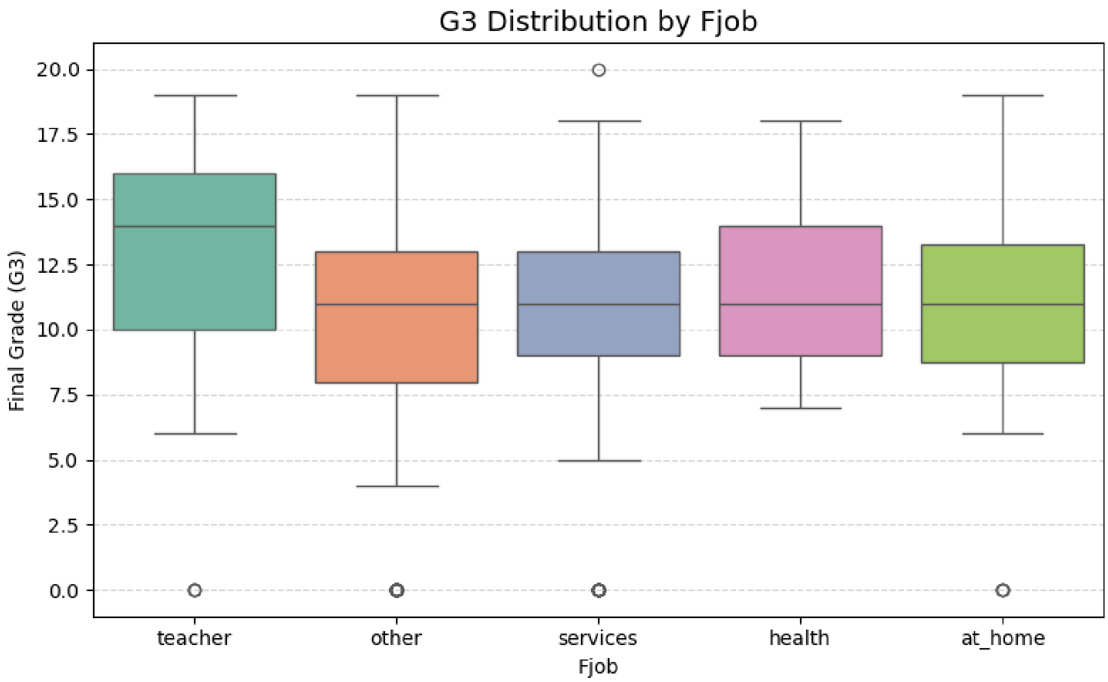
[37]:
```
#call function comparing a Categorical column with G3

box_plot("Mjob",data)
```



As we can say: Insight: Students whose mothers work in education or health sectors tend to have higher G3 scores.

Father's Job (Fjob) vs G3

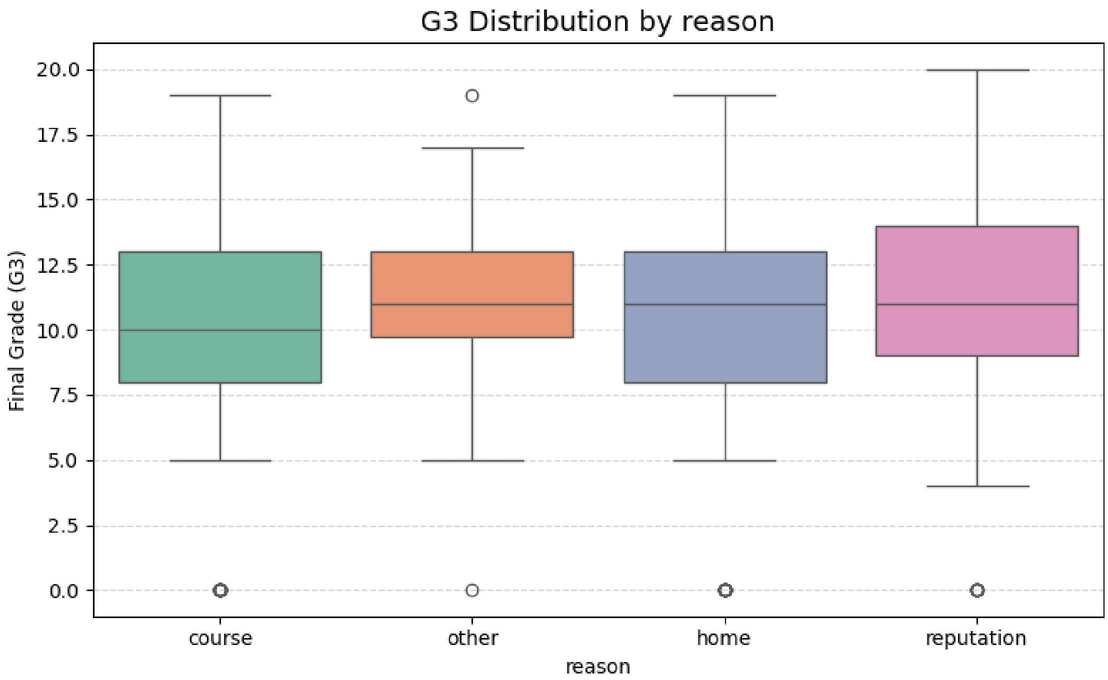
[14]:
```
#call function comparing a Categorical column with G3

box_plot("Fjob",data)
```



