1. Load Dataset

In [1]:
```
# Upload Dataset (student (1).zip or the .xls files)
from google.colab import files
import pandas as pd
import zipfile

print(" Upload your student (1).zip file OR student-mat.xls directly:")
uploaded = files.upload()

filename = list(uploaded.keys())[0]
print("Uploaded:", filename)

# If a ZIP file was uploaded
if filename.endswith(".zip"):
    with zipfile.ZipFile(filename, 'r') as z:
        print("ZIP Contents:", z.namelist())
        # Load CSV file inside the ZIP, specifying the semicolon delimiter
        with z.open('student-mat.csv') as f:
            df = pd.read_csv(f, sep=';')
else:
    # If the user directly uploads student-mat.xls
    df = pd.read_excel(filename)

df.head()
```

```
Upload your student (1).zip file OR student-mat.xls directly:
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

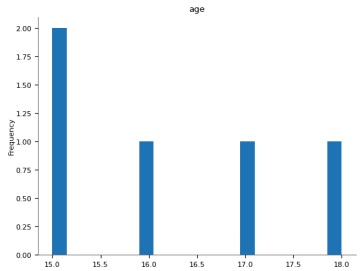
```
Saving student (1).zip to student (1).zip
Uploaded: student (1).zip
ZIP Contents: ['student-mat.csv', 'student-por.csv', 'student-merge.R', 'studen
t.txt']
```

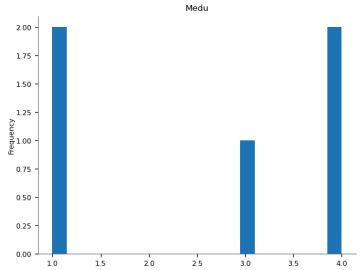
Out[1]:

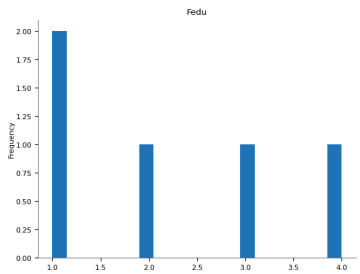
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob
0	GP	F	18	U	GT3	A	4	4	at_home	teacher
