

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
# I'm loading the dataset in this line of code.
df = pd.read_csv("/content/ai_student_impact_dataset (1).csv")
```

```
# I'm dropping this column simply because it's not usefull for statistical analysis.
if 'Student_ID' in df.columns:
    df.drop('Student_ID', axis=1, inplace=True)
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 15 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Major_Category                        49993 non-null  object
1   Year_of_Study                        49991 non-null  object
2   Pre_Semester_GPA                    49993 non-null  float64
3   Weekly_GenAI_Hours                  49989 non-null  float64
4   Primary_Use_Case                    49972 non-null  object
5   Prompt_Engineering_Skill            49995 non-null  object
6   Tool_Diversity                      49991 non-null  float64
7   Paid_Subscription                   50000 non-null  bool
8   Traditional_Study_Hours              49993 non-null  float64
9   Perceived_AI_Dependency              50000 non-null  int64
10  Institutional_Policy                 49983 non-null  object
11  Anxiety_Level_During_Exams           49997 non-null  float64
12  Post_Semester_GPA                   49993 non-null  float64
13  Skill_Retention_Score                49993 non-null  float64
14  Burnout_Risk_Level                   50000 non-null  object
dtypes: bool(1), float64(7), int64(1), object(6)
memory usage: 5.4+ MB
None
```

```
df.head()
```

	Major_Category	Year_of_Study	Pre_Semester_GPA	Weekly_GenAI_Hours	Primary_Use_Case	Prompt_Engineering_Skill	Tool_Diversity	Paid_Subscription
0	Humanities	Senior	2.418	23.31	Copywriting/Drafting	Beginner	1.0	True
1	Medical	Junior	3.821	1.12	Ideation	Advanced	5.0	True
2	Business	Freshman	3.398	21.26	Summarizing_Reading	Beginner	2.0	True
3	Business	Senior	3.789	1.82	Copywriting/Drafting	Intermediate	4.0	True
4	STEM	Sophomore	3.635	9.29	Debugging/Troubleshooting	Advanced	4.0	True

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
df.dtypes
```

	0
Major_Category	object
Year_of_Study	object
Pre_Semester_GPA	float64
Weekly_GenAI_Hours	float64
Primary_Use_Case	object
Prompt_Engineering_Skill	object
Tool_Diversity	float64
Paid_Subscription	bool
Traditional_Study_Hours	float64
Perceived_AI_Dependency	int64
Institutional_Policy	object
Anxiety_Level_During_Exams	float64
Post_Semester_GPA	float64
Skill_Retention_Score	float64
Burnout_Risk_Level	object

dtype: object

```
df.columns
```

```
Index(['Major_Category', 'Year_of_Study', 'Pre_Semester_GPA',  
      'Weekly_GenAI_Hours', 'Primary_Use_Case', 'Prompt_Engineering_Skill',  
      'Tool_Diversity', 'Paid_Subscription', 'Traditional_Study_Hours',  
      'Perceived_AI_Dependency', 'Institutional_Policy',  
      'Anxiety_Level_During_Exams', 'Post_Semester_GPA',  
      'Skill_Retention_Score', 'Burnout_Risk_Level'],  
      dtype='object')
```

```
df.isnull().sum()
```

