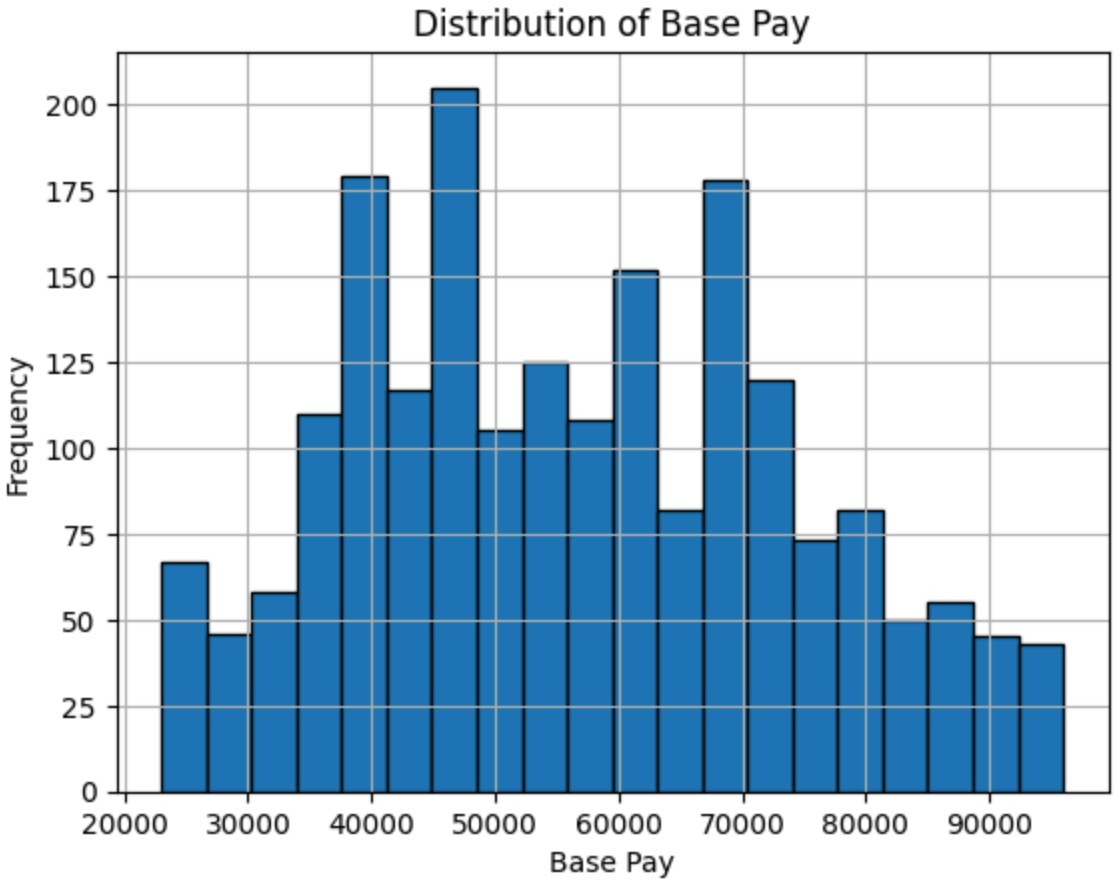
```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

```
1 df=pd.read_csv('/content/new_modified_dataset.csv')
2 # List of columns to scale
3 columns_to_scale = ['Grad', 'HSC', 'SSC']
4
5 # Divide each column by 2 to scale from 20 to 10
6 df[columns_to_scale] = df[columns_to_scale] / 2
7
8 # Optional: Round the values to eliminate any floating-point issues
9 df[columns_to_scale] = df[columns_to_scale].round(2)
10
11 df.head()
```

	Base Pay	Job Role	Skills	Base Pay Range	sex	age	Mother_education	Father_educati
0	72000	Teacher	Subject Knowledge, Communication, Patience, Cr...	30000-40000	Male	27	2	
1	50000	Salesperson	Communication, Negotiation, Customer Service, ...	20000-30000	Male	20	1	
2	38000	Salesperson	Communication, Negotiation, Customer Service, ...	20000-30000	Male	24	3	
3	72000	Engineer	Technical Skills, Mathematics, Project Managem...	40000-50000	Male	26	0	
4	65000	Engineer	Technical Skills, Mathematics, Project Managem...	40000-50000	Male	26	0	

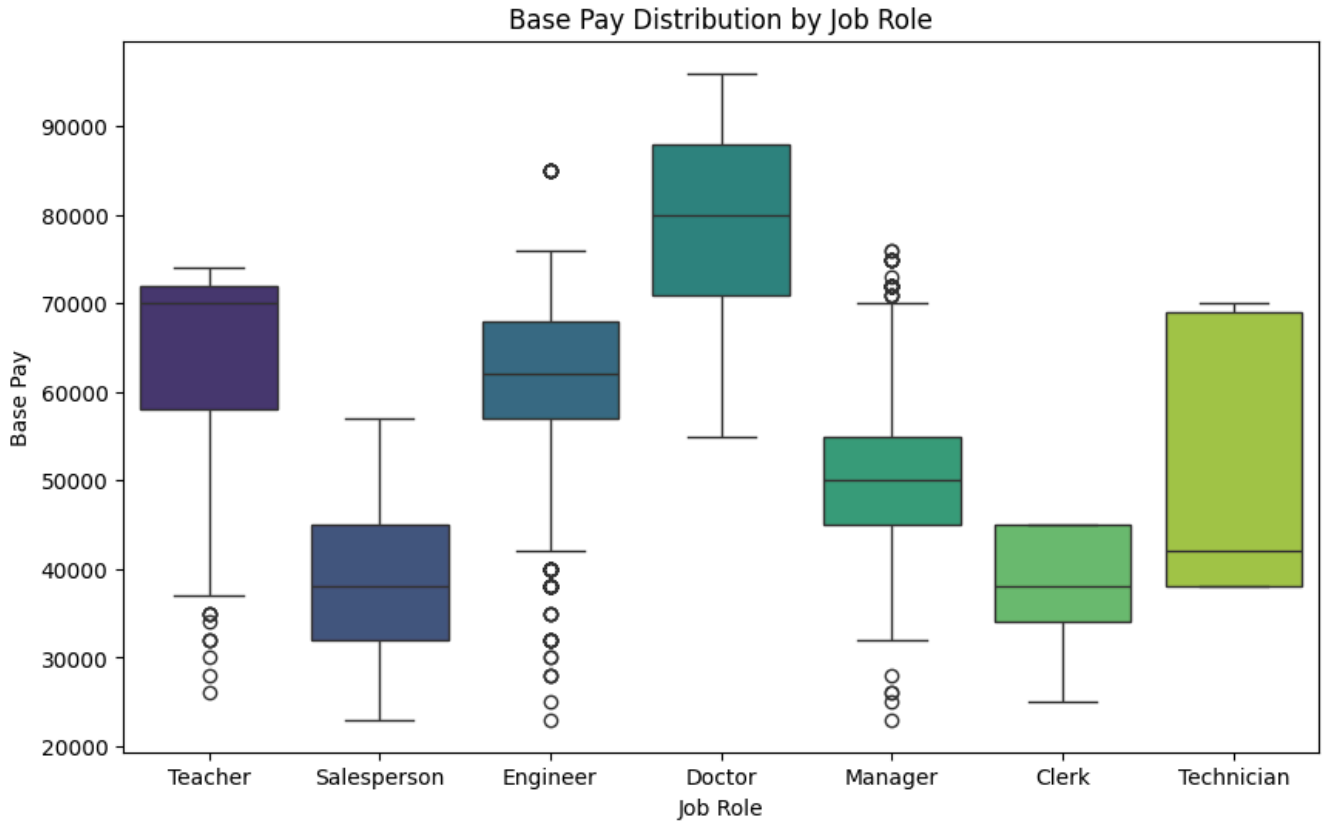
```
job_role_categories = pd.Categorical(df['Job Role'])
job_role_codes = job_role_categories.codes
job_role_names = job_role_categories.categories

# Create the scatter plot using job role codes for color mapping
scatter = plt.scatter(x=df['SSC'], y=df['HSC'],
                      c=job_role_codes,
                      cmap='viridis',
                      s=32, alpha=.8)

# Customize plot appearance
plt.gca().spines[['top', 'right']].set_visible(False)
plt.xlabel('SSC')
plt.ylabel('HSC')
plt.title('Scatter Plot of SSC vs. HSC by Job Role')

for role_code, role_name in enumerate(job_role_names):
    plt.scatter([], [], color=scatter.cmap(scatter.norm(role_code)), label=role_name)
plt.legend(title='Job Role', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout(rect=[0, 0, 0.85, 1])