As we can say: Insight: Similar to mothers, students with fathers in education or health sectors often perform better.

Reason for Choosing School(reason) vs G3

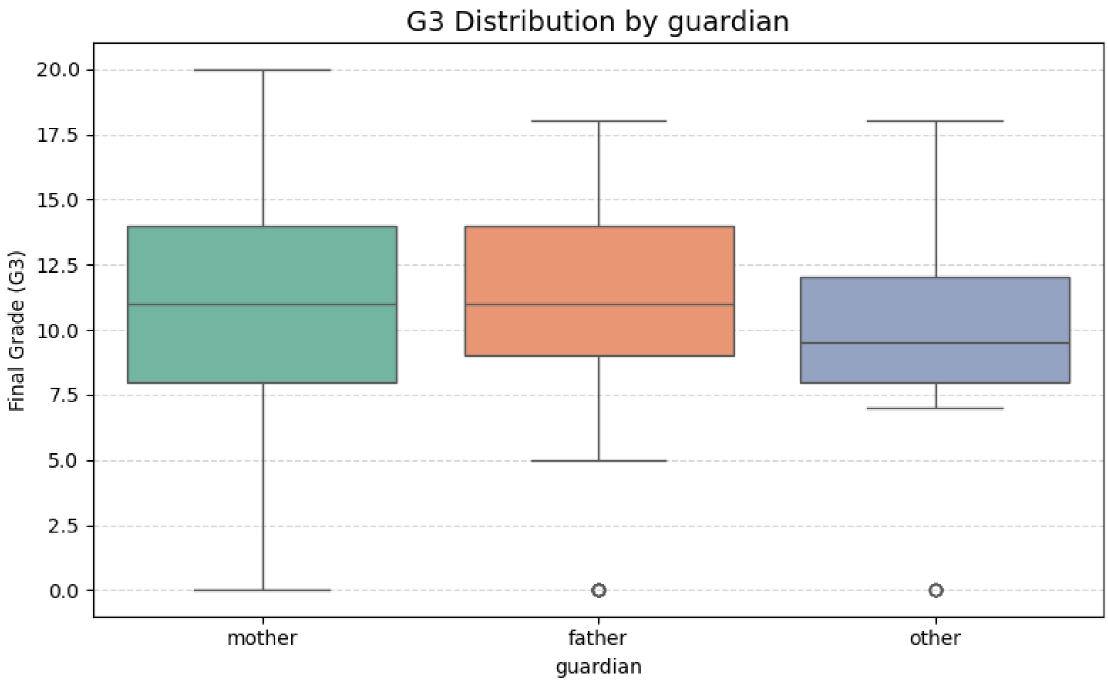
[16]:
```
#call function comparing a Categorical column with G3

box_plot("reason",data)
```



As we can say: Insight: Students who chose the school for its reputation tend to have higher G3 scores.