	0
Major_Category	7
Year_of_Study	9
Pre_Semester_GPA	7
Weekly_GenAI_Hours	11
Primary_Use_Case	28
Prompt_Engineering_Skill	5
Tool_Diversity	9
Paid_Subscription	0
Traditional_Study_Hours	7
Perceived_AI_Dependency	0
Institutional_Policy	17
Anxiety_Level_During_Exams	3
Post_Semester_GPA	7
Skill_Retention_Score	7
Burnout_Risk_Level	0

```
dtype: int64
```

```
missing_percentage = df.isnull().mean() * 100
```

```
threshold = 5
```

```
columns_to_drop = missing_percentage[missing_percentage >  
threshold].index
```

```
df = df.drop(columns=columns_to_drop)
```

```
print(f"Columns dropped: {list(columns_to_drop)}")  
print(f"Remaining columns: {df.columns}")
```

```
Columns dropped: []  
Remaining columns: Index(['Major_Category', 'Year_of_Study', 'Pre_Semester_GPA',  
      'Weekly_GenAI_Hours', 'Primary_Use_Case', 'Prompt_Engineering_Skill',  
      'Tool_Diversity', 'Paid_Subscription', 'Traditional_Study_Hours',  
      'Perceived_AI_Dependency', 'Institutional_Policy',  
      'Anxiety_Level_During_Exams', 'Post_Semester_GPA',  
      'Skill_Retention_Score', 'Burnout_Risk_Level'],  
      dtype='object')
```

```
# In this code block I'm encoding two types of data (1- object. 2- bool.)
```

```
from sklearn.preprocessing import LabelEncoder  
from sklearn.preprocessing import OrdinalEncoder  
label_encoder = LabelEncoder()  
ordinal_encoder = OrdinalEncoder()
```

```
#I'm using OrdinalEncoder to encode 'Burnout_Risk_Level' column in a specific order Low: 0 < Medium: 1 < High: 2.  
categories_order = [['Low', 'Medium', 'High']]  
encoder = OrdinalEncoder(categories=categories_order)  
df[['Burnout_Risk_Level']] = encoder.fit_transform(df[['Burnout_Risk_Level']])  
#-----
```

```
#Encoding the other categorical columns  
for column in df.select_dtypes(include=['object','bool']).columns:  
    df[column] = label_encoder.fit_transform(df[column])
```

```
df.head()
```

	Major_Category	Year_of_Study	Pre_Semester_GPA	Weekly_GenAI_Hours	Primary_Use_Case	Prompt_Engineering_Skill	Tool_Diversity	Paid_Sut
0		2	3	2.418	23.31	0	1	1.0
1		3	2	3.821	1.12	3	0	5.0
2		1	0	3.398	21.26	4	1	2.0
3		1	3	3.789	1.82	0	2	4.0
4		4	4	3.635	9.29	1	0	4.0

Next steps: [Generate code with df](#) [New interactive sheet](#)

df.dtypes

	0
Major_Category	int64
Year_of_Study	int64
Pre_Semester_GPA	float64
Weekly_GenAI_Hours	float64
Primary_Use_Case	int64
Prompt_Engineering_Skill	int64
Tool_Diversity	float64
Paid_Subscription	int64
Traditional_Study_Hours	float64
Perceived_AI_Dependency	int64
Institutional_Policy	int64
Anxiety_Level_During_Exams	float64
Post_Semester_GPA	float64
Skill_Retention_Score	float64
Burnout_Risk_Level	float64

dtype: object

```
# Here I'm handling missing values. I'm using the mean since all the data are numerical and the missing values percentage is less than 5%
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean()
    df[column] = df[column].fillna(mean_value)

df.isnull().sum()
```

	0
Major_Category	0
Year_of_Study	0
Pre_Semester_GPA	0
Weekly_GenAI_Hours	0
Primary_Use_Case	0
Prompt_Engineering_Skill	0
Tool_Diversity	0
Paid_Subscription	0
Traditional_Study_Hours	0
Perceived_AI_Dependency	0
Institutional_Policy	0
Anxiety_Level_During_Exams	0
Post_Semester_GPA	0
Skill_Retention_Score	0
Burnout_Risk_Level	0

