Hackverse'23 - Team Hogwards

Visualization File

Name:

- **KRISH GOYAL**
- **ADITYA PRATAP SINGH**
- **ADITYA PRATAP SINGH**
- **KEDHAR KRISSHAN**

Importing Libraries

In [2]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import warnings
import statsmodels.formula.api as smf
import statsmodels.api as sm
warnings.filterwarnings("ignore")
```

Loading the dataset

In [3]:
```
df=pd.read_csv("Student Info.csv")
df
```

Out[3]:

	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	...
...	...	...	...	...	...	...	...	...	...	...	...
1039	SLA	F	19	R	GT3	T	2	3	services	other	...
1040	SLA	F	18	U	LE3	T	3	1	teacher	services	...
1041	SLA	F	18	U	GT3	T	1	1	other	other	...
1042	SLA	M	17	U	LE3	T	3	1	services	services	...
1043	SLA	M	18	R	LE3	T	3	2	services	other	...

1044 rows × 33 columns

Dataset Description

General Information

- **school:** Student's school (binary: 'GP','LVA','MS','SLA')
- **sex:** Student's sex (binary: 'F' - female or 'M' - male)
- **age:** Student's age (numeric: from 15 to 22)
- **address:** Student's home address type (binary: 'U' - urban or 'R' - rural)
- **famsize:** Family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
- **Pstatus:** Parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
- **Medu:** Mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
- **Fedu:** Father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
- **Mjob:** Mother's job (nominal: 'teacher', 'health' care related, civil 'services', 'at_home' or 'other')
- **Fjob:** Father's job (nominal: 'teacher', 'health' care related, civil 'services', 'at_home' or 'other')

Academic Information

- **reason:** Reason to choose this school (nominal: 'home', 'reputation', 'course' preference or 'other')
- **guardian:** Student's guardian (nominal: 'mother', 'father' or 'other')

- **traveltime:** Home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
- **studytime:** Weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- **failures:** Number of past class failures (numeric: n if 1<=n<3, else 4)
- **schoolsup:** Extra educational support (binary: yes or no)
- **famsup:** Family educational support (binary: yes or no)
- **paid:** Extra paid classes within the course subject (binary: yes or no)
- **activities:** Extra-curricular activities (binary: yes or no)
- **nursery:** Attended nursery school (binary: yes or no)
- **higher:** Wants to take higher education (binary: yes or no)
- **internet:** Internet access at home (binary: yes or no)
- **romantic:** With a romantic relationship (binary: yes or no)

Personal and Lifestyle Information

- **famrel:** Quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
- **freetime:** Free time after school (numeric: from 1 - very low to 5 - very high)
- **goout:** Going out with friends (numeric: from 1 - very low to 5 - very high)
- **Dalc:** Workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
- **Walc:** Weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
- **health:** Current health status (numeric: from 1 - very bad to 5 - very good)

Academic Performance

- **absences:** Number of school absences (numeric: from 0 to 93)
- **G1:** First grade (numeric: from 0 to 20)
- **G2:** Second grade (numeric: from 0 to 20)
- **G3:** Final grade (numeric: from 0 to 20)

In [17]:
```
columns_to_check = ['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Me
                    'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studyti
                    'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nu
                    'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goc
                    'Walc', 'health', 'absences', 'G1', 'G2', 'G3']

for column in columns_to_check:
    value_counts = df[column].value_counts()
    print(f"\nValue counts for {column}:\n{value_counts}")
```

```
Value counts for school:
school
LVA    485
GP     349
SLA    164
MS      46
Name: count, dtype: int64

Value counts for sex:
sex
F    591
M    453
Name: count, dtype: int64

Value counts for age:
age
16    281
17    277
18    222
15    194
19     56
20      9
21      3
22      2
Name: count, dtype: int64

Value counts for address:
address
U    759
R    285
Name: count, dtype: int64

Value counts for famsize:
famsize
GT3    738
LE3    306
Name: count, dtype: int64

Value counts for Pstatus:
Pstatus
T    923
A    121
Name: count, dtype: int64

Value counts for Medu:
Medu
4    306
2    289
3    238
1    202
0      9
Name: count, dtype: int64

Value counts for Fedu:
Fedu
2    324
1    256
3    231
4    224
0      9
```

```
Name: count, dtype: int64

Value counts for Mjob:
Mjob
other       399
services    239
at_home     194
teacher     130
health       82
Name: count, dtype: int64

Value counts for Fjob:
Fjob
other       584
services    292
teacher      65
at_home      62
health       41
Name: count, dtype: int64

Value counts for reason:
reason
course        430
home          258
reputation    248
other         108
Name: count, dtype: int64

Value counts for guardian:
guardian
mother    728
father    243
other      73
Name: count, dtype: int64

Value counts for traveltime:
traveltime
1    623
2    320
3     77
4     24
Name: count, dtype: int64

Value counts for studytime:
studytime
2    503
1    317
3    162
4     62
Name: count, dtype: int64

Value counts for failures:
failures
0    861
1    120
2     33
3     30
Name: count, dtype: int64

Value counts for schoolsup:
```

```
schoolsup
no     925
yes    119
Name: count, dtype: int64

Value counts for famsup:
famsup
yes    640
no     404
Name: count, dtype: int64

Value counts for paid:
paid
no     824
yes    220
Name: count, dtype: int64

Value counts for activities:
activities
no     528
yes    516
Name: count, dtype: int64

Value counts for nursery:
nursery
yes    835
no     209
Name: count, dtype: int64

Value counts for higher:
higher
yes    955
no      89
Name: count, dtype: int64

Value counts for internet:
internet
yes    827
no     217
Name: count, dtype: int64

Value counts for romantic:
romantic
no     673
yes    371
Name: count, dtype: int64

Value counts for famrel:
famrel
4    512
5    286
3    169
2     47
1     30
Name: count, dtype: int64

Value counts for freetime:
freetime
3    408
4    293
```

```
2    171
5    108
1     64
Name: count, dtype: int64

Value counts for goout:
goout
3    335
2    248
4    227
5    163
1     71
Name: count, dtype: int64

Value counts for Dalc:
Dalc
1    727
2    196
3     69
5     26
4     26
Name: count, dtype: int64

Value counts for Walc:
Walc
1    398
2    235
3    200
4    138
5     73
Name: count, dtype: int64

Value counts for health:
health
5    395
3    215
4    174
1    137
2    123
Name: count, dtype: int64

Value counts for absences:
absences
0     359
2     175
4     146
6      80
8      64
10     38
12     24
14     20
16     17
5      17
1      15
3      15
9      10
7      10
11      8
18      8
15      5
```

```
22     5
13     4
20     4
21     3
24     2
26     2
30     2
40     1
23     1
17     1
38     1
28     1
19     1
75     1
56     1
54     1
25     1
32     1
Name: count, dtype: int64

Value counts for G1:
G1
10    146
11    130
12    117
13    105
14    101
9      96
8      83
7      70
15     59
16     44
6      33
17     24
18     15
5      12
19      4
4       3
3       1
0       1
Name: count, dtype: int64

Value counts for G2:
G2
11    138
10    129
12    127
9     122
13    117
14     77
8      72
15     72
16     38
7      37
18     26
17     25
6      21
0      20
5      18
19      4
```