# Show plot
plt.show()

```

```
df['Base Pay'].hist(bins=20, edgecolor='black')
plt.title('Distribution of Base Pay')
plt.xlabel('Base Pay')
plt.ylabel('Frequency')
plt.show()
```



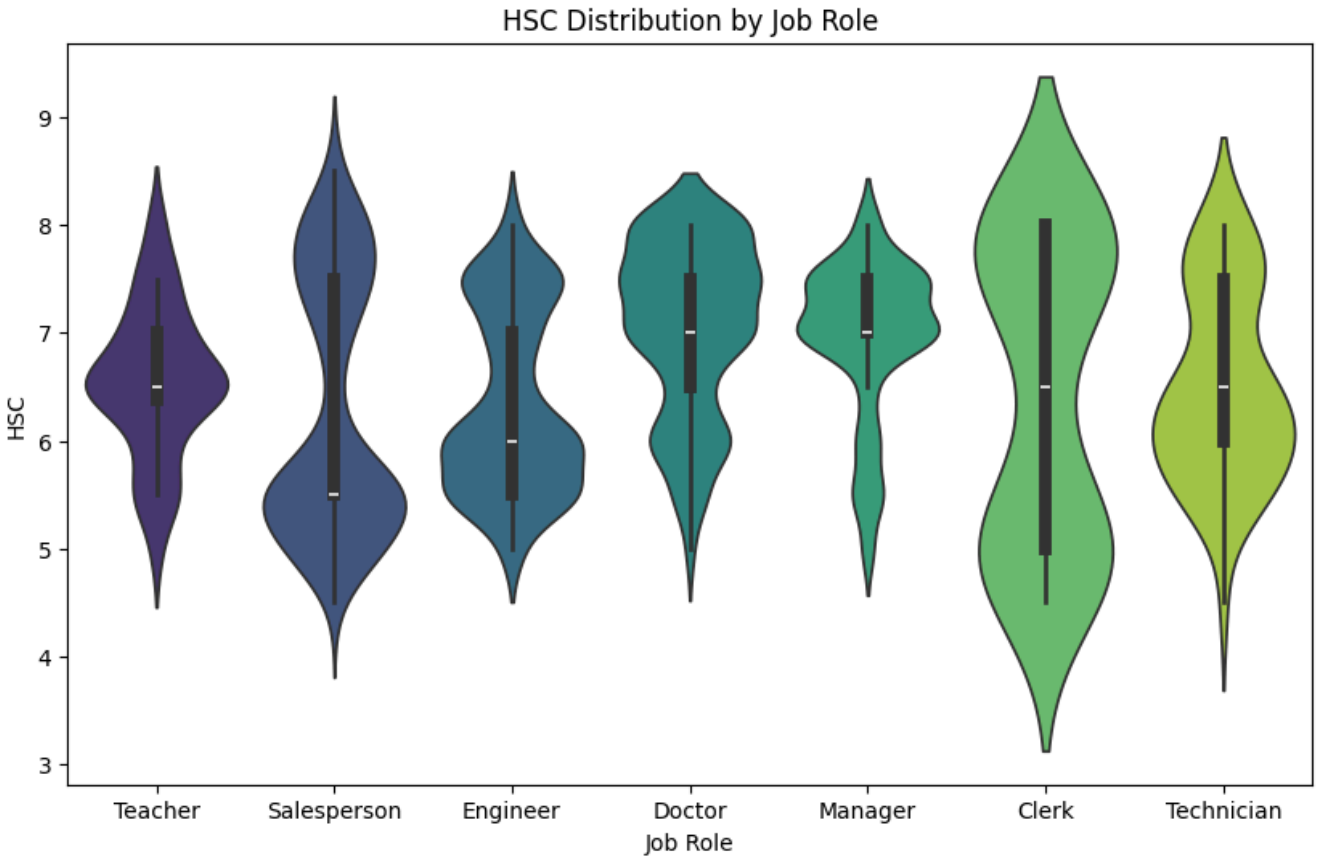

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Job Role', y='Base Pay', palette='viridis')
plt.title('Base Pay Distribution by Job Role')
plt.show()
```

```
<ipython-input-5-81334bad51fb>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.boxplot(data=df, x='Job Role', y='Base Pay', palette='viridis')
```

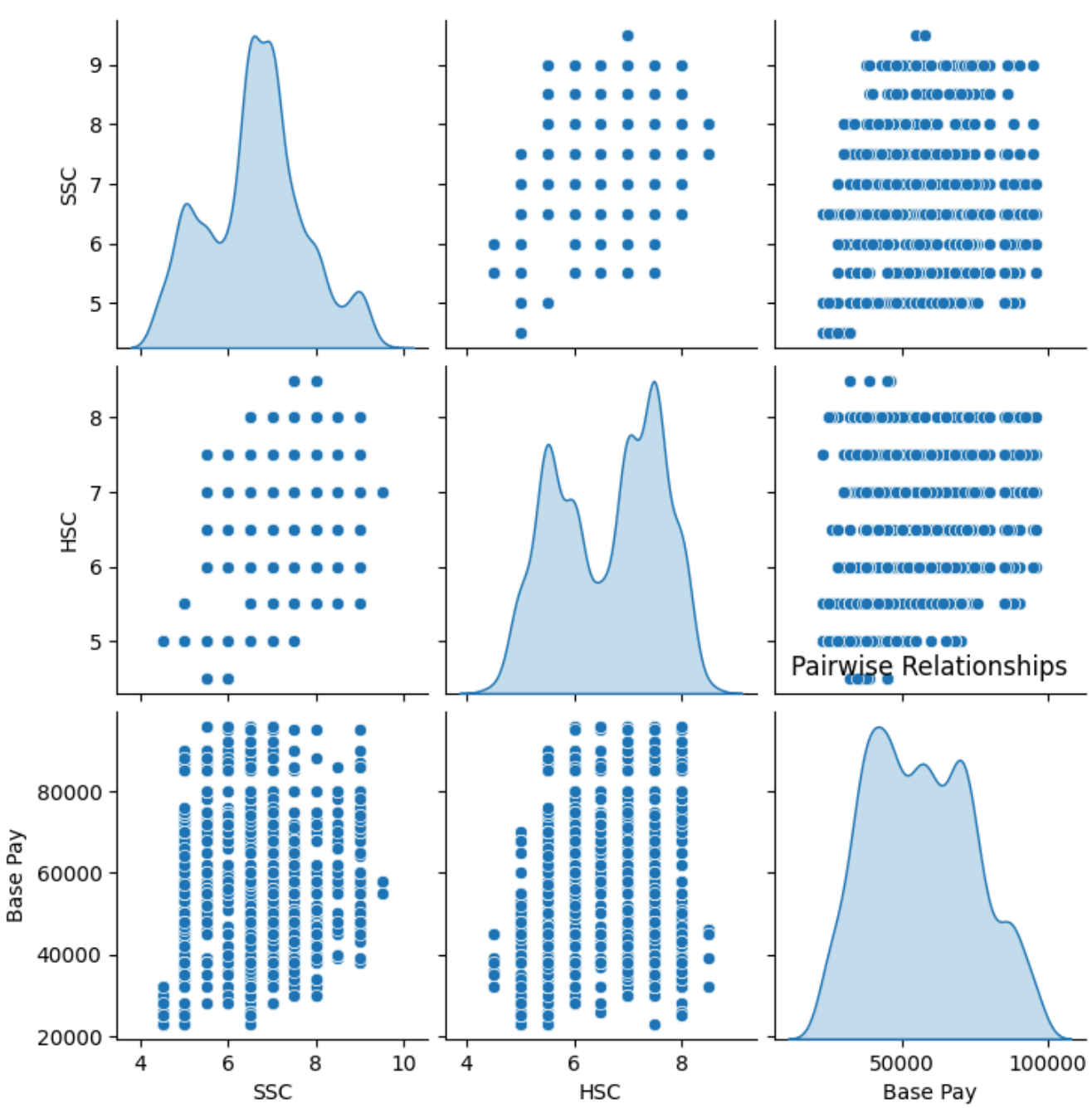



```
1 plt.figure(figsize=(10, 6))
2 sns.violinplot(data=df, x='Job Role', y='HSC', palette='viridis')
3 plt.title('HSC Distribution by Job Role')
4 plt.show()
```

```
<ipython-input-6-b37f067712f2>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.violinplot(data=df, x='Job Role', y='HSC', palette='viridis')
```