Guardian vs G3

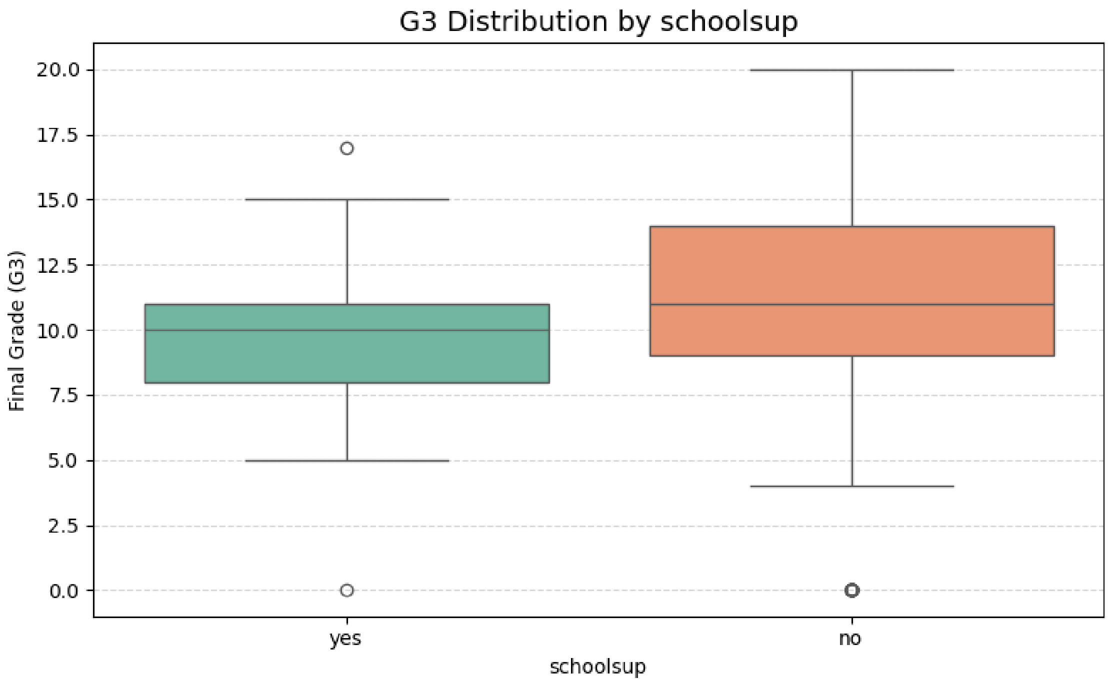
[22]:
```
#call function comparing a Categorical column with G3

box_plot("guardian",data)
```



As we can say: Insight: Students under the guardianship of their mother or father perform slightly better than those with other guardians.

School Support (schoolsup)

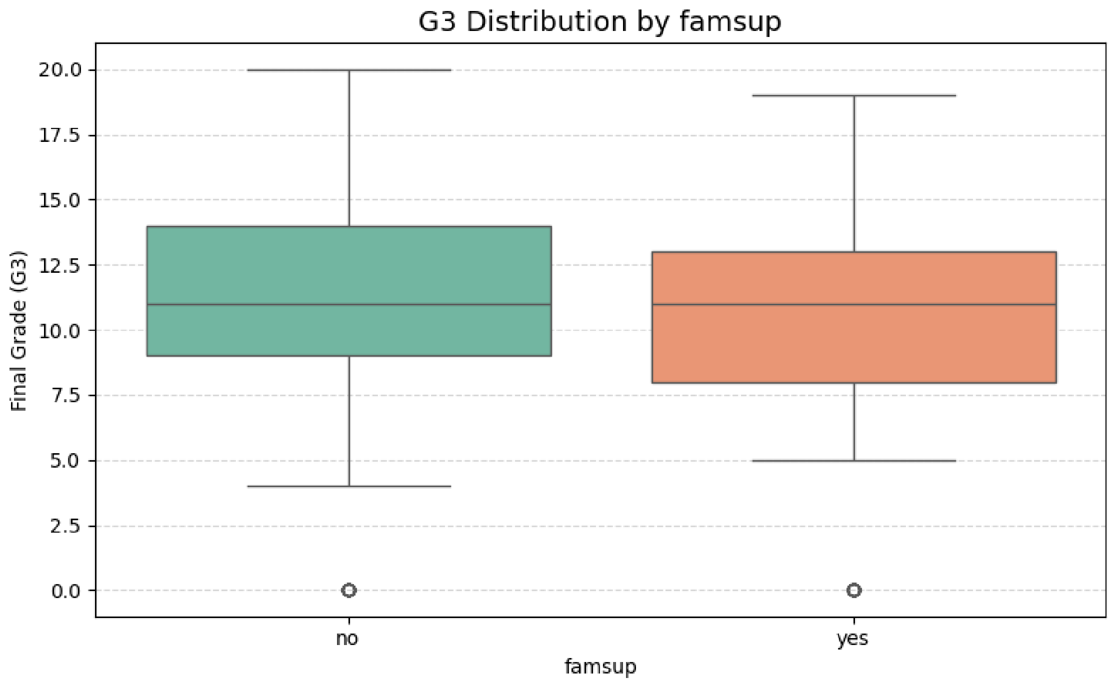
[28]:
```
#call function comparing a Categorical column with G3

box_plot("schoolsup",data)
```