```
4       1
Name: count, dtype: int64

Value counts for G3:
G3
10    153
11    151
13    113
12    103
14     90
15     82
8      67
9      63
0      53
16     52
17     35
18     27
7      19
6      18
5       8
19      7
20      1
4       1
1       1
Name: count, dtype: int64
```

In [18]:
```
df.describe()
```

Out[18]:

	age	Medu	Fedu	traveltime	studytime	failures	
count	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1
mean	16.726054	2.603448	2.387931	1.522989	1.970307	0.264368	
std	1.239975	1.124907	1.099938	0.731727	0.834353	0.656142	
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	
25%	16.000000	2.000000	1.000000	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	

In [19]:
```
df.info
```

Out[19]:
```
<bound method DataFrame.info of       school sex  age address famsize Pstatus  M
edu  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
...     ...  ..  ...     ...     ...     ...   ...   ...       ...       ...
1039    SLA   F   19       R     GT3       T     2     3  services     other
1040    SLA   F   18       U     LE3       T     3     1   teacher  services
1041    SLA   F   18       U     GT3       T     1     1     other     other
1042    SLA   M   17       U     LE3       T     3     1  services  services
1043    SLA   M   18       R     LE3       T     3     2  services     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
...   ...    ...      ...    ...   ...   ...    ...      ...  ..  ..  ..
1039  ...      5        4      2     1     2      5        4  10  11  10
1040  ...      4        3      4     1     1      1        4  15  15  16
1041  ...      1        1      1     1     1      5        6  11  12   9
1042  ...      2        4      5     3     4      2        6  10  10  10
1043  ...      4        4      1     3     4      5        4  10  11  11

[1044 rows x 33 columns]>
```

In [20]:
```
df.nunique()
```

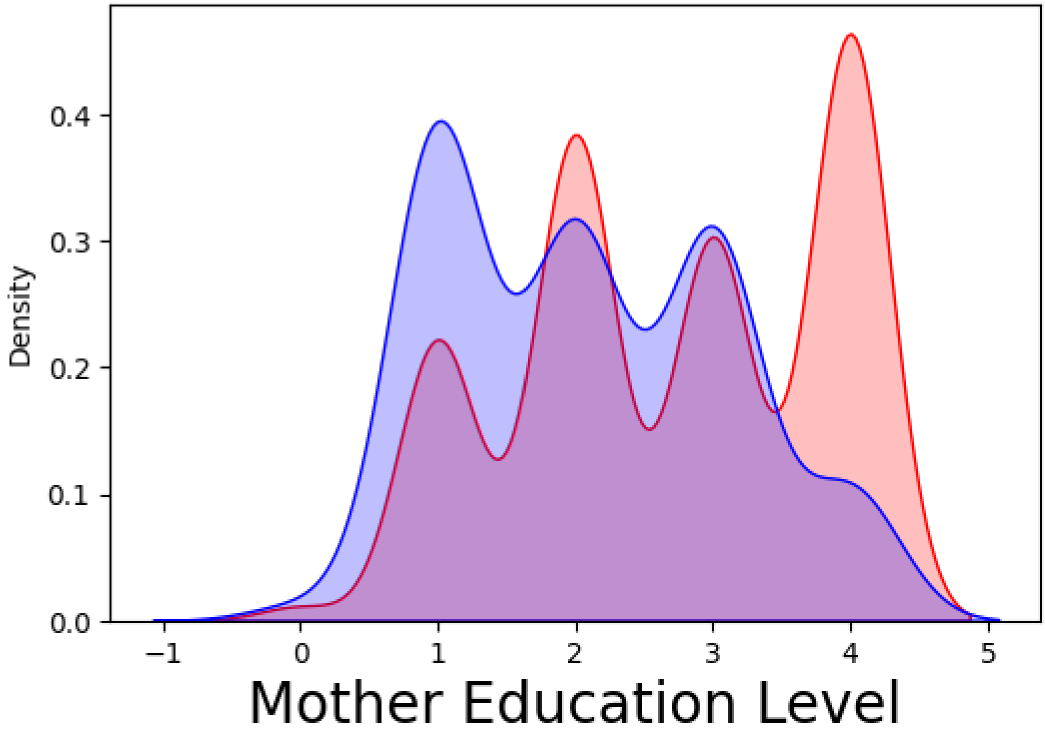
Out[20]:
```
school         4
sex            2
age            8
address        2
famsize        2
Pstatus        2
Medu           5
Fedu           5
Mjob           5
Fjob           5
reason         4
guardian       3
traveltime     4
studytime      4
failures       4
schoolsup      2
famsup         2
paid           2
activities     2
nursery        2
higher         2
internet       2
romantic       2
famrel         5
freetime       5
goout          5
Dalc           5
Walc           5
health         5
absences      35
G1            18
G2            17
G3            19
dtype: int64
```

In [21]:
```
df.duplicated().sum()
```

Out[21]:
```
0
```

In [22]:
```
good = df.loc[df.failures==0]
poor=df.loc[df.failures>=1]
good['good_student_mother_education'] = good.Medu
poor['poor_student_mother_education'] = poor.Medu
plt.figure(figsize=(6,4))
p=sns.kdeplot(good['good_student_mother_education'], shade=True, color="r")#gooc
p=sns.kdeplot(poor['poor_student_mother_education'], shade=True, color="b")#poor
plt.xlabel('Mother Education Level', fontsize=20)
```

Out[22]:
```
Text(0.5, 0, 'Mother Education Level')
```



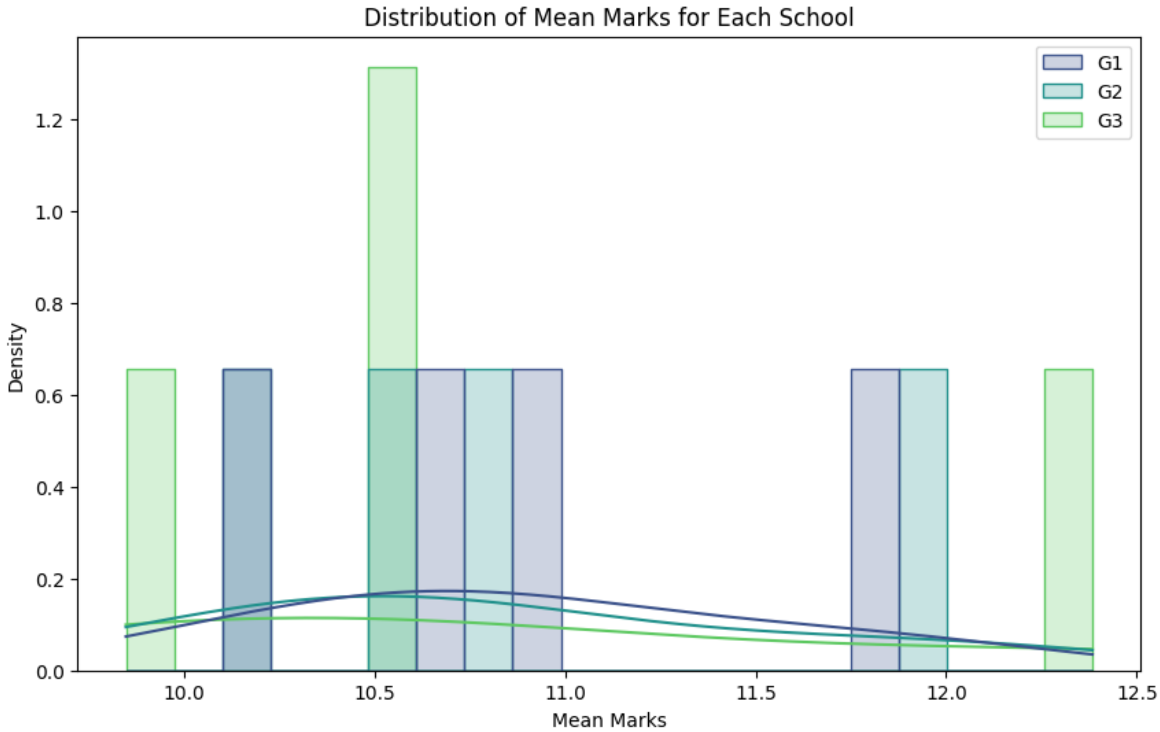
In [23]:
```
# Creating a frequency plot
fig = px.histogram(df, x='school', color='failures',
                   title='Frequency of Failures for Each School',
                   labels={'school': 'School Code', 'failures': 'Number of Failu
                   category_orders={'school': ['LVA', 'GP', 'SLA', 'MS']},
                   color_discrete_sequence=px.colors.sequential.Plasma)