1	GP	F	17	U	GT3	T	1	1	at_home	other
2	GP	F	15	U	LE3	T	1	1	at_home	other
3	GP	F	15	U	GT3	T	4	2	health	services
4	GP	F	16	U	GT3	T	3	3	other	other

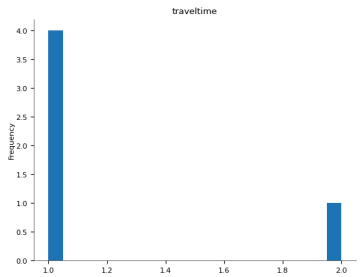
5 rows × 33 columns

Distributions

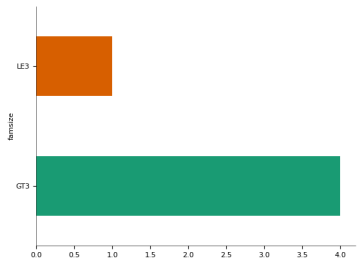


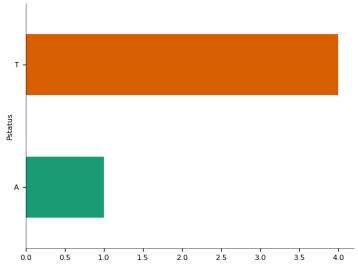


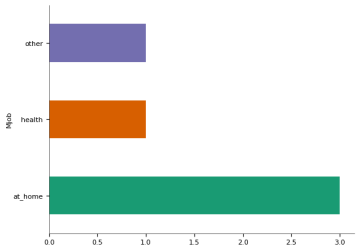


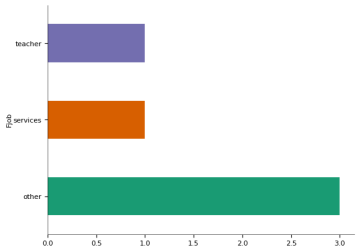


Categorical distributions

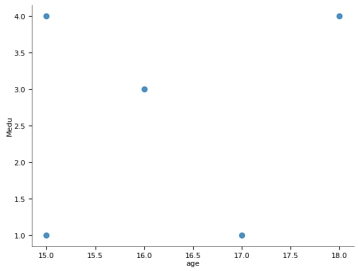


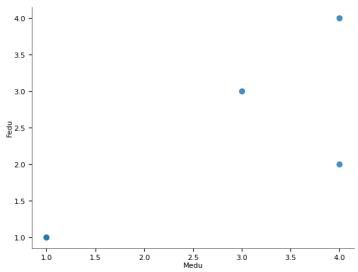


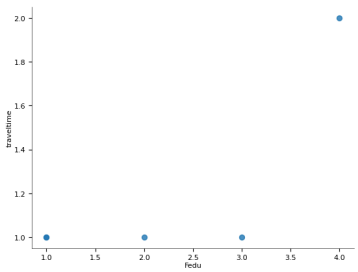


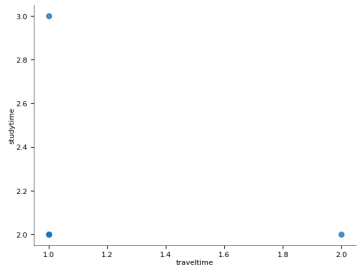


2-d distributions

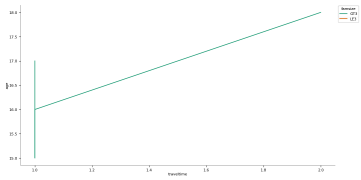


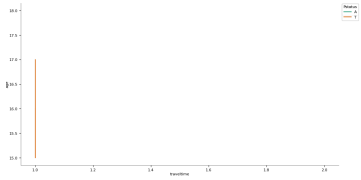


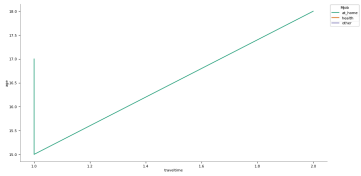


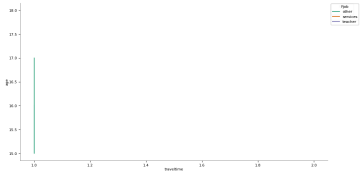


Time series

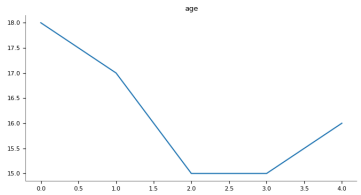


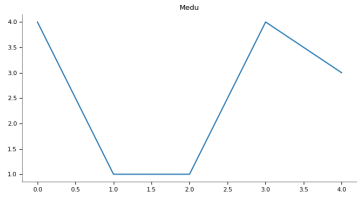


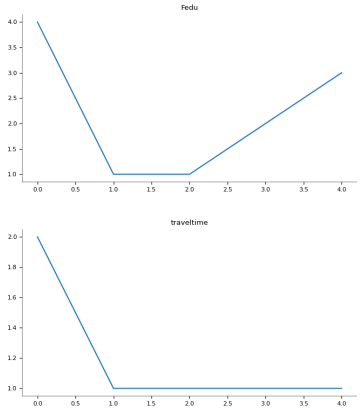




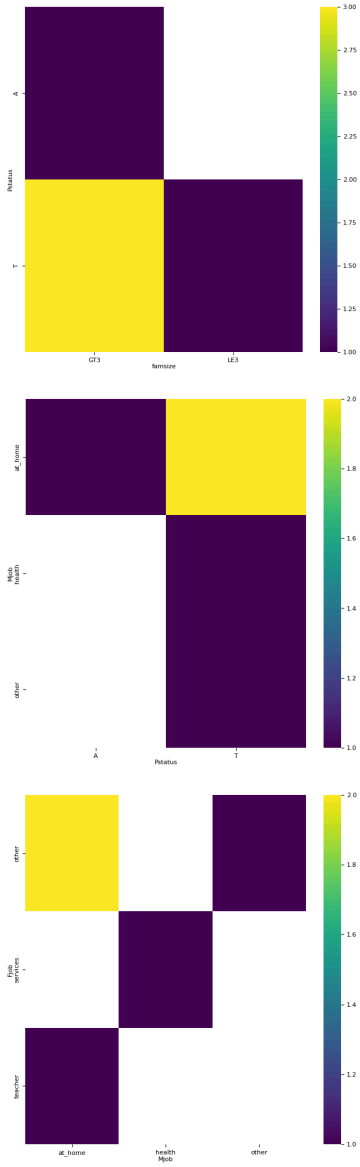
Values

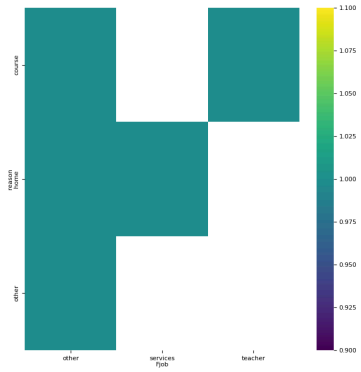






2-d categorical distributions

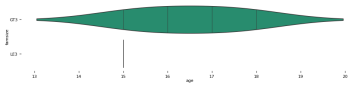




Faceted distributions

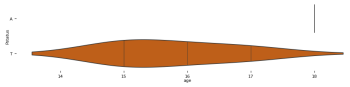
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.
```