As we can say: insight: Students receiving extra educational support ('yes') often have lower G3 scores.

Family Support (famsup)

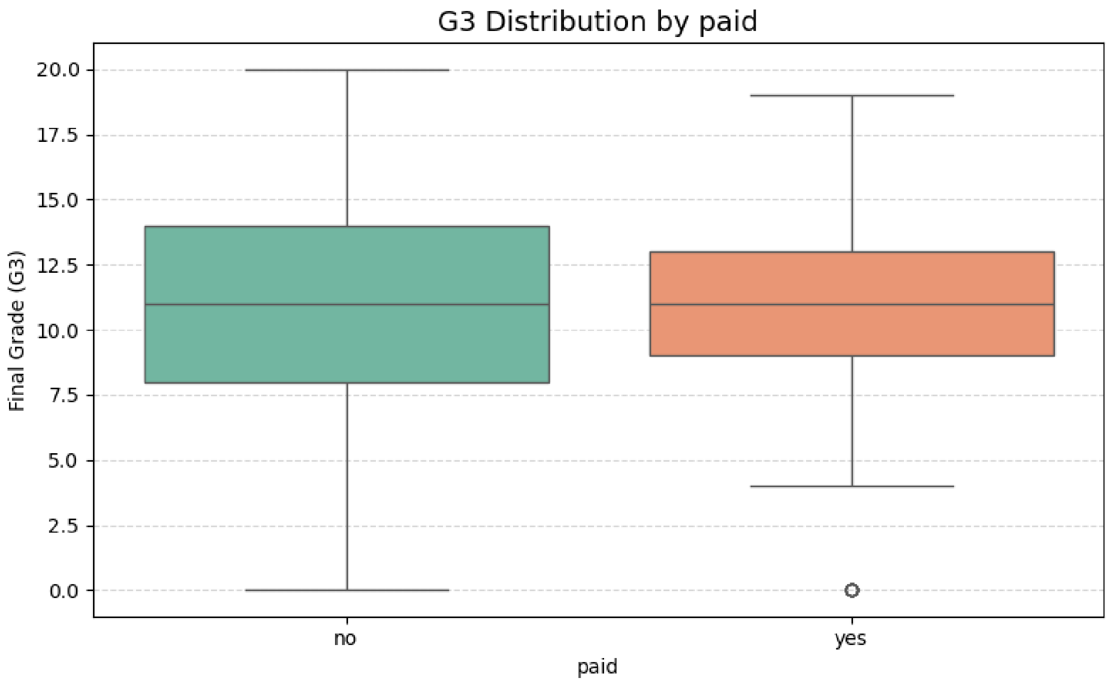
[31]:
```
#call function comparing a Categorical column with G3

box_plot("famsup",data)
```



As we can say: Insight: Students with family educational support ('yes') show slightly higher G3 scores.