# Showing the plot
fig.show()
```

In [24]:
```
# Grouping by 'school' and calculating the mean for each group
school_means = df.groupby('school')[['G1', 'G2', 'G3']].mean()

# Displaying the mean marks for each school
print("Mean marks for each school:")
print(school_means)
```

```
Mean marks for each school:
               G1         G2         G3
school
GP      10.939828  10.782235  10.489971
LVA     11.797938  11.934021  12.383505
MS      10.673913  10.195652   9.847826
SLA     10.219512  10.493902  10.493902
```

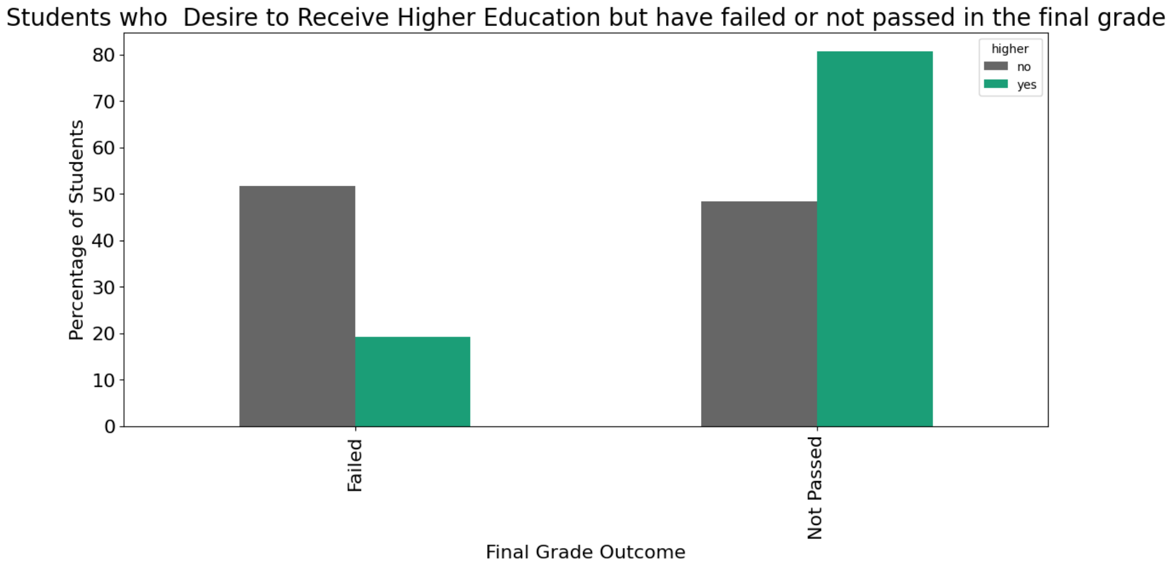
In [25]:
```
school_means = df.groupby('school')[['G1', 'G2', 'G3']].mean()

# Plotting the distribution for each school using Seaborn
plt.figure(figsize=(10, 6))
sns.histplot(data=school_means, kde=True, bins=20, palette='viridis', element='s

# Adding labels and title
plt.xlabel('Mean Marks')
plt.ylabel('Density')
plt.title('Distribution of Mean Marks for Each School')
```

```python
# Showing the plot
plt.show()
```



The graph shows the distribution of mean marks for each school

In [26]:
```python
df['failed_not_passed'] = df.apply(lambda row: 'Failed' if row['G3'] < 10 else '


def perc(val):
    return val / val.sum() * 100

# Creating a cross-tabulation
failed_not_passed_tab = pd.crosstab(index=df['failed_not_passed'], columns=df['h

# Applying the percentage function and reindex the columns
failed_not_passed_perc = failed_not_passed_tab.apply(perc).reindex(['Failed', 'N

# Plotting the bar chart
failed_not_passed_perc.plot.bar(colormap="Dark2_r", figsize=(14, 6), fontsize=16
plt.title('Students who  Desire to Receive Higher Education but have failed or n
plt.xlabel('Final Grade Outcome', fontsize=16)
plt.ylabel('Percentage of Students', fontsize=16)
plt.show()
```



In [4]:
```
label_encoder = LabelEncoder()

categorical_columns = ['sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', '
                        'schoolsup', 'famsup', 'paid', 'activities', 'nursery',

for column in categorical_columns:
    df[column] = label_encoder.fit_transform(df[column])

# Displaying the updated DataFrame
df
```

Out[4]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famr
0	GP	0	18	1	0	0	4	4	0	4	...	
1	GP	0	17	1	0	1	1	1	0	2	...	
2	GP	0	15	1	1	1	1	1	0	2	...	
3	GP	0	15	1	0	1	4	2	1	3	...	
4	GP	0	16	1	0	1	3	3	2	2	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1039	SLA	0	19	0	0	1	2	3	3	2	...	
1040	SLA	0	18	1	1	1	3	1	4	3	...	
1041	SLA	0	18	1	0	1	1	1	2	2	...	
1042	SLA	1	17	1	1	1	3	1	3	3	...	
1043	SLA	1	18	0	1	1	3	2	3	2	...	

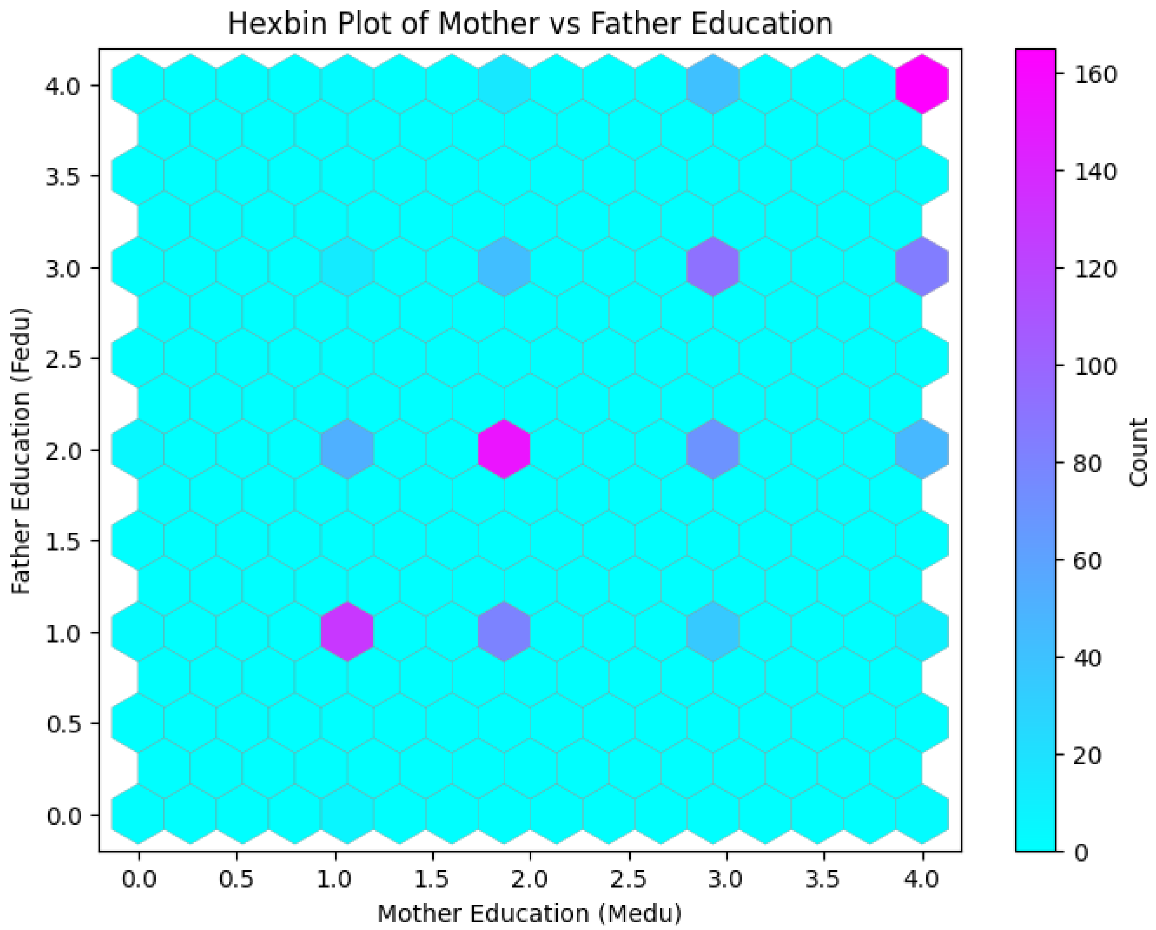
1044 rows × 33 columns

In [28]:
```
numeric_features = [feature for feature in df.columns if df[feature].dtype != "c
categorical_features = [feature for feature in df.columns if df[feature].dtype =