dtype: int64

```
print(f"Shape before removing outliers: {df.shape}")

# I'm using the interquartile range (IQR) to do handle the outliers.
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
```

```
# Define bounds for identifying outliers.
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the DataFrame to keep only rows where ALL columns are within bounds.
outlier_mask = ((df >= lower_bound) & (df <= upper_bound)).all(axis=1)

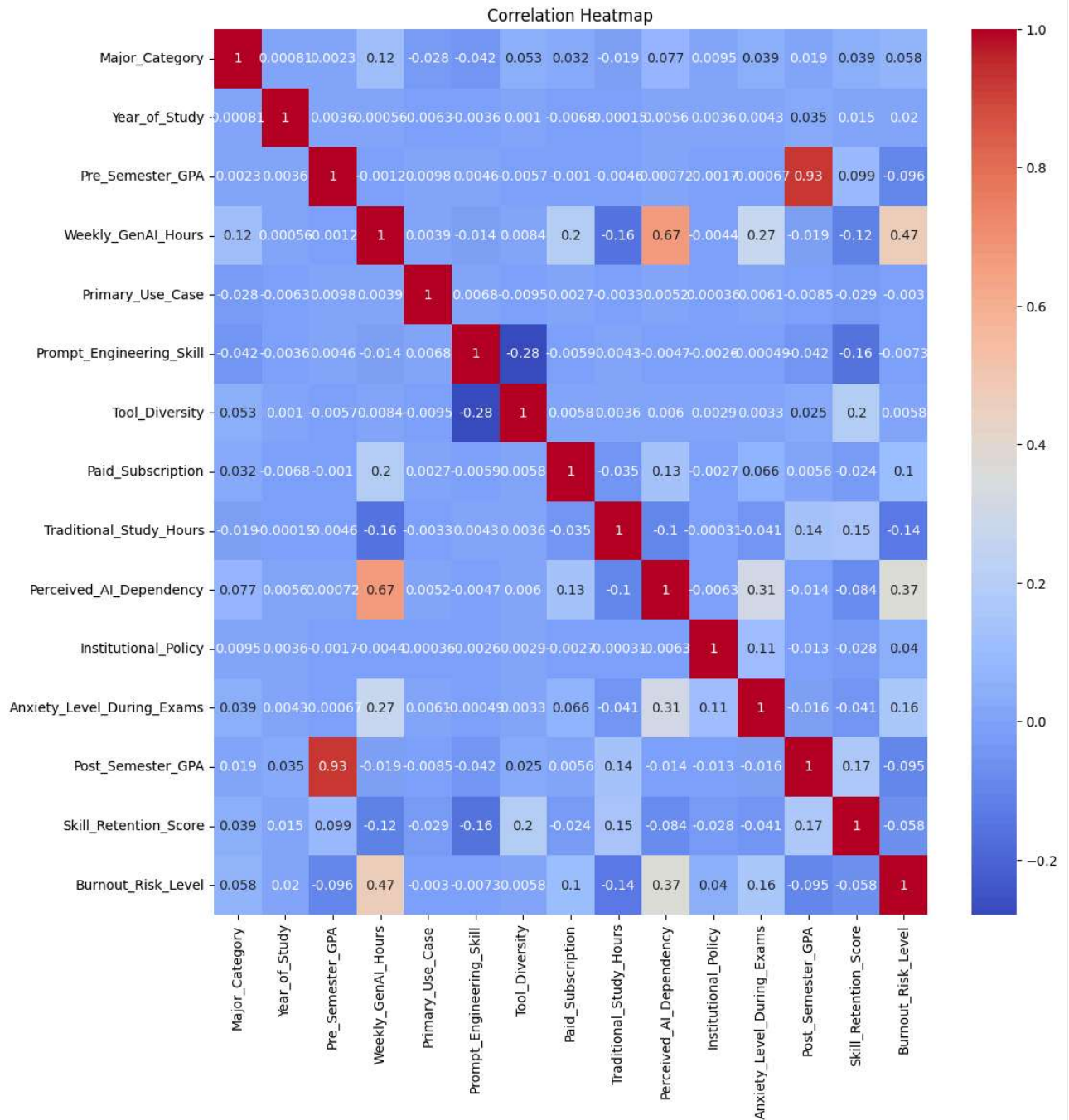
df_final = df[outlier_mask].copy()

print(f"Shape after removing outliers: {df_final.shape}")
```

```
Shape before removing outliers: (50000, 15)
Shape after removing outliers: (46716, 15)
```

```
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')

plt.show()
```