Paid Classes (paid)

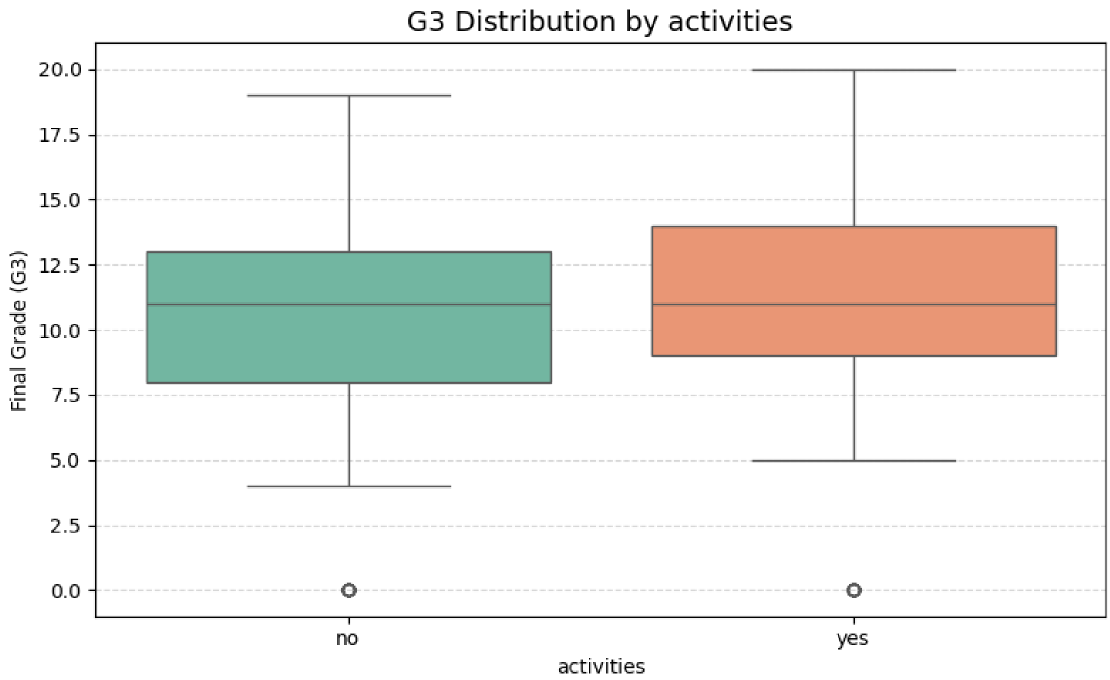
[35]:
```
#call function comparing a Categorical column with G3

box_plot("paid",data)
```



As we can say: Insight: Students attending extra paid classes ('yes') have marginally higher G3 scores

Extracurricular Activities (activities)vs G3

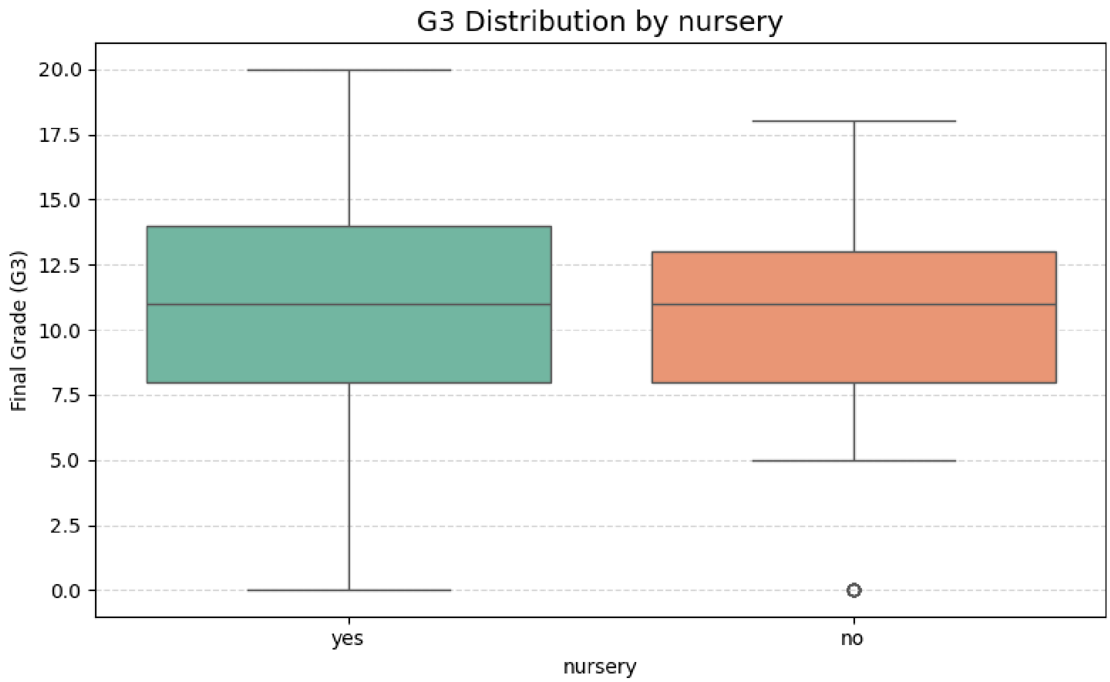
[39]:
```
#call function comparing a Categorical column with G3

box_plot("activities",data)
```