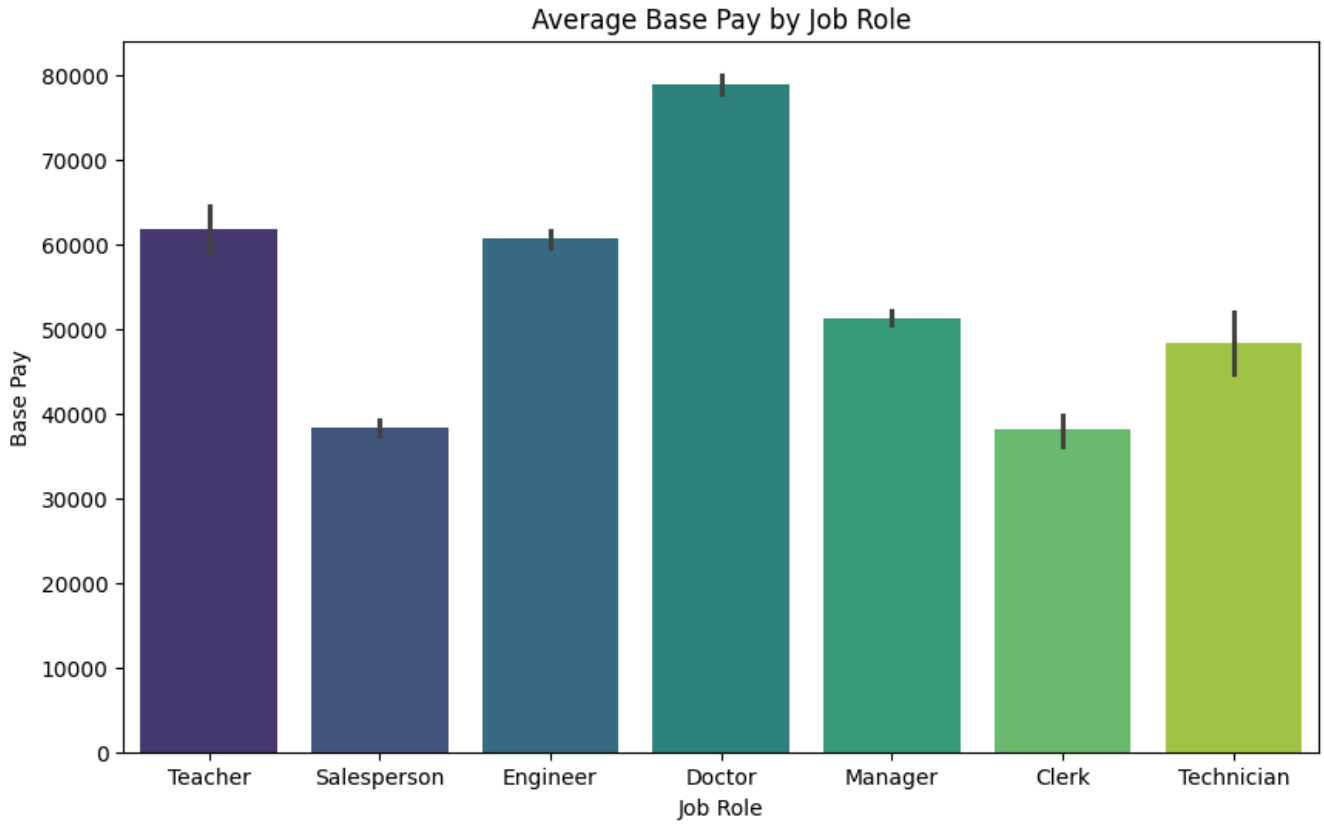



```
1 sns.pairplot(df[['SSC', 'HSC', 'Base Pay']], diag_kind='kde')
2 plt.title('Pairwise Relationships')
3 plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='Job Role', y='Base Pay', palette='viridis')
plt.title('Average Base Pay by Job Role')
plt.show()
```

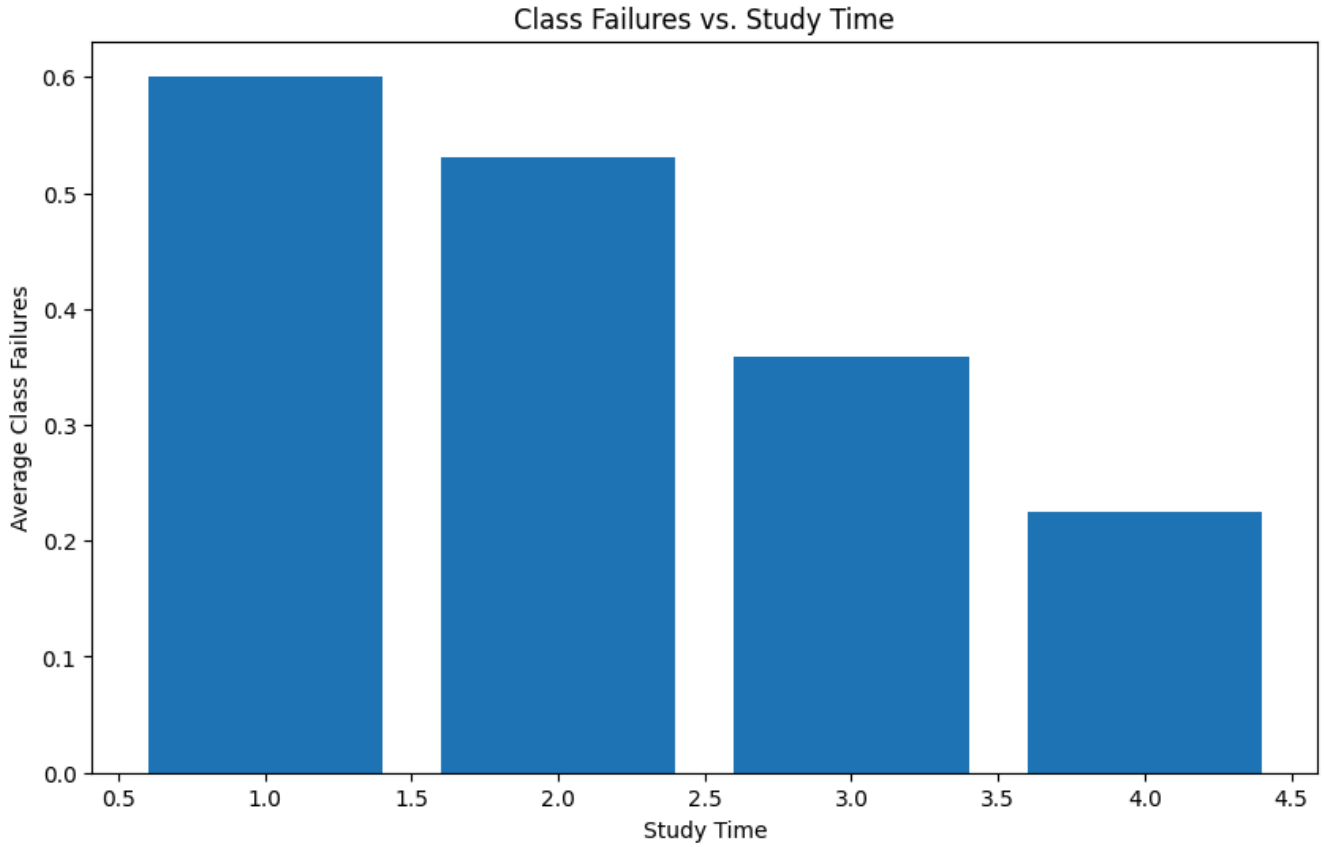
```
<ipython-input-8-1bdf2e4f01df>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.barplot(data=df, x='Job Role', y='Base Pay', palette='viridis')
```