print("We have {} numerical features: {}".format(len(numeric_features),numeric_f
print("We have {} categorical features: {}".format(len(categorical_features),cat
```

```
We have 32 numerical features: ['sex', 'age', 'address', 'famsize', 'Pstatus', 'M
edu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'f
ailures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'inte
rnet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'abse
nces', 'G1', 'G2', 'G3']
We have 2 categorical features: ['school', 'failed_not_passed']
```

In [29]:

```
plt.figure(figsize=(8, 6))
plt.hexbin(df['Medu'], df['Fedu'], gridsize=15, cmap='cool', edgecolors='gray',
plt.colorbar(label='Count')
plt.xlabel('Mother Education (Medu)')
plt.ylabel('Father Education (Fedu)')
plt.title('Hexbin Plot of Mother vs Father Education')
plt.show()
```



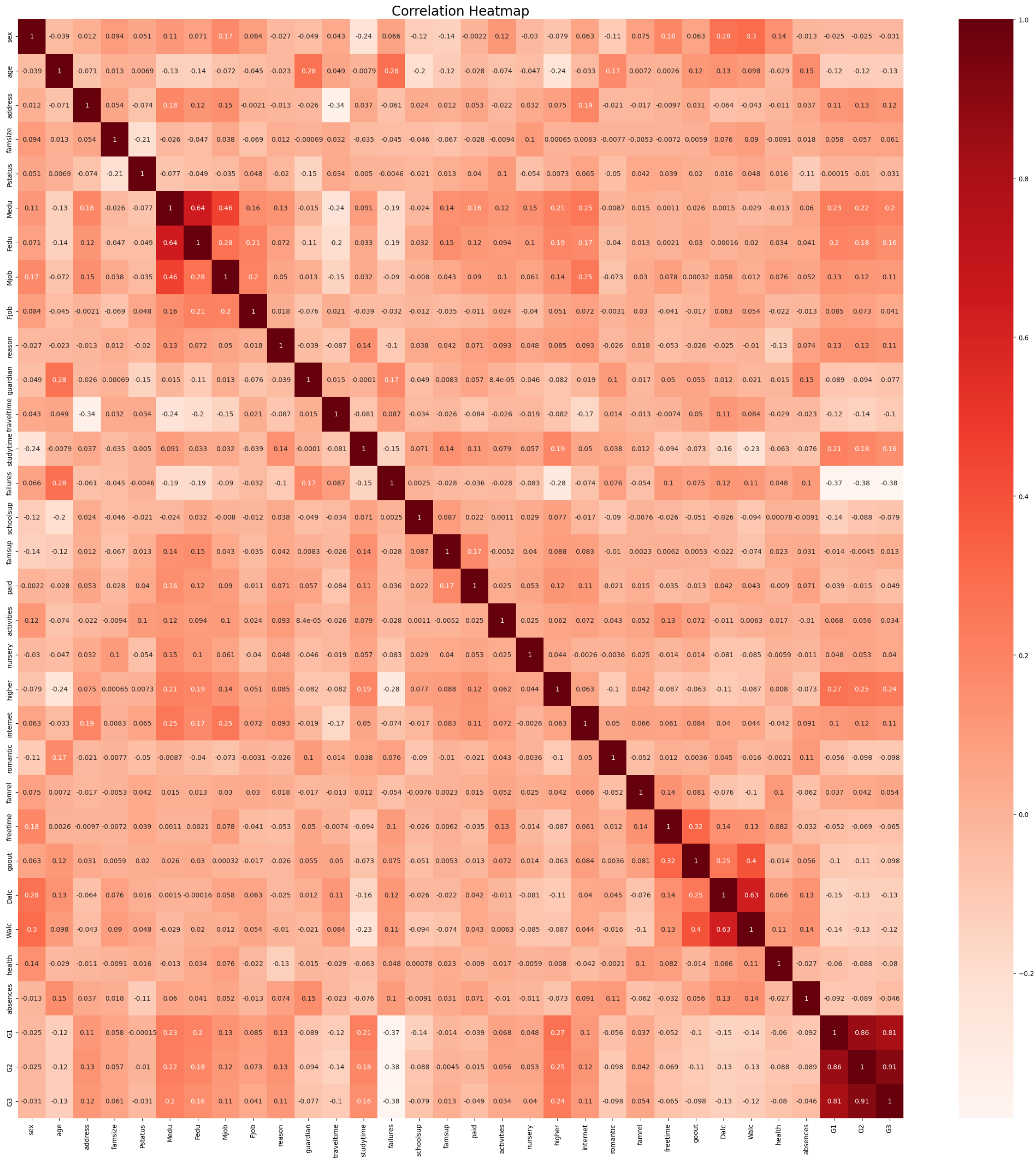
- **Hexbin Plot Description:**
  - Represents the relationship between mother and father education using a hexbin plot.
- **Axes Representation:**
  - X-axis represents mother education.
  - Y-axis represents father education.
- **Color Representation:**
  - The color of the hexagons represents the count of data points in that bin.
  - Darker colors indicate a higher count, and lighter colors indicate a lower count.
- **Observations:**

- The plot effectively illustrates the relationship between mother and father education.
- Darker colors represent a higher density of data points, while lighter colors represent a lower density.
- A positive correlation is observed between mother and father education, suggesting that as one parent's education level increases, the other parent's education level tends to increase as well.
- The highest count is found in the bin where mother education is around 2.5 and father education is around 2.5, indicating that this combination is the most common among the data points.

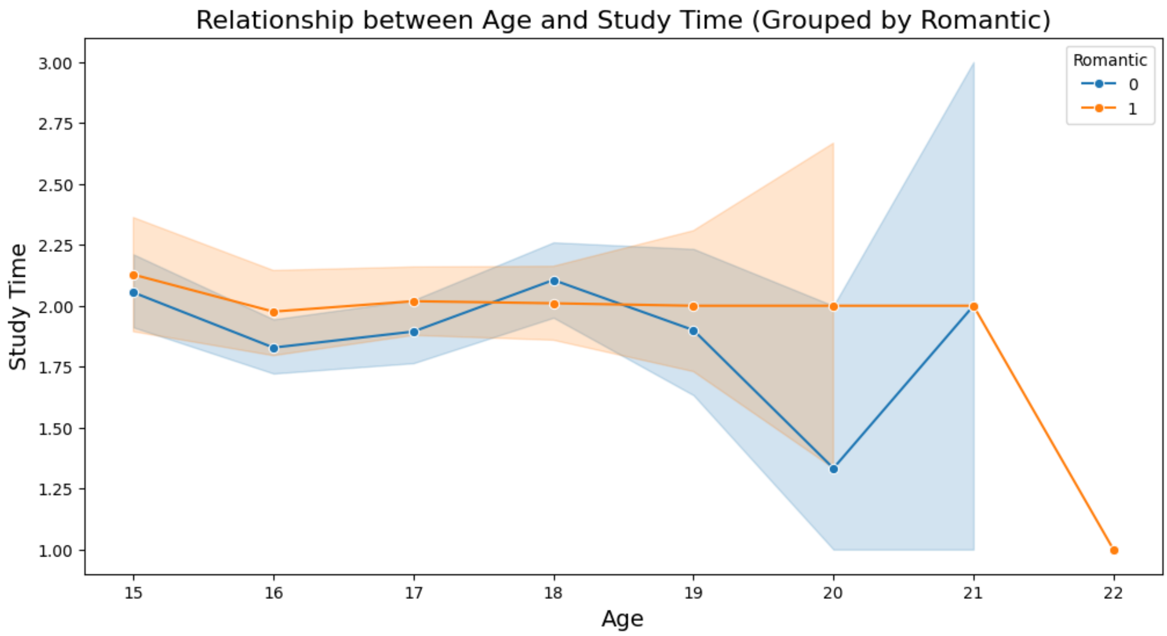
In [5]:
```
df1 = df
df1.drop(['school'], axis=1, inplace=True)
```

In [6]:
```
plt.figure(figsize=(30,30))
sns.heatmap(df1.corr(), annot=True, cmap="Reds")
plt.title('Correlation Heatmap', fontsize=20)
```

Out[6]:
```
Text(0.5, 1.0, 'Correlation Heatmap')
```



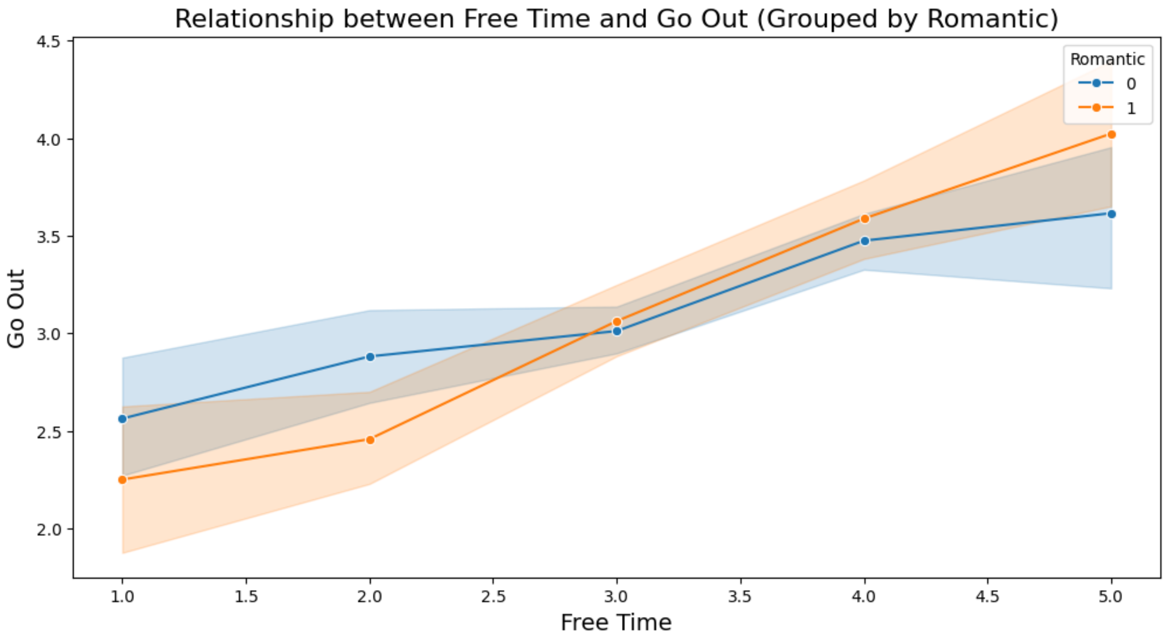
```
In [ ]: plt.figure(figsize=(12, 6))
        sns.lineplot(x='age', y='studytime', hue='romantic', data=df, marker='o')