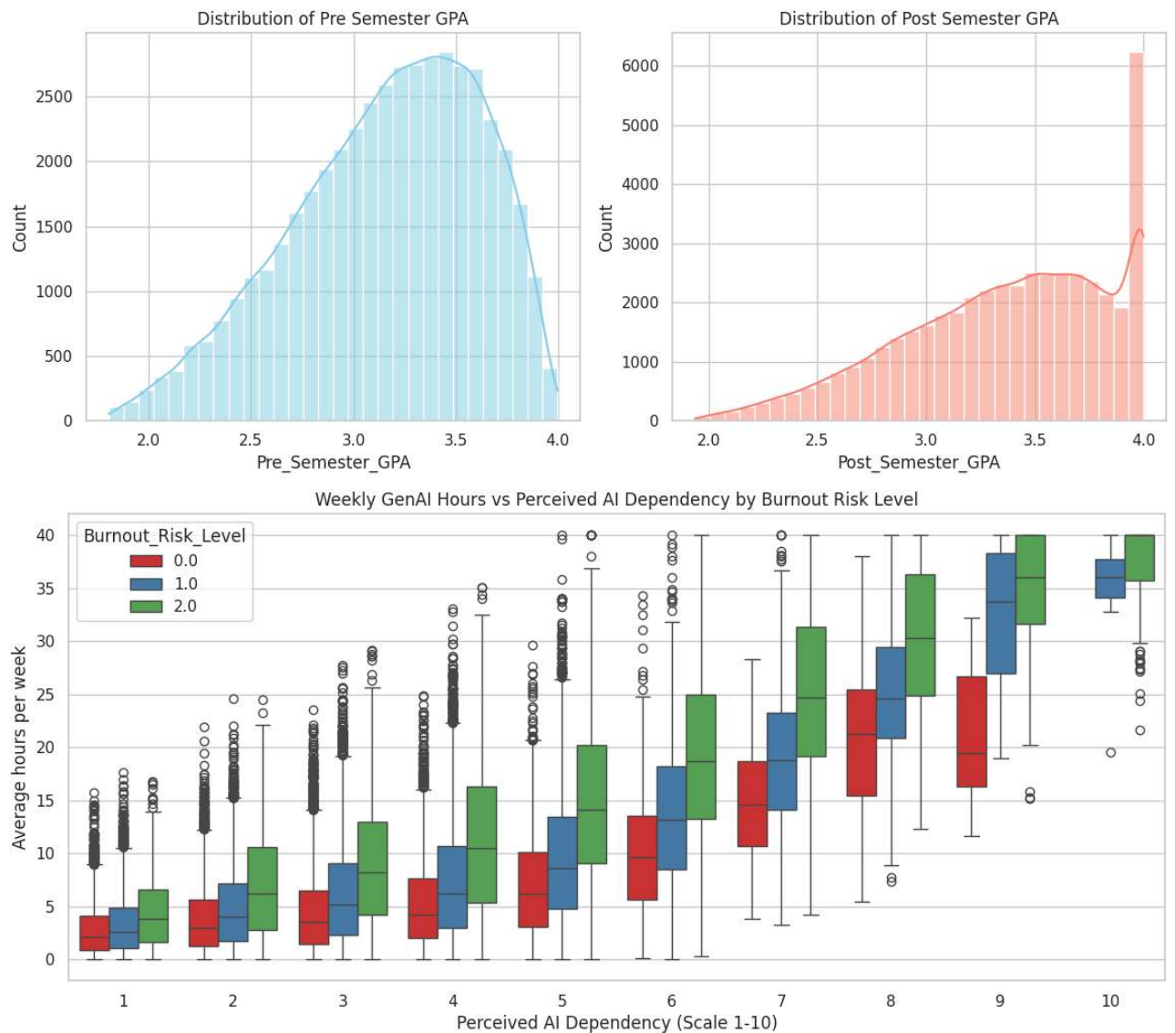


```
sns.set_theme(style="whitegrid")

# I'm using histograms to show the distribution of GPA before and after the semester, which allows us to compare academic performance.
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.histplot(df_final['Pre_Semester_GPA'], kde=True, bins=30, color='skyblue')
plt.title('Distribution of Pre Semester GPA')
```

```
plt.subplot(1, 2, 2)
sns.histplot(df_final['Post_Semester_GPA'], kde=True, bins=30, color='salmon')
plt.title('Distribution of Post Semester GPA')
plt.tight_layout()
plt.show()

# The boxplot here helps visualize the higher student self-rate GenAI dependency the more weekly hours they use GenAI which increase burnout
plt.figure(figsize=(14, 6))
sns.boxplot(
    x='Perceived_AI_Dependency',
    y='Weekly_GenAI_Hours',
    hue='Burnout_Risk_Level',
    data=df, palette='Set1'
)
plt.title('Weekly GenAI Hours vs Perceived AI Dependency by Burnout Risk Level')
plt.xlabel('Perceived AI Dependency (Scale 1-10)')
plt.ylabel('Average hours per week')
plt.show()
```



Start coding or generate with AI.