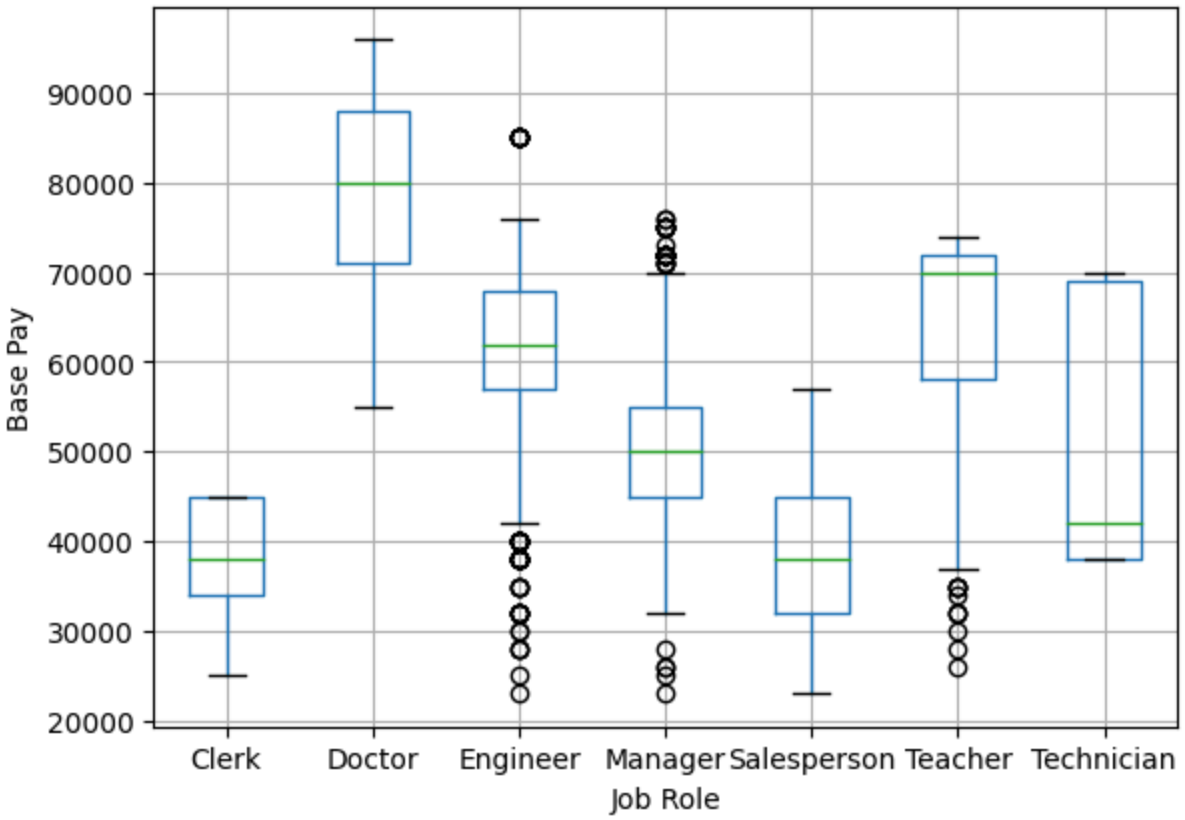

```
df.columns
```

```
Index(['Base Pay', 'Job Role', 'Skills', 'Base Pay Range', 'sex', 'age',
       'Mother_education', 'Father_education', 'Mother_job', 'Father_job',
       'studytime', 'backlogs', 'tuition', 'pursue_higher_studies',
       'Internet_usage', 'absences', 'SSC', 'HSC', 'Grad'],
      dtype='object')
```

```
study_time_groups = df.groupby('studytime')['backlogs'].mean()

plt.figure(figsize=(10, 6))
plt.bar(study_time_groups.index, study_time_groups.values)
plt.xlabel('Study Time')
plt.ylabel('Average Class Failures')
_ = plt.title('Class Failures vs. Study Time')
```




```
df.boxplot(column='Base Pay', by='Job Role')
plt.xlabel('Job Role')
plt.ylabel('Base Pay')
_ = plt.title('Distribution of Base Pay by Job Role')
```




```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Base Pay              2000 non-null   int64  
 1   Job Role              2000 non-null   object 
 2   Skills                2000 non-null   object 
 3   Base Pay Range        2000 non-null   object 
 4   sex                   2000 non-null   object 
 5   age                   2000 non-null   int64  
 6   Mother_education      2000 non-null   int64  
 7   Father_education      2000 non-null   int64  
 8   Mother_job            2000 non-null   object 
 9   Father_job            2000 non-null   object 
 10  studytime             2000 non-null   int64  
 11  backlogs              2000 non-null   int64  
 12  tuition               2000 non-null   object 
 13  pursue_higher_studies 2000 non-null   object 
 14  Internet_usage        2000 non-null   object 
 15  absences              2000 non-null   int64  
 16  SSC                   2000 non-null   float64
 17  HSC                   2000 non-null   float64
 18  Grad                  2000 non-null   float64
dtypes: float64(3), int64(7), object(9)
```

```
memory usage: 297.0+ KB
```

```
1 df.describe()
```

	Base Pay	age	Mother_education	Father_education	studytime	bac
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.0
mean	56594.500000	25.849000	1.407500	1.451000	2.133000	0.4
std	17880.350198	2.426962	1.041627	0.937571	1.000906	0.5
min	23000.000000	20.000000	0.000000	0.000000	1.000000	0.0
25%	42000.000000	24.000000	1.000000	1.000000	1.000000	0.0
50%	55000.000000	26.000000	1.000000	1.000000	2.000000	0.0
75%	70000.000000	28.000000	2.000000	2.000000	3.000000	1.0
max	96000.000000	31.000000	3.000000	3.000000	4.000000	2.0

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

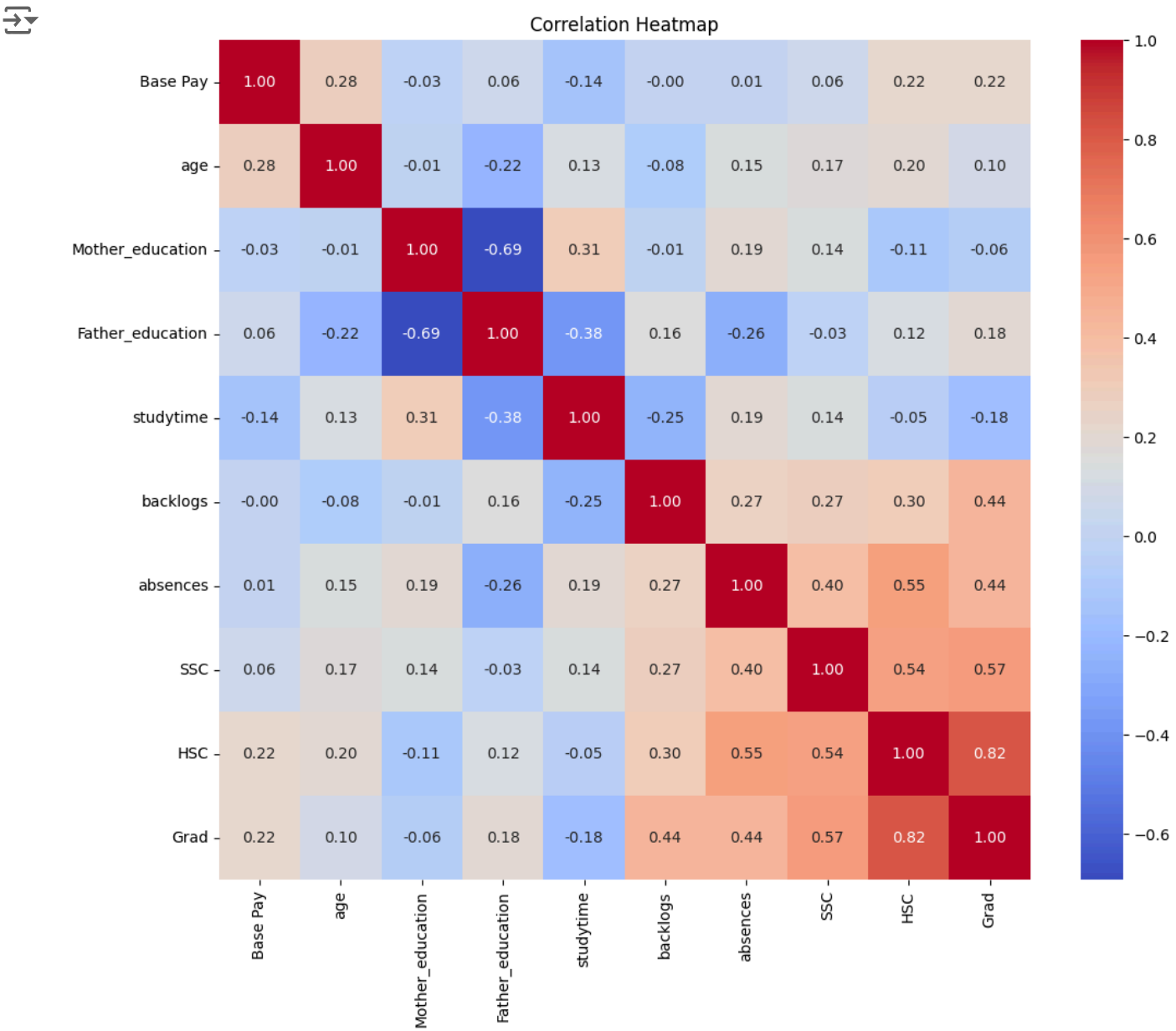
	0
Base Pay	0
Job Role	0
Skills	0
Base Pay Range	0
sex	0
age	0
Mother_education	0
Father_education	0
Mother_job	0
Father_job	0
studytime	0
backlogs	0
tuition	0
pursue_higher_studies	0
Internet_usage	0
absences	0
SSC	0
HSC	0
Grad	0

dtype: int64

```
# from sklearn.preprocessing import LabelEncoder
# label_encoder = LabelEncoder()

# for column in df.columns:
#   if df[column].dtype == 'object':  # Check if the column is categorical
#     df[column] = label_encoder.fit_transform(df[column])
```

```
# df.head()
```

```
numerical_df = df.select_dtypes(include=np.number)

# Calculate correlation matrix
corr_matrix = numerical_df.corr()

# Create heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