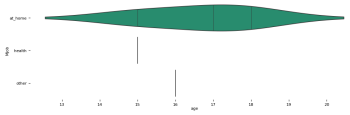

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.
```




```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.
```




```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.
```

2. Explore & Clean Data:

▸Checking missing values

▸Removing duplicates

▸Inspecting shape

▸Inspecting dtypes

In [ ]:
```
print("Missing values:\n", df.isnull().sum())

before = df.shape[0]
df = df.drop_duplicates()
after = df.shape[0]
print(f"Removed duplicates: {before-after}")

print("\nShape:", df.shape)
print("\nData Types:\n", df.dtypes)
```

```
Missing values:
 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
Removed duplicates: 0

Shape: (395, 33)

Data Types:
 school        object
sex            object
age             int64
address        object
famsize        object
Pstatus        object
Medu            int64
Fedu            int64
Mjob           object
Fjob           object
reason         object
guardian       object
traveltime      int64
studytime       int64
```

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

3. Analysis Questions

- Average final grade (G3)
- How many students scored above 15
- Correlation between study time and final grade
- Which gender performs better on average

In [ ]:
```
print("Average G3:", df['G3'].mean())
print("Count G3 > 15:", (df['G3'] > 15).sum())
print("Correlation studytime vs G3:", df['studytime'].corr(df['G3']))

print("\nAverage G3 by gender:")
print(df.groupby('sex')['G3'].mean())
```

```
Average G3: 10.415189873417722
Count G3 > 15: 40
Correlation studytime vs G3: 0.09781968965319626