As we can say: Insight: Participation in extracurricular activities ('yes') correlates with slightly higher G3 scores.

Nursery School Attendance (nursery) vs G3

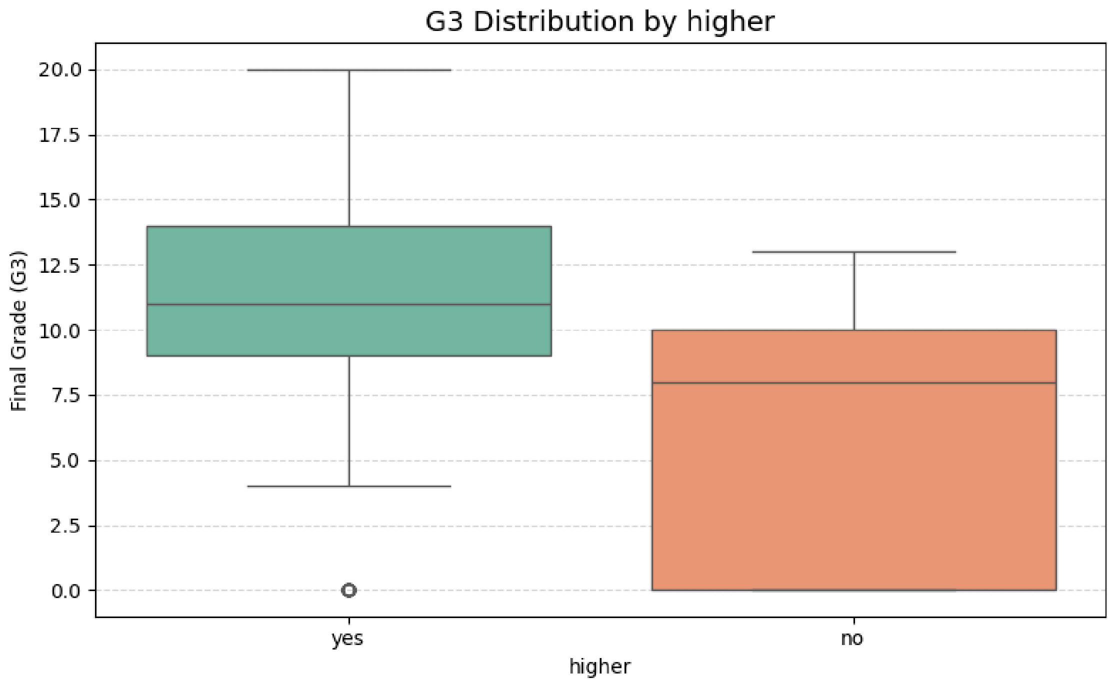
[42]:
```
#call function comparing a Categorical column with G3

box_plot("nursery",data)
```



As we can say: Insight: Students who attended nursery school ('yes') tend to perform better.