```


Correlation Heatmap
Base Pay 1.00 0.28 -0.03 0.06 -0.14 -0.00 0.01 0.06 0.22 0.22
age 0.28 1.00 -0.01 -0.22 0.13 -0.08 0.15 0.17 0.20 0.10
Mother_education -0.03 -0.01 1.00 -0.69 0.31 -0.01 0.19 0.14 -0.11 -0.06
Father_education 0.06 -0.22 -0.69 1.00 -0.38 0.16 -0.26 -0.03 0.12 0.18
studytime -0.14 0.13 0.31 -0.38 1.00 -0.25 0.19 0.14 -0.05 -0.18
backlogs -0.00 -0.08 -0.01 0.16 -0.25 1.00 0.27 0.27 0.30 0.44
absences 0.01 0.15 0.19 -0.26 0.19 0.27 1.00 0.40 0.55 0.44
SSC 0.06 0.17 0.14 -0.03 0.14 0.27 0.40 1.00 0.54 0.57
HSC 0.22 0.20 -0.11 0.12 -0.05 0.30 0.55 0.54 1.00 0.82
Grad 0.22 0.10 -0.06 0.18 -0.18 0.44 0.44 0.57 0.82 1.00
Base Pay
age
Mother_education
Father_education
studytime
backlogs
absences
SSC
HSC
Grad
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```
1 df.duplicated().value_counts()
```

	count
False	1488
True	512

dtype: int64

```
1 df['Job Role'].value_counts()
```

Job Role	count
Engineer	472
Salesperson	460
Manager	449
Doctor	425
Teacher	108
Technician	49
Clerk	37

dtype: int64

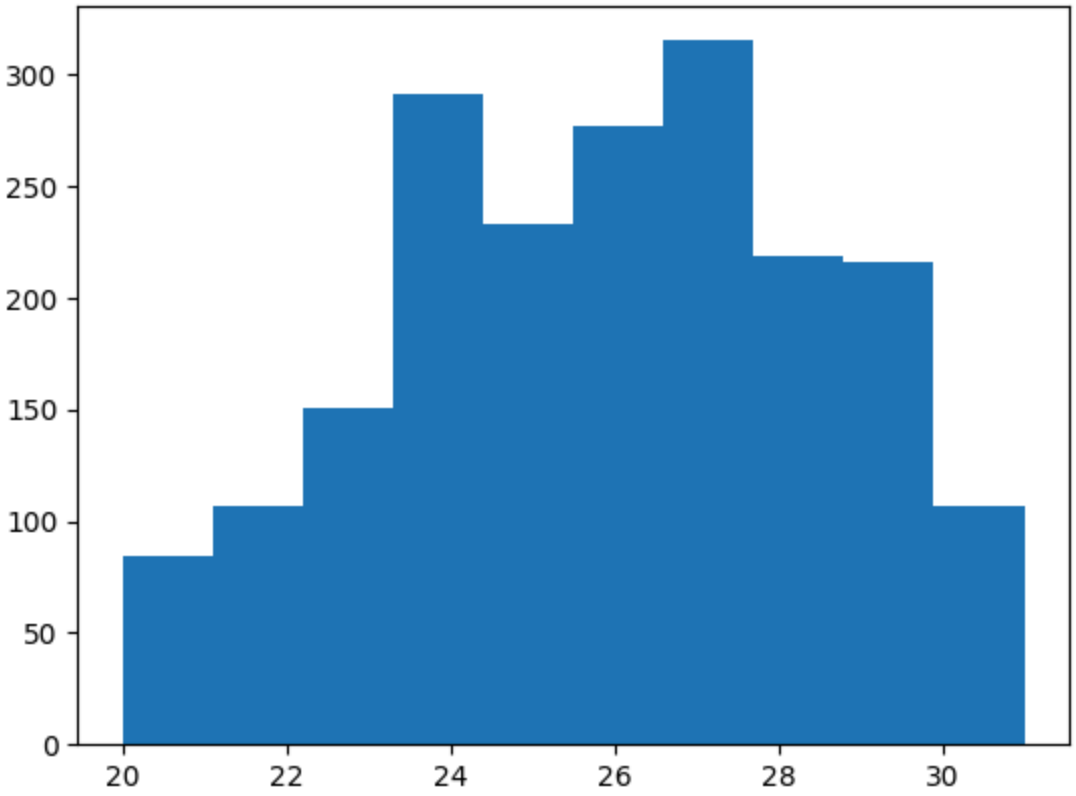
```
1 df['Base Pay Range'].value_counts()
```

Base Pay Range	count
40000-50000	472
20000-30000	460
70000-90000	449
60000-80000	425
30000-40000	108
10000-20000	49
10000-15000	37

dtype: int64

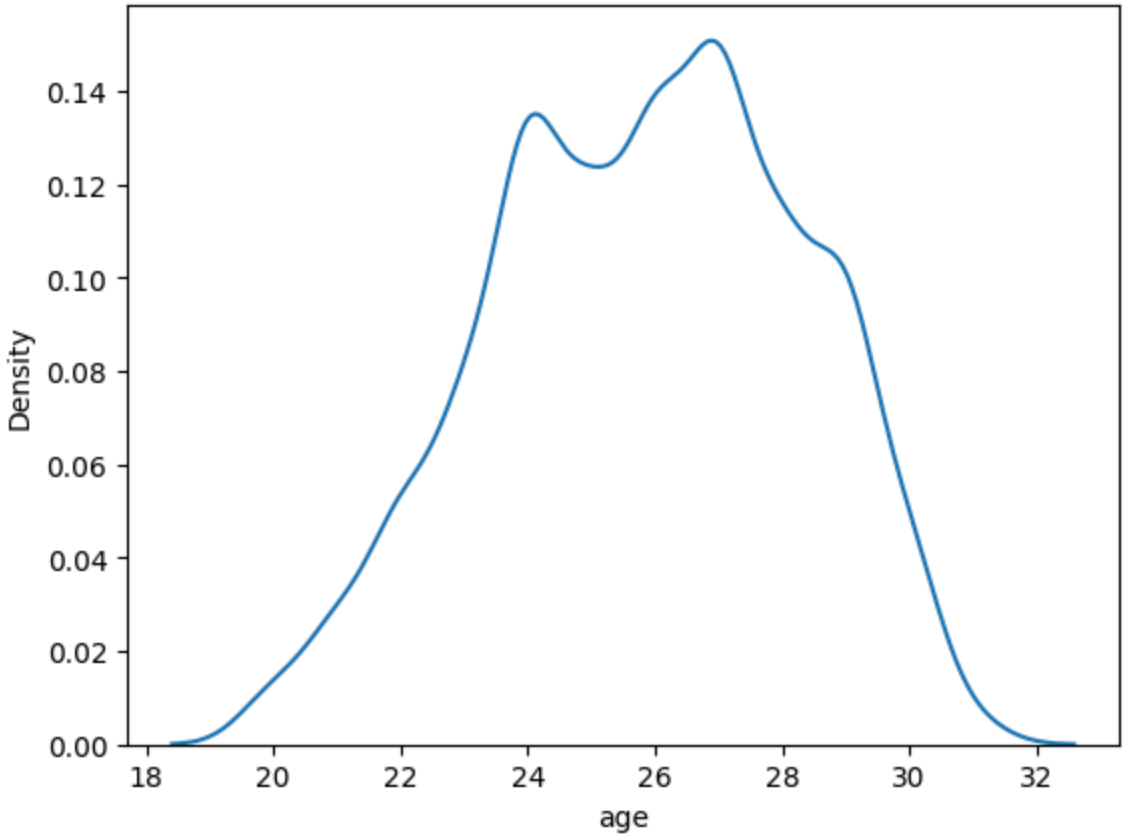
```
1 plt.hist(df['age'])
```