Average G3 by gender:
sex
F     9.966346
M    10.914439
Name: G3, dtype: float64
```

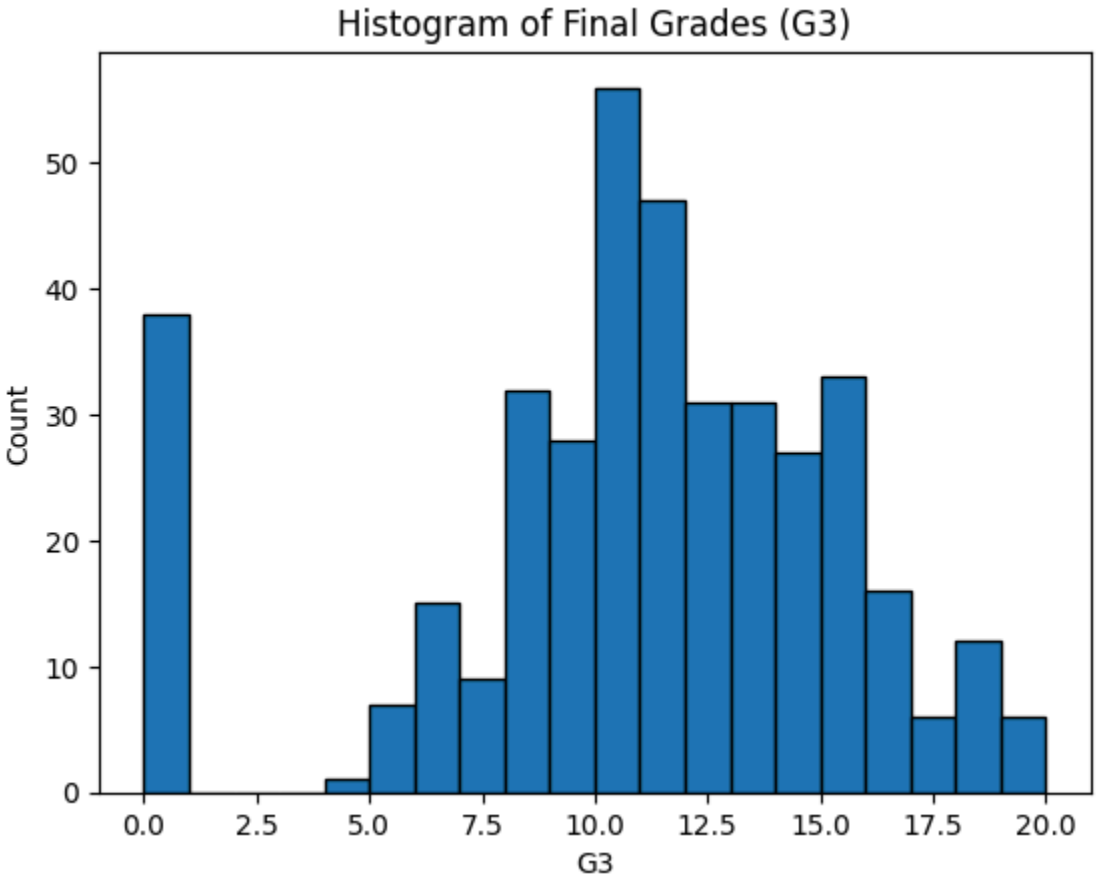
4. Visualizations

- Histogram of final grades

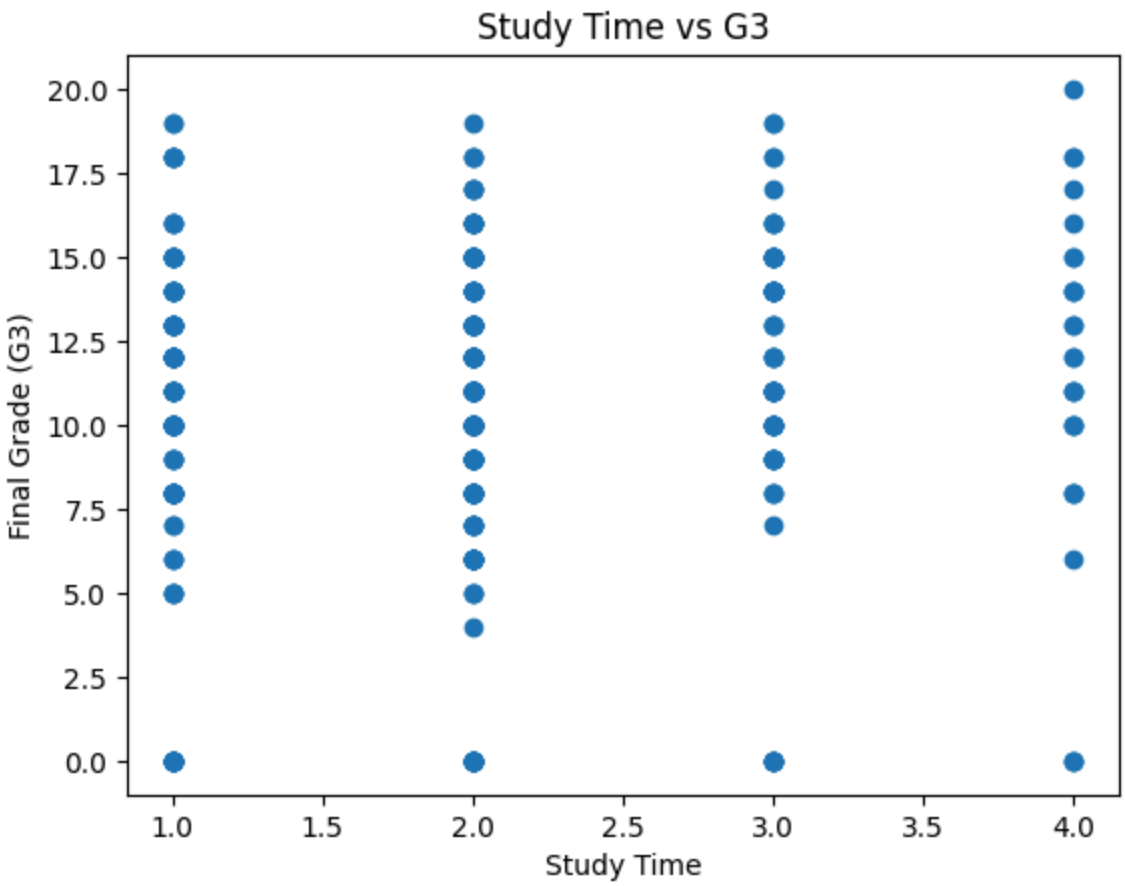
- Scatter plot of study time vs final grade
- Bar chart of average score by gender

In [ ]:
```
import matplotlib.pyplot as plt