        # Setting labels and title
        plt.xlabel('Age', fontsize=14)
        plt.ylabel('Study Time', fontsize=14)
        plt.title('Relationship between Age and Study Time (Grouped by Romantic)', fonts

        # Showing the legend
        plt.legend(title='Romantic', loc='upper right')

        # Showing the plot
        plt.show()
```



- **Line Graph Description:**
  - Shows the relationship between age and study time.
  - Grouped by romantic status.
- **Axes Representation:**
  - X-axis represents age.
  - Y-axis represents study time in hours.
- **Lines:**
  - Two lines present: one for individuals in a romantic relationship, and one for those who are not.
- **Line Colors:**
  - Line for those in a romantic relationship is orange.
  - Line for those not in a romantic relationship is blue.
- **Data Representation:**
  - Lines are connected by dots representing data points.
- **Shading:**
  - Area between the lines is shaded in light blue.
- **Observations:**
  - Both lines have a positive slope, indicating that as age increases, study time also increases.
  - The line for those in a romantic relationship has a steeper slope, suggesting a stronger relationship between age and study time for this group.

In [ ]:
```
# Creatin a line plot for 'freetime' and 'goout' grouped by 'romantic'
plt.figure(figsize=(12, 6))
sns.lineplot(x='freetime', y='goout', hue='romantic', data=df, marker='o')

# Setting labels and title
```

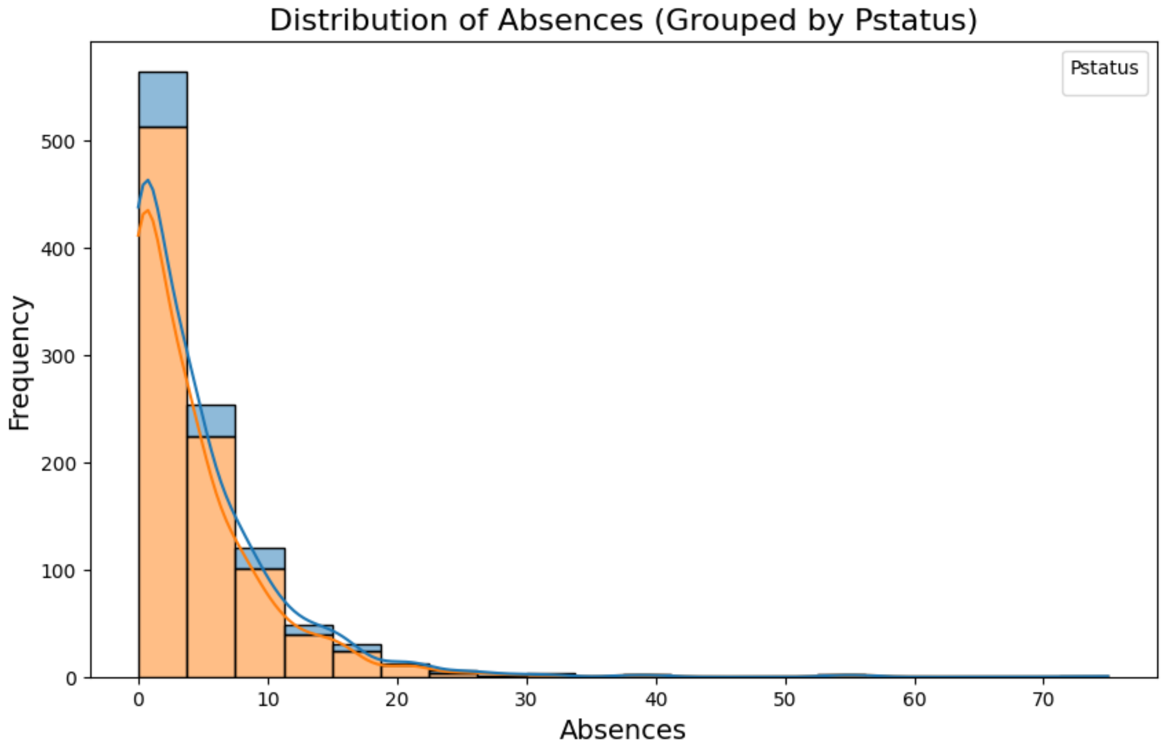
```
plt.xlabel('Free Time', fontsize=14)
plt.ylabel('Go Out', fontsize=14)
plt.title('Relationship between Free Time and Go Out (Grouped by Romantic)', fon