```
(array([ 84., 107., 151., 291., 233., 277., 315., 219., 216., 107.]),
 array([20. , 21.1, 22.2, 23.3, 24.4, 25.5, 26.6, 27.7, 28.8, 29.9, 31. ]),
 <BarContainer object of 10 artists>)
```



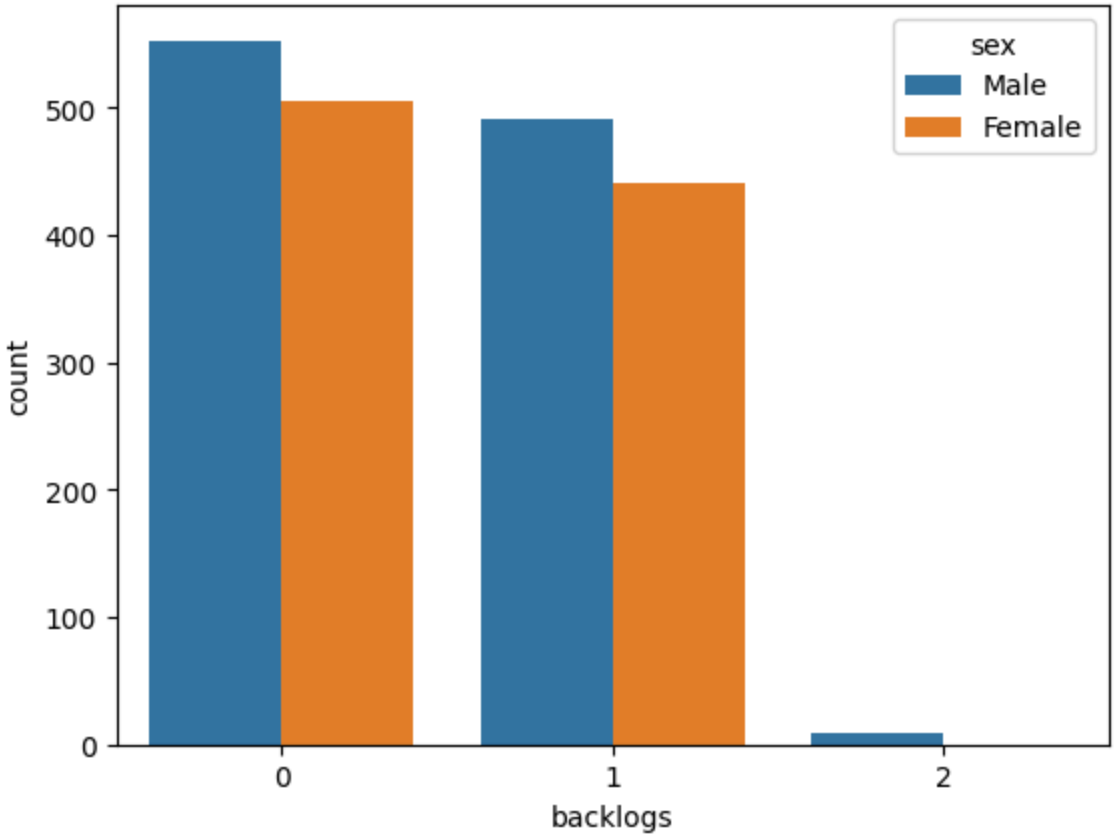

```
1 sns.kdeplot(df['age'])
```

```
<Axes: xlabel='age', ylabel='Density'>
```



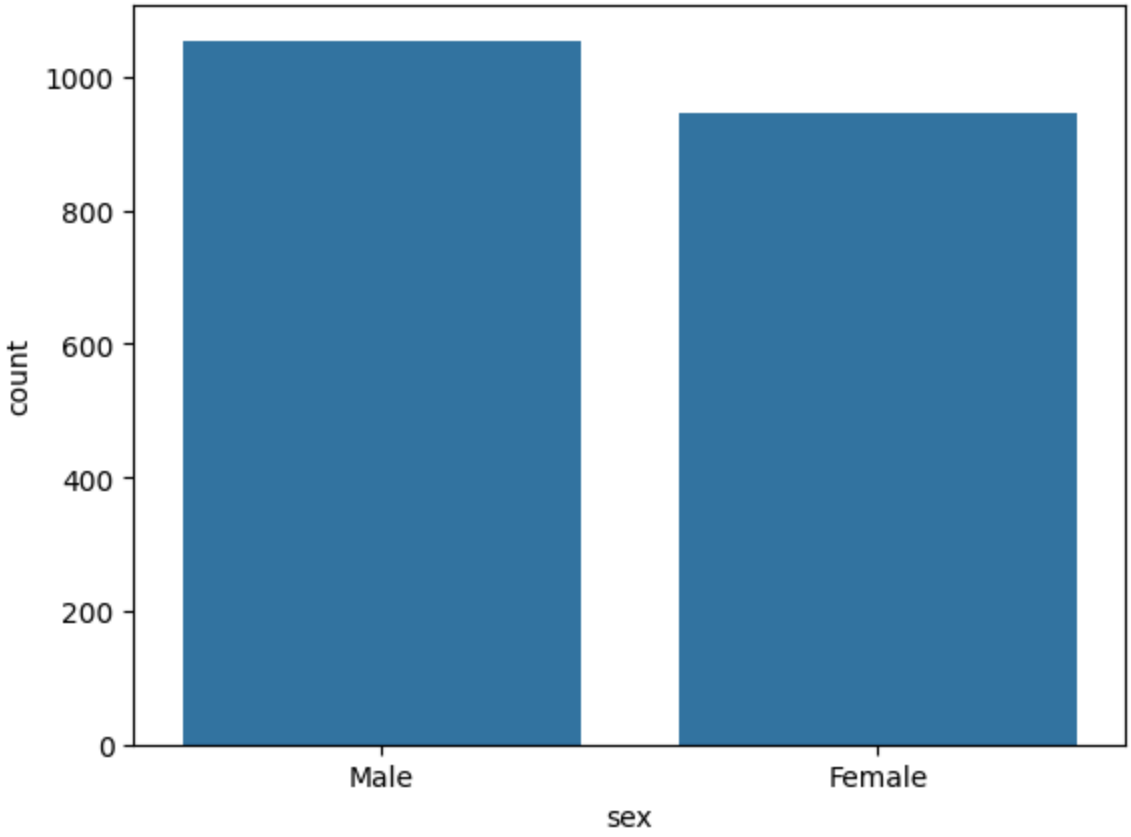

```
1 sns.countplot(data=df, x="backlogs", hue="sex")
```

```
<Axes: xlabel='backlogs', ylabel='count'>
```



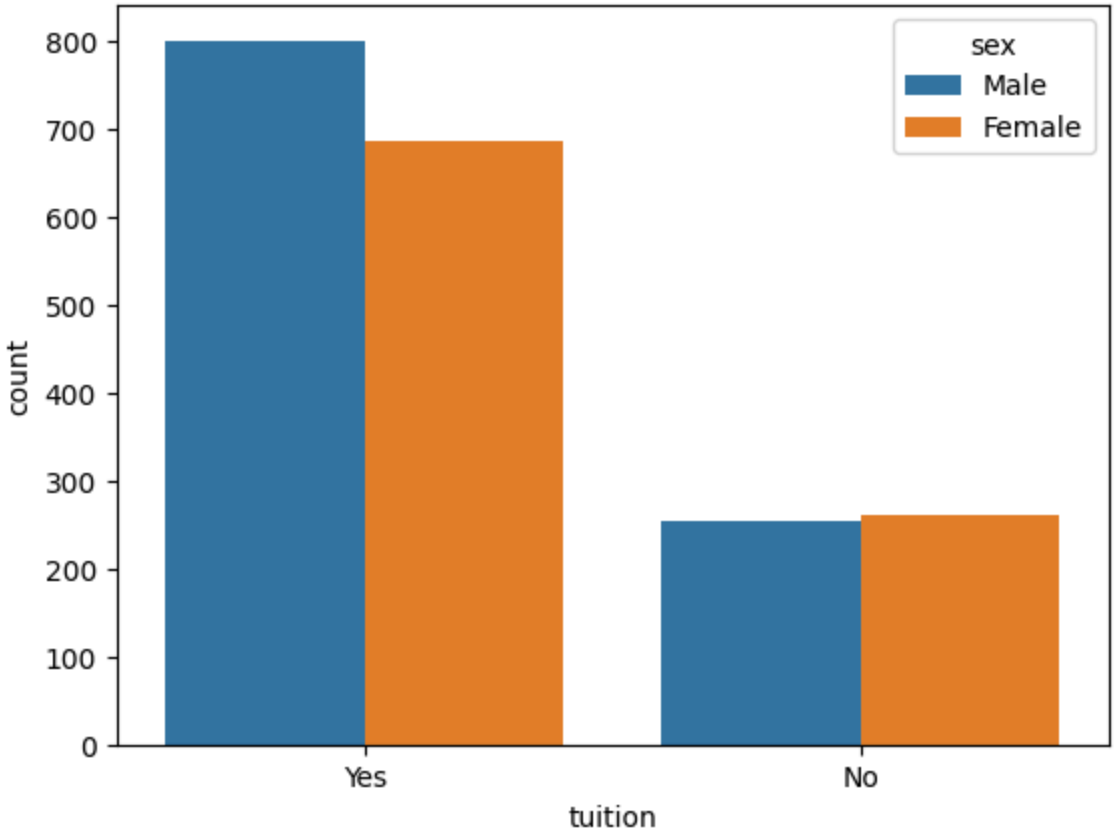