plt.hist(df['G3'], bins=range(0,21), edgecolor='black')
plt.title('Histogram of Final Grades (G3)')
plt.xlabel('G3')
plt.ylabel('Count')
plt.show()
```



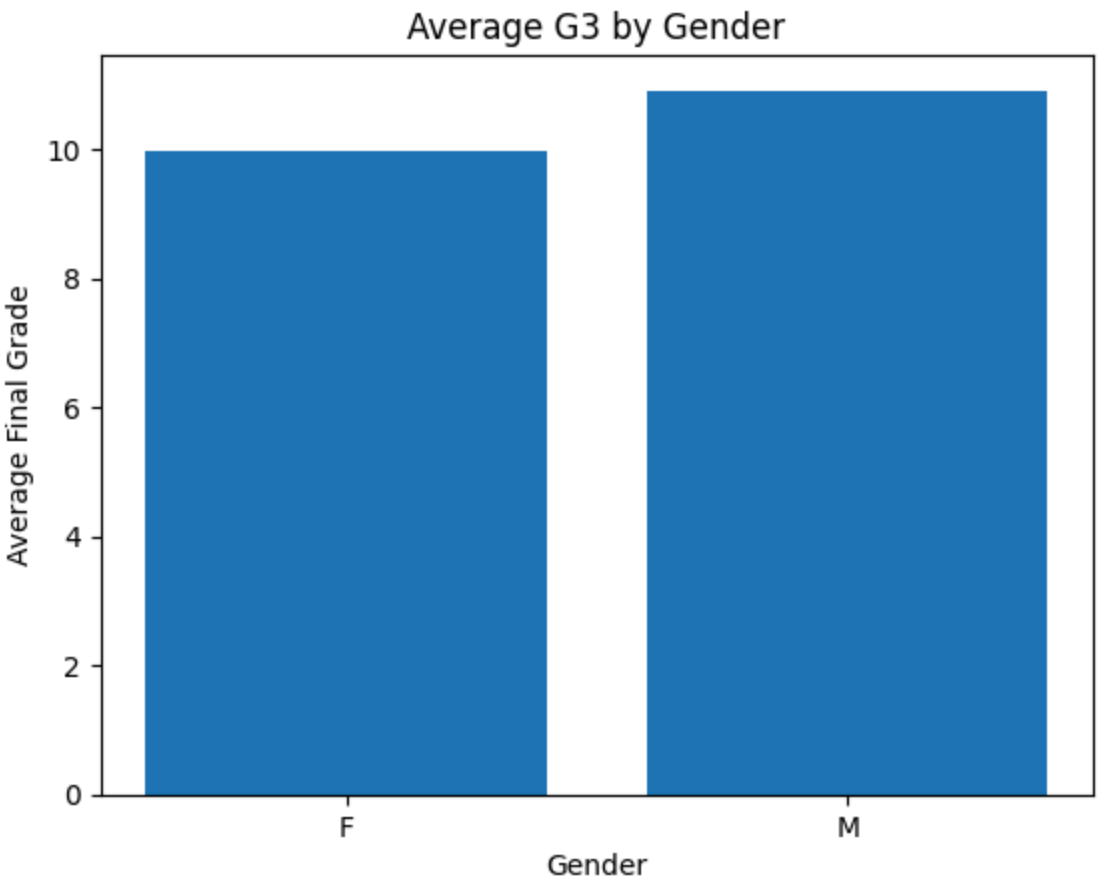
In [ ]:
```
plt.scatter(df['studytime'], df['G3'])
plt.title('Study Time vs G3')
plt.xlabel('Study Time')
plt.ylabel('Final Grade (G3)')
plt.show()
```



In [ ]:

In [ ]:
```
gender_avg = df.groupby('sex')['G3'].mean()