# Showing the legend
plt.legend(title='Romantic', loc='upper right')

# Showing the plot
plt.show()
```



- **Line Graph Description:**
  - Shows the relationship between free time and going out.
  - Grouped by romantic status.
- **Axes Representation:**
  - X-axis represents free time.
  - Y-axis represents going out.
- **Lines:**
  - Two lines present: one for individuals in a romantic relationship, and one for those who are not.
- **Line Colors:**
  - Line for those in a romantic relationship is red.
  - Line for those not in a romantic relationship is blue.
- **Line Slopes:**
  - Both lines have a positive slope.
  - Indicates that as free time increases, going out also increases.
- **Comparison of Slopes:**
  - The line for those in a romantic relationship has a steeper slope.
  - Suggests a stronger relationship between free time and going out for individuals in a romantic relationship.

In [ ]:
```
# Creating a histogram for 'absences' grouped by 'Pstatus'
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='absences', hue='Pstatus', multiple='stack', kde=True, b

# Setting labels and title
plt.xlabel('Absences', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.title('Distribution of Absences (Grouped by Pstatus)', fontsize=16)

# Showing the legend
plt.legend(title='Pstatus')

# Showing the plot
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```

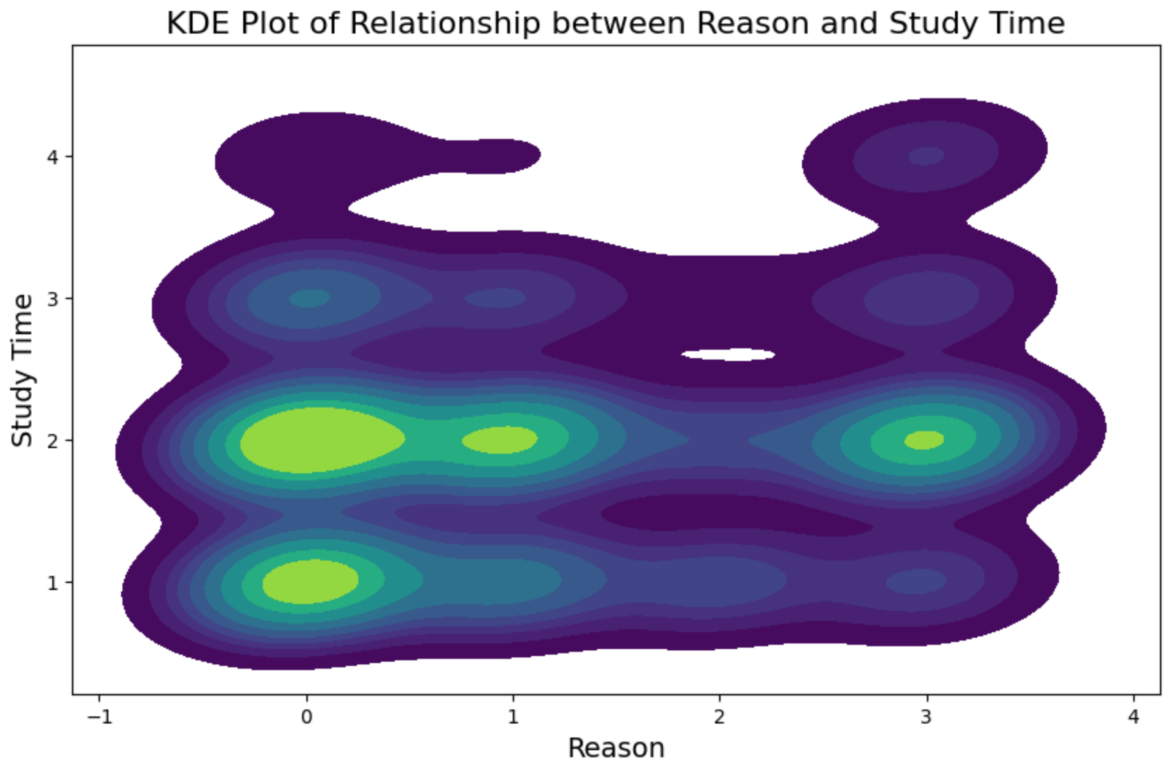


- **Histogram Description:**
  - Shows the distribution of absences grouped by Pstatus.
- **Axes Representation:**
  - X-axis represents the number of absences.
  - Y-axis represents the frequency.
- **Distribution Shape:**
  - The graph is skewed to the right.
- **Skewness Interpretation:**
  - Indicates more students with a lower number of absences than students with a higher number of absences.
- **Frequency Peaks:**

- The highest frequency of absences is around 0-10 absences.
- The lowest frequency of absences is around 60-70 absences.

- **Observations:**
    - The graph effectively illustrates the distribution of absences based on Pstatus.

In [32]:
```
# Creating a KDE plot for the relationship between 'reason' and 'studytime'
plt.figure(figsize=(10, 6))
sns.kdeplot(data=df, x='reason', y='studytime', fill=True, cmap='viridis')

# Setting labels and title
plt.xlabel('Reason', fontsize=14)
plt.ylabel('Study Time', fontsize=14)
plt.title('KDE Plot of Relationship between Reason and Study Time', fontsize=16)

# Showing the plot
plt.show()
```

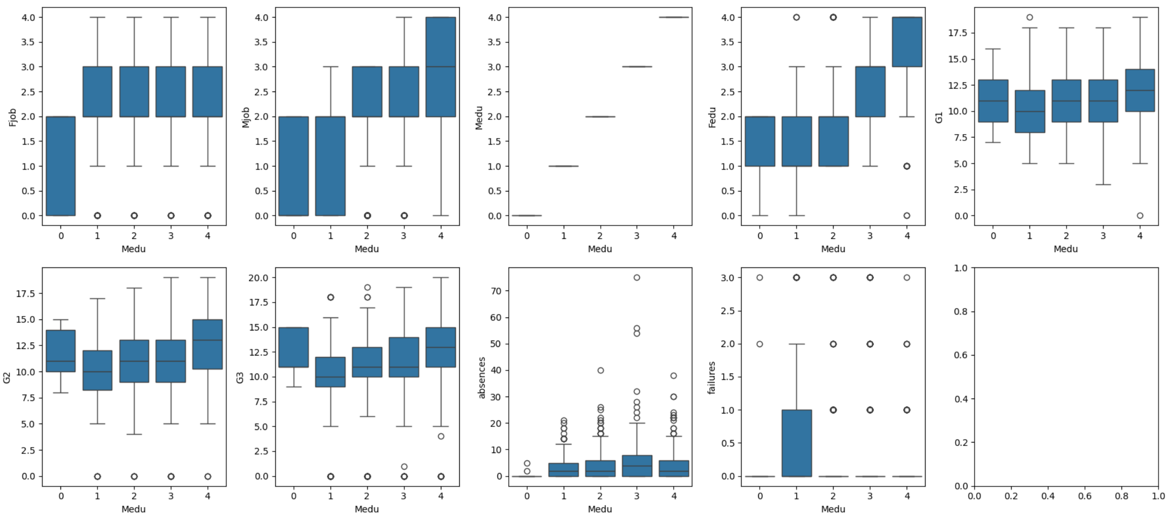


- **KDE Plot Description:**
    - Represents the relationship between reason and study time using Kernel Density Estimation (KDE).
- **Axes Representation:**
    - X-axis represents reason.
    - Y-axis represents study time.
- **Plot Type:**
    - Contour plot.
- **Color Representation:**