```
1 sns.countplot(data=df, x="sex")
```

```
<Axes: xlabel='sex', ylabel='count'>
```




```
1 sns.countplot(data=df, x="tuition", hue="sex")
```

```
<Axes: xlabel='tuition', ylabel='count'>
```

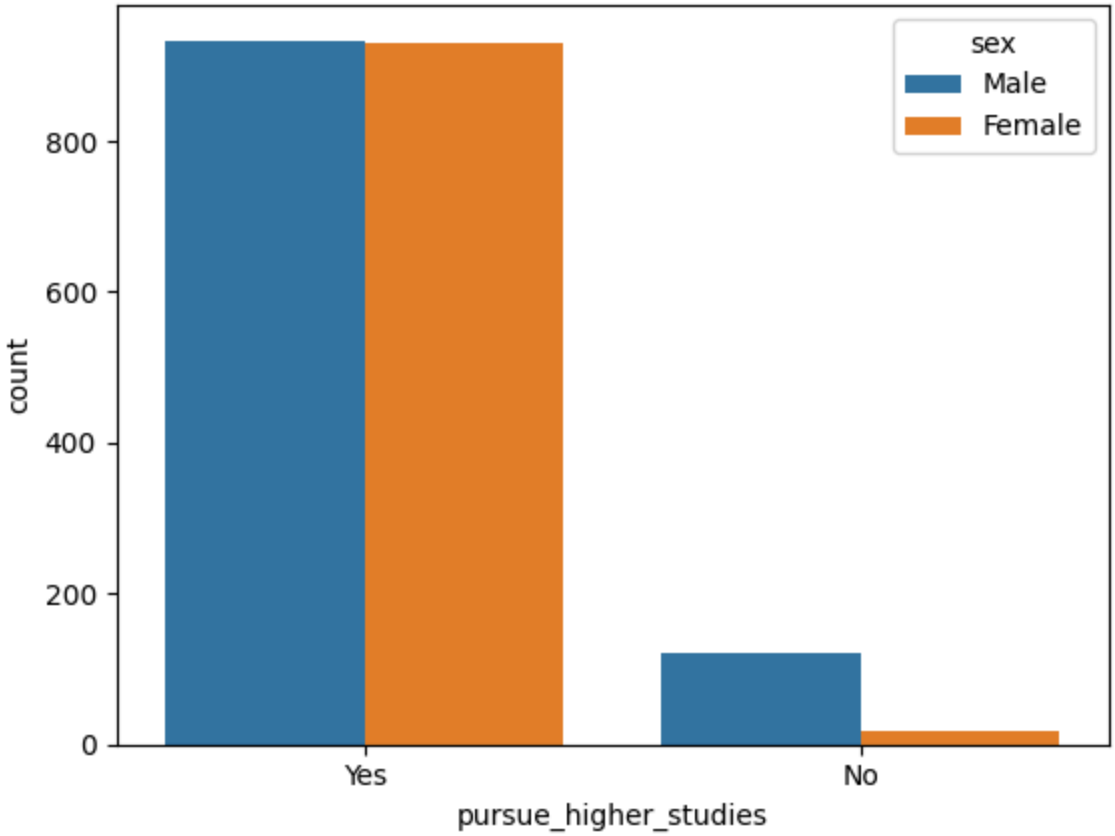



```
df.columns
```

```
Index(['Base Pay', 'Job Role', 'Skills', 'Base Pay Range', 'sex', 'age',
       'Mother_education', 'Father_education', 'Mother_job', 'Father_job',
       'studytime', 'backlogs', 'tuition', 'pursue_higher_studies',
       'Internet_usage', 'absences', 'SSC', 'HSC', 'Grad'],
      dtype='object')
```

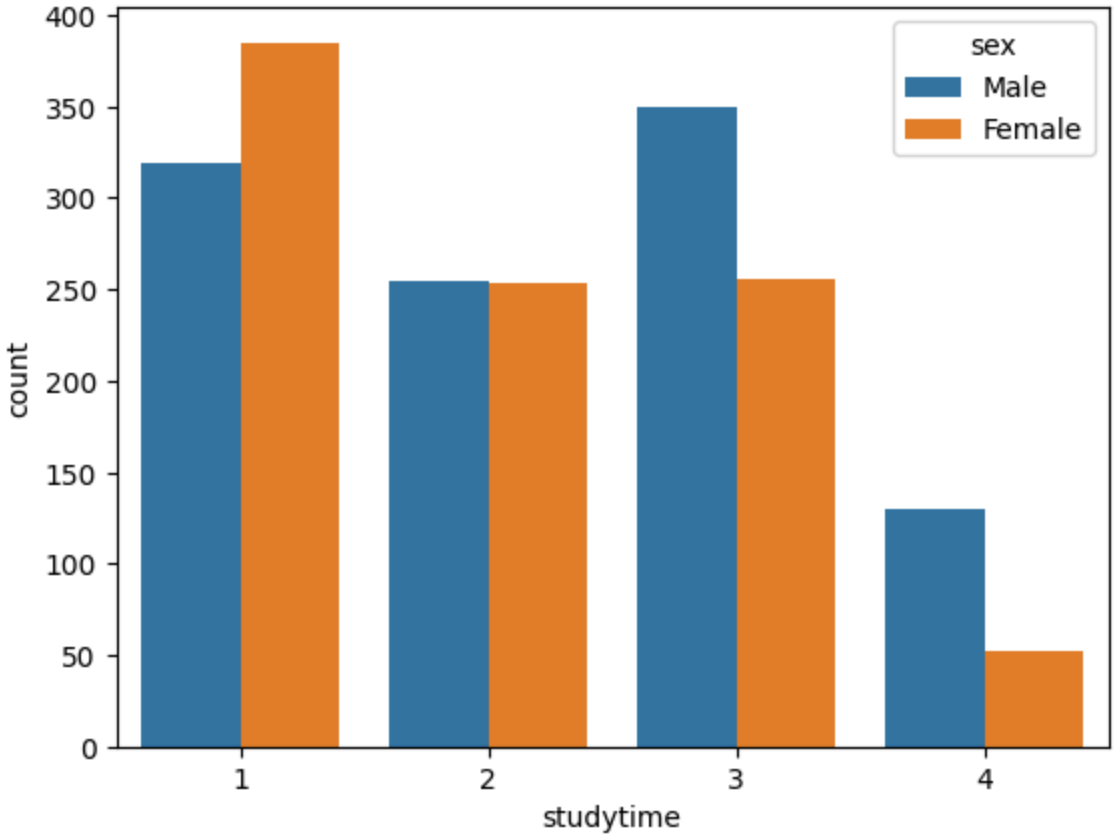
```
sns.countplot(data=df, x="pursue_higher_studies", hue="sex")
```

```
<Axes: xlabel='pursue_higher_studies', ylabel='count'>
```

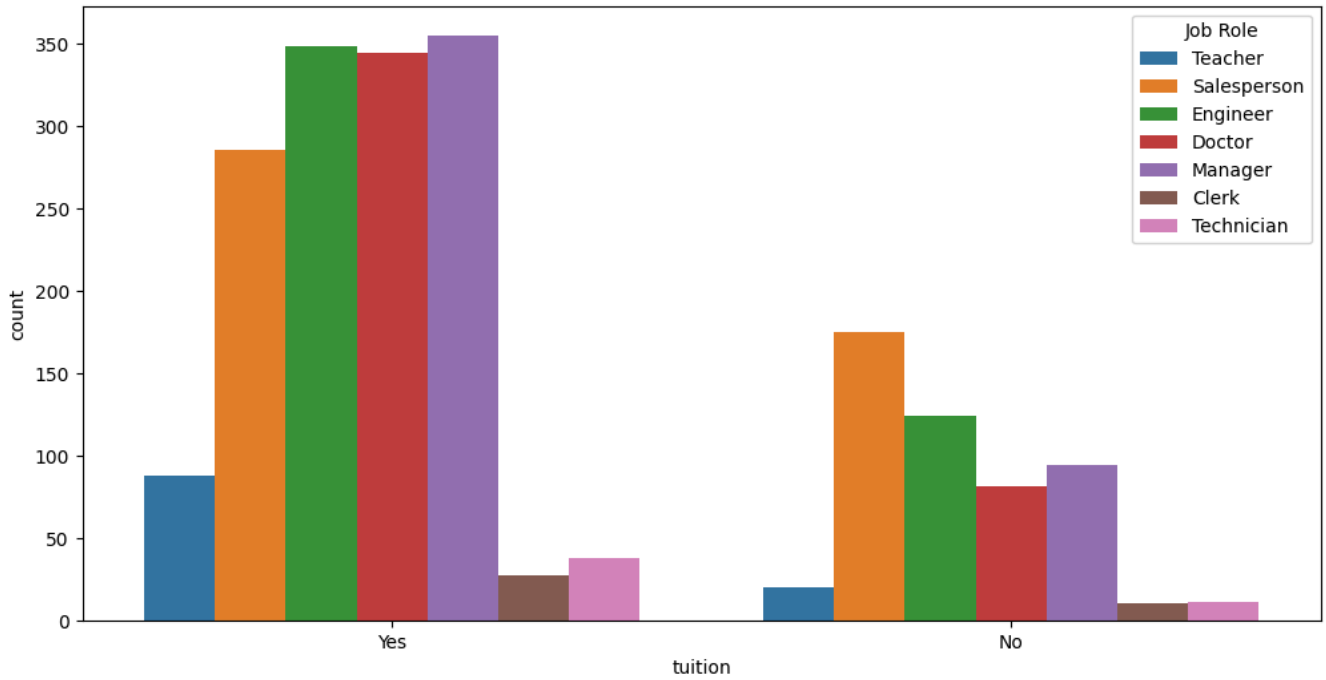