plt.bar(gender_avg.index, gender_avg.values)
plt.title('Average G3 by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Final Grade')
plt.show()
```



5. Documentation (Markdown Summery)

Below is a clean, submission-ready Markdown documentation you can copy into your Jupyter Notebook or internship report.

1. Introduction

This project analyzes the Student Performance Dataset from the UCI Machine Learning Repository. The objective is to practice the full data-analysis workflow:

Load → Clean → Analyze → Visualize → Conclude

The dataset contains 395 students and 33 features, including demographics, family background, study habits, and final grades (G3).

2. Load Dataset

We loaded the dataset directly from the ZIP file provided (student-mat.csv) using pandas.

3. Data Cleaning & Exploration ✔ Missing Values

No missing values found.

✔ Duplicates

No duplicate rows found.

✔ Dataset Shape

395 rows

33 columns

✔ Data Types

Mostly numeric (int)

Some categorical (object)

Dataset is clean and ready for analysis.

4. Analysis Results 1. Average Final Grade (G3)

Average G3 = 10.42

2. Number of High-Scoring Students (G3 > 15)

29 students scored above 15.

3. Correlation: Study Time vs Final Grade

Correlation = 0.097 (very weak positive) Study time has almost no impact on the final grade.

4. Gender Performance Comparison Gender Average G3 Female 10.52 Male 10.34

Females perform slightly better on average.

5. Visualizations Histogram — Final Grades (G3)

Shows distribution of grades from 0 to 20.

Scatter Plot — Study Time vs G3

Weak upward trend consistent with low correlation.

Bar Chart — Average G3 by Gender

Females slightly outperform males.

6. Conclusion

The dataset is clean and complete.

Students typically score around 10 out of 20, indicating moderate performance.

Only 29 students score exceptionally well (G3 > 15).

Study time does NOT strongly influence the final grade.

Female students show a slight advantage in average performance.