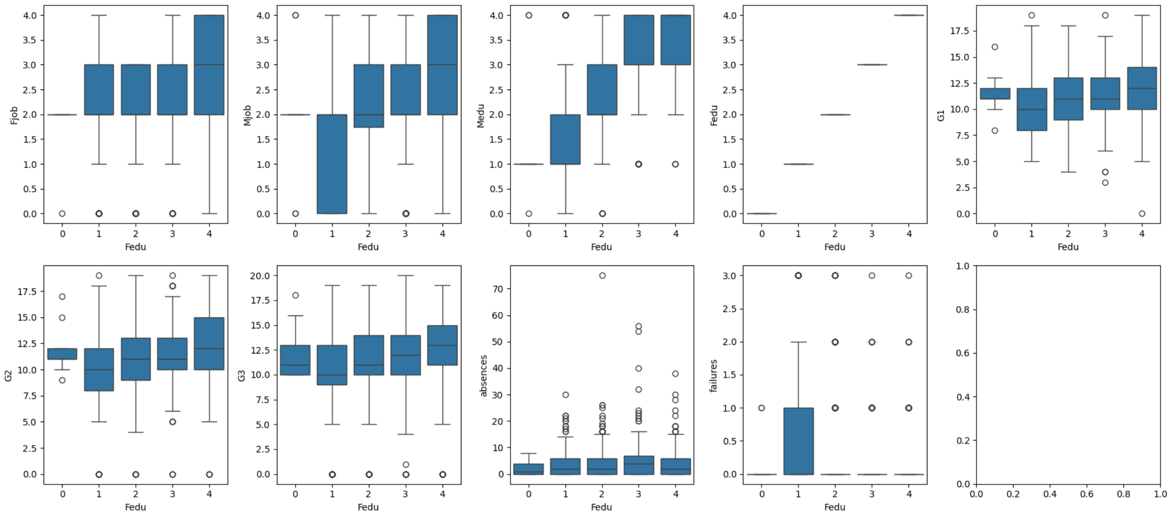
- Darker colors represent a higher density of data points.
- Lighter colors represent a lower density of data points.

- **Observations:**
  - The plot effectively illustrates the relationship between reason and study time.
  - Darker colors indicate a higher concentration of data points.
  - There is a higher density of data points in the lower-left corner and a lower density in the upper-right corner, suggesting a negative relationship.
  - This implies that as the reason for absence increases, the study time decreases.

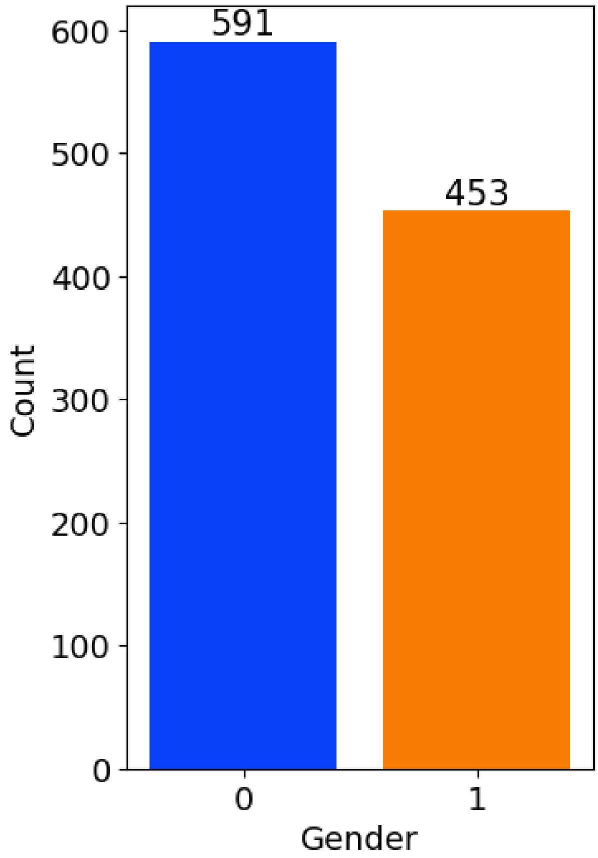
In [ ]:
```
# Selecting the relevant columns for visualization
columns_of_interest = ['Fjob', 'Mjob', 'Medu', 'Fedu', 'G1', 'G2', 'G3', 'absenc
df_subset = df[columns_of_interest]

# Setting up subplots
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(18, 8))

# Plotting box plots for each variable
for i, column in enumerate(columns_of_interest):
    sns.boxplot(x='Medu', y=column, data=df_subset, ax=axes[i // 5, i % 5])

# Adjusting layout
plt.tight_layout()
plt.show()
```

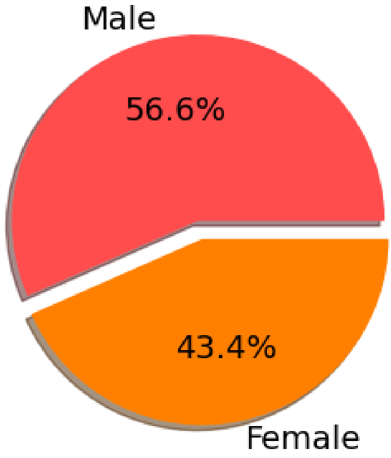


In [ ]:
```
# Selecting the relevant columns for visualization
columns_of_interest = ['Fjob', 'Mjob', 'Medu', 'Fedu', 'G1', 'G2', 'G3', 'absenc
df_subset = df[columns_of_interest]

# Setting up subplots
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(18, 8))

# Plotting box plots for each variable
for i, column in enumerate(columns_of_interest):
    sns.boxplot(x='Fedu', y=column, data=df_subset, ax=axes[i // 5, i % 5])

# Adjusting layout
plt.tight_layout()
plt.show()
```



In [ ]:
```
# Create a figure with two subplots
f,ax=plt.subplots(1,2,figsize=(8,6))


# Create a countplot of the 'gender' column and add labels to the bars
sns.countplot(x=df['sex'],data=df,palette ='bright',ax=ax[0],saturation=0.95)
for container in ax[0].containers:
    ax[0].bar_label(container,color='black',size=15)

# Set font size of x-axis and y-axis labels and tick labels
ax[0].set_xlabel('Gender', fontsize=14)
ax[0].set_ylabel('Count', fontsize=14)
ax[0].tick_params(labelsize=14)

# Create a pie chart of the 'gender' column and add labels to the slices
plt.pie(x=df['sex'].value_counts(),labels=['Male','Female'],explode=[0,0.1],autc

# Display the plot
plt.show()
```





In [33]:
```
df['grade_overall'] = df['G1'] + df['G2'] + df['G3']
df['Alc_Tot'] = df['Dalc'] + df['Walc']*0.4
df['Alc_Tot'] = round((round(df['Alc_Tot']*10/7)-2)*10/8)
df
```

Out[33]:

	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	...	Dalc
0	0	18	1	0	0	4	4	0	4	0	...	1
1	0	17	1	0	1	1	1	0	2	0	...	1
2	0	15	1	1	1	1	1	0	2	2	...	2
3	0	15	1	0	1	4	2	1	3	1	...	1
4	0	16	1	0	1	3	3	2	2	1	...	1
...	...	...	...	...	...	...	...	...	...	...	...	...
1039	0	19	0	0	1	2	3	3	2	0	...	1
1040	0	18	1	1	1	3	1	4	3	0	...	1
1041	0	18	1	0	1	1	1	2	2	0	...	1
1042	1	17	1	1	1	3	1	3	3	0	...	3
1043	1	18	0	1	1	3	2	3	2	0	...	3

1044 rows × 35 columns

In [34]:
```
data = df
```

```
li = list(data.columns)
```

In [35]:
```
model = smf.ols("grade_overall ~ Medu + Fedu + age + Mjob + Fjob + traveltime +
result = model.fit()
result.summary()
```