Desire for Higher Education (higher) vs G3

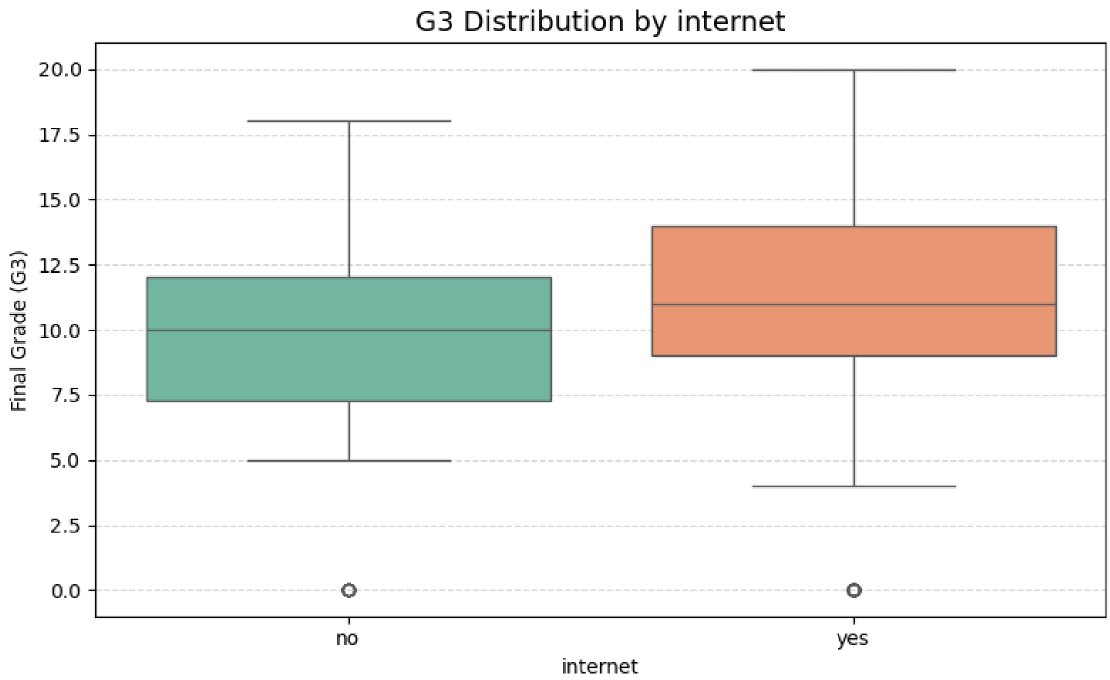
[49]:
```
#call function comparing a Categorical column with G3

box_plot("higher",data)
```



As we can say: Insight: Students aspiring for higher education ('yes') have significantly higher G3 scores.

Internet Access at Home (internet) vs G3

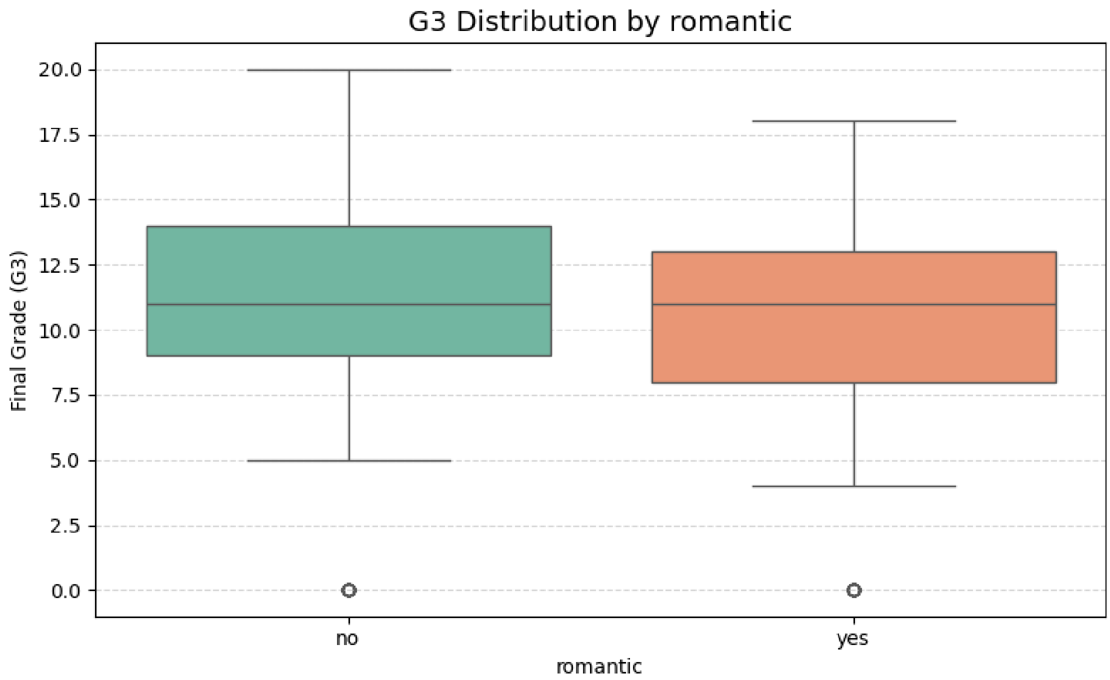
[53]:
```
#call function comparing a Categorical column with G3

box_plot("internet",data)
```



As we can say: Insight: Access to the internet at home ('yes') is associated with slightly higher G3 scores.