```
1 sns.countplot(data=df, x="studytime", hue="sex")
```

```
<Axes: xlabel='studytime', ylabel='count'>
```

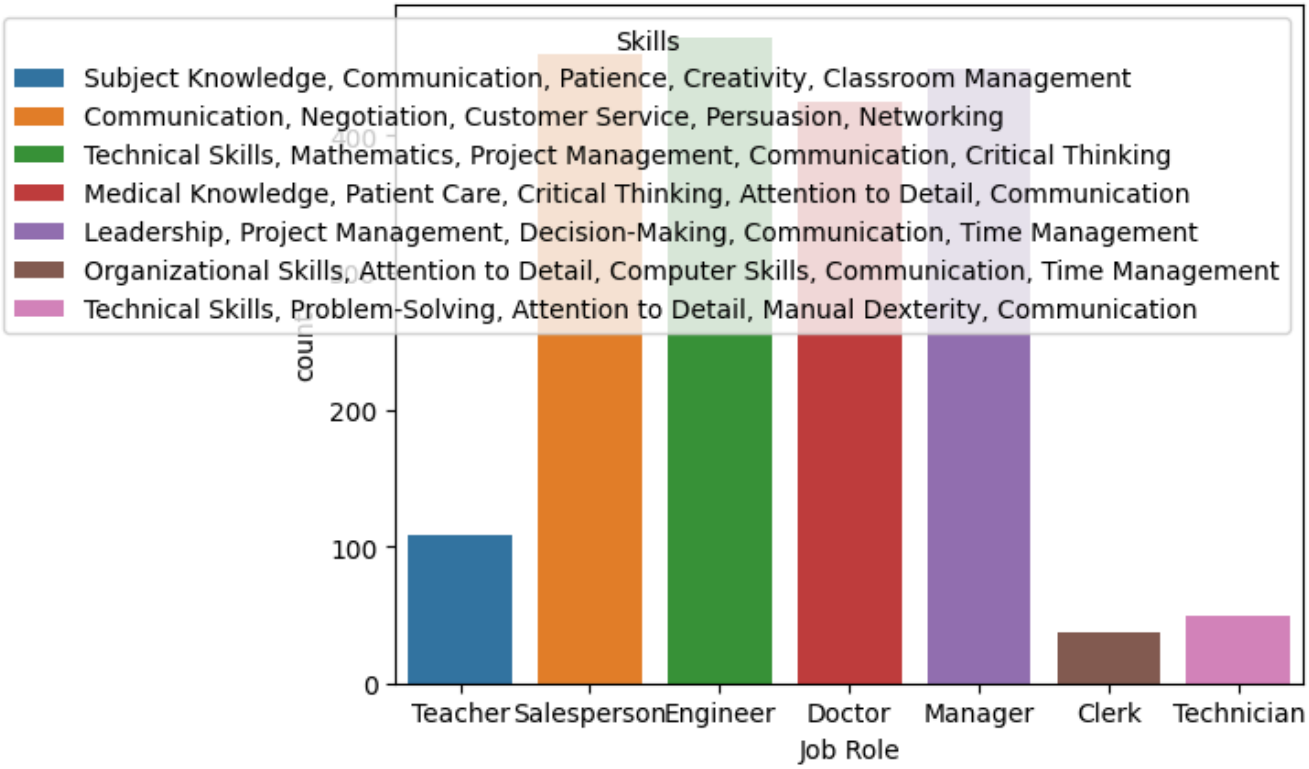



```
1 plt.figure(figsize=(12, 6))
2
3 # Create the countplot
4 sns.countplot(data=df, x="tuition", hue="Job Role")
5
6 # Show the plot
7 plt.show()
```



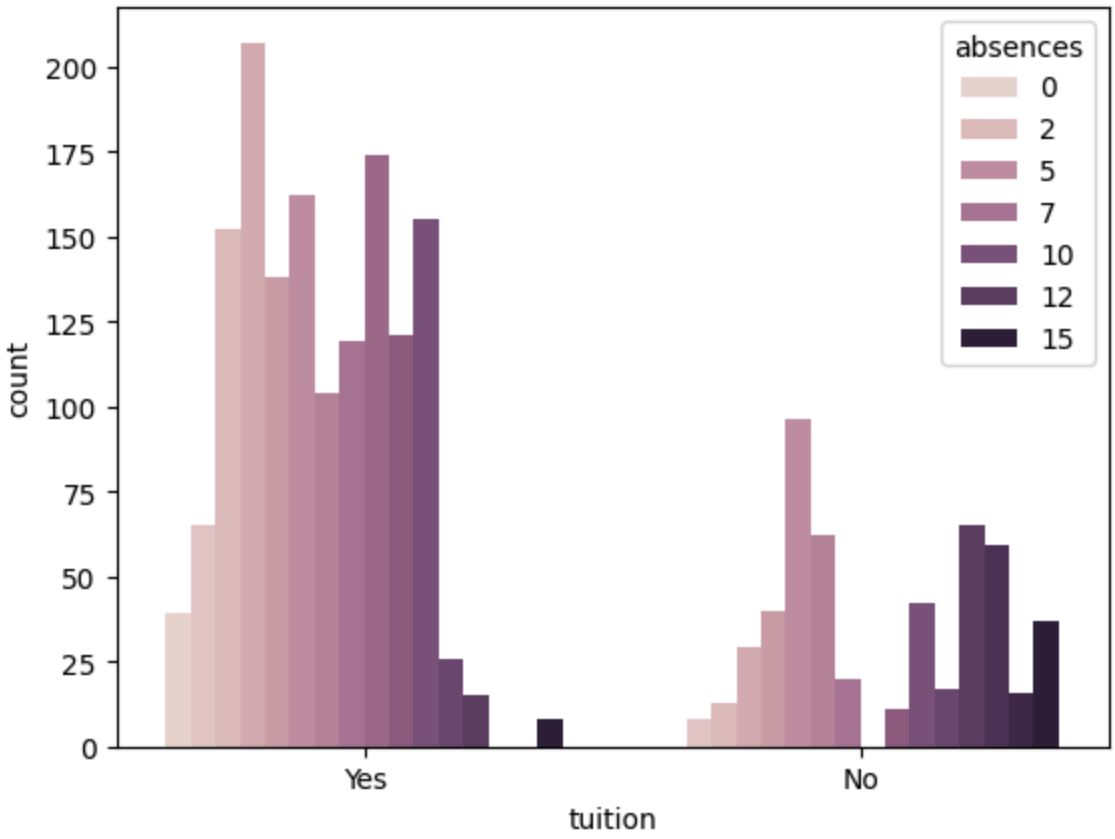

```
1 sns.countplot(data=df, x="Job Role", hue="Skills")
```

```
<Axes: xlabel='Job Role', ylabel='count'>
```




```
1 sns.countplot(data=df, x="tuition", hue="absences")
```

```
<Axes: xlabel='tuition', ylabel='count'>
```




```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Job Role'] = le.fit_transform(df['Job Role'])
```

```
label_encoder = LabelEncoder()

for column in df.columns:
  if df[column].dtype == 'object':  # Check if the column is categorical
    df[column] = label_encoder.fit_transform(df[column])
```

PREDICTING JOB ROLE BASED ON STUDENT DEMOGRAPHICS

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
X = df.drop(columns=['Job Role'])
y = df['Job Role']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100)

# Train the model
rf_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