Out[35]:

OLS Regression Results

Dep. Variable:	grade_overall	**R-squared:**	0.121
Model:	OLS	**Adj. R-squared:**	0.110
Method:	Least Squares	**F-statistic:**	10.87
Date:	Sun, 03 Dec 2023	**Prob (F-statistic):**	5.04e-22
Time:	06:54:53	**Log-Likelihood:**	-3781.2
No. Observations:	1044	**AIC:**	7590.
Df Residuals:	1030	**BIC:**	7660.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	39.7436	4.489	8.853	0.000	30.934	48.553
Medu	1.1160	0.361	3.095	0.002	0.408	1.823
Fedu	0.6149	0.341	1.802	0.072	-0.055	1.285
age	-0.6033	0.235	-2.563	0.011	-1.065	-0.141
Mjob	0.2184	0.262	0.834	0.405	-0.296	0.732
Fjob	0.2806	0.342	0.821	0.412	-0.390	0.951
traveltime	-0.7770	0.402	-1.934	0.053	-1.565	0.011
studytime	1.7044	0.352	4.849	0.000	1.015	2.394
freetime	-0.2733	0.295	-0.926	0.354	-0.852	0.306
goout	-0.5739	0.285	-2.016	0.044	-1.133	-0.015
famrel	0.5358	0.312	1.720	0.086	-0.075	1.147
Walc	-0.2536	0.253	-1.001	0.317	-0.751	0.244
health	-0.5393	0.203	-2.652	0.008	-0.938	-0.140
absences	-0.0912	0.047	-1.948	0.052	-0.183	0.001

Omnibus:	39.153	**Durbin-Watson:**	1.827
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	44.749
Skew:	-0.436	**Prob(JB):**	1.92e-10
Kurtosis:	3.517	**Cond. No.**	313.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [36]:
```
model = smf.ols("grade_overall ~ Medu + Fedu",data = data)
result = model.fit()

print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          grade_overall   R-squared:                       0.054
Model:                            OLS   Adj. R-squared:                  0.053
Method:                 Least Squares   F-statistic:                     29.91
Date:                Sun, 03 Dec 2023   Prob (F-statistic):           2.34e-13
Time:                        06:54:58   Log-Likelihood:                -3819.1
No. Observations:                1044   AIC:                             7644.
Df Residuals:                    1041   BIC:                             7659.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     28.2700      0.778     36.316      0.000      26.742      29.797
Medu           1.5643      0.337      4.635      0.000       0.902       2.226
Fedu           0.6111      0.345      1.771      0.077      -0.066       1.288
==============================================================================
Omnibus:                       27.523   Durbin-Watson:                   1.783
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               29.870
Skew:                          -0.367   Prob(JB):                     3.26e-07
Kurtosis:                       3.386   Cond. No.                         10.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Pivot Tables

In [37]:
```
pivot = pd.pivot_table(df,
            values = ['G1', 'G2', 'G3'],
            index = ['Alc_Tot'],
                        columns= ['failures'],
                        aggfunc='mean',
                        margins=True).fillna(0)
pivot
```

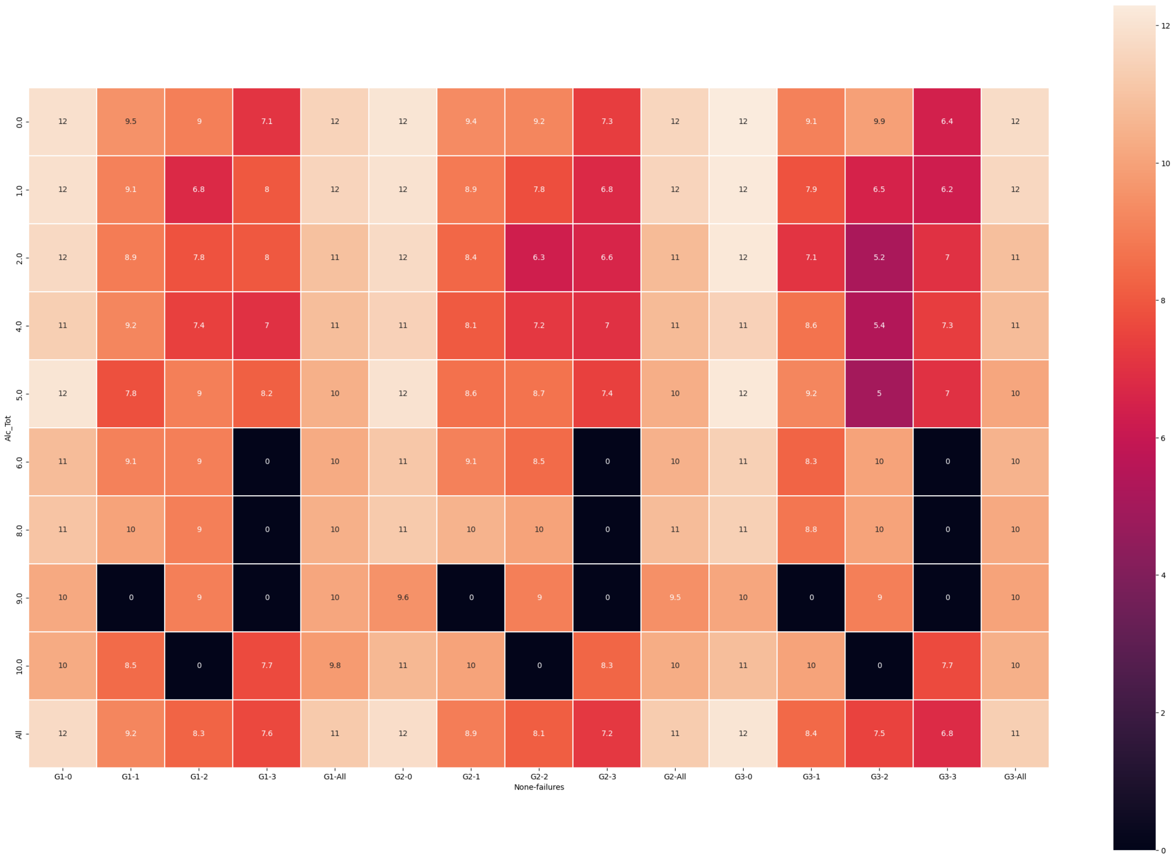
Out[37]:

	G1							
failures	**0**	**1**	**2**	**3**	**All**	**0**	**1**	
Alc_Tot								
0.0	11.942943	9.513514	9.000000	7.100000	11.506394	12.084084	9.351351	9.1
1.0	11.936759	9.074074	6.750000	8.000000	11.541667	11.968379	8.851852	7.7
2.0	11.687500	8.923077	7.833333	8.000000	10.942308	11.725000	8.384615	6.3
4.0	11.367925	9.176471	7.400000	7.000000	10.832061	11.452830	8.058824	7.2
5.0	12.058824	7.800000	9.000000	8.200000	10.400000	12.000000	8.600000	8.6
6.0	10.794872	9.066667	9.000000	0.000000	10.267857	11.102564	9.066667	8.5
8.0	11.000000	10.000000	9.000000	0.000000	10.400000	11.200000	10.500000	10.0
9.0	10.222222	0.000000	9.000000	0.000000	10.100000	9.555556	0.000000	9.0
10.0	10.263158	8.500000	0.000000	7.666667	9.791667	10.631579	10.000000	0.0
All	11.736353	9.175000	8.272727	7.600000	11.213602	11.829268	8.933333	8.1

In [38]:

```
cmap = sns.cubehelix_palette(start = 1.5, rot = 1.5, as_cmap = True)
plt.subplots(figsize = (30, 20))
sns.heatmap(pivot,linewidths=0.2,square=True, annot = True )
```

Out[38]: <Axes: xlabel='None-failures', ylabel='Alc_Tot'>



_________________________********************___________________________