Romantic Relationship (romantic) vs G3

[61]:
```
#call function comparing a Categorical column with G3

box_plot("romantic",data)
```



As we can say: Insight: Students not in a romantic relationship ('no') tend to have higher G3 scores.

Data Preprocessing And Modeling

[70]:
```
# Import necessary libraries for data preprocessing, feature selection, model␣
↪training and evaluation
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score

# Create a copy of the original data to avoid modifying it

data_encoded = data.copy()

# Initialize a dictionary to store label encoders for potential inverse␣
↪transform later
label_encoders = {}

# Encode all categorical (object) columns to numeric values
```

```
for col in data_encoded.select_dtypes(include='object').columns:
    le = LabelEncoder()
    data_encoded[col] = le.fit_transform(data_encoded[col])
    label_encoders[col] = le  # store encoders if needed later

# Separate features (X) and target variable (y)
X = data_encoded.drop("G3", axis=1)  # G3 appears to be the target variable␣
 ↪(final grade)
y = data_encoded["G3"]

# Perform feature selection to identify the most important features
# Using f_regression which is suitable for regression problems
selector = SelectKBest(score_func=f_regression, k="all")  # select all features
X_selected = selector.fit_transform(X, y)

# Split data into training (80%) and testing (20%) sets
# Setting random_state ensures reproducibility
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.
 ↪2, random_state=42)
```

Train and evaluate multiple regression models to compare performance

[79]:
```
# Model 1: Linear Regression - a simple baseline model
lr = LinearRegression()
lr.fit(X_train, y_train)  # Train the model
y_pred_lr = lr.predict(X_test)  # Make predictions
# Evaluate model performance using R² (coefficient of determination) and RMSE
print("\nLinear Regression:")
print("R2 Score:", r2_score(y_test, y_pred_lr))  # Higher is better, max is 1.0
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))  # Lower is␣
 ↪better
```

```
Linear Regression:
R2 Score: 0.75457778550435
RMSE: 2.2432998258963828
```

[74]:
```
# Model 2: Random Forest Regressor - an ensemble method that often performs well
rf = RandomForestRegressor(random_state=42)  # Set random_state for␣
 ↪reproducibility
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("\nRandom Forest:")
print("R2 Score:", r2_score(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
```

```
Random Forest:
```

```
R2 Score: 0.8299569015097052
RMSE: 1.867281920908543
```

[76]:
```
# Model 3: Support Vector Regressor - good for non-linear relationships
svr = SVR()  # Using default parameters
svr.fit(X_train, y_train)
y_pred_svr = svr.predict(X_test)
print("\nSupport Vector Regressor:")
print("R2 Score:", r2_score(y_test, y_pred_svr))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_svr)))
```

```
Support Vector Regressor:
R2 Score: 0.7939368724306656
RMSE: 2.055561765312519
```

Model Evaluation Used R^2 Score and RMSE:

Random Forest: $R^2 = 0.83$, RMSE = 1.87 (Best)

Linear Regression: $R^2 = 0.75$, RMSE = 2.24

SVR: $R^2 = 0.79$, RMSE = 2.06

Conclusion & Observations:

Random Forest Regressor performed the best among all models, achieving the highest R^2 score (0.83) and the lowest RMSE (1.87).

Insight: This suggests that Random Forest can capture complex, non-linear relationships in the student performance data effectively.

SVR also performed well, indicating that kernel-based methods can generalize decently with a bit more tuning.

Linear Regression had the lowest performance, meaning that a simple linear model doesn't capture all the dependencies and interactions in the data. Random Forest Regressor gave the best prediction accuracy.

Features like previous grades (G1, G2), study time, and absences were strong predictors.

The project highlights the importance of family, academic support, and student habits in performance outcomes.

Potential Applications -Early warning systems in schools

-Personalized academic support

-Policy recommendations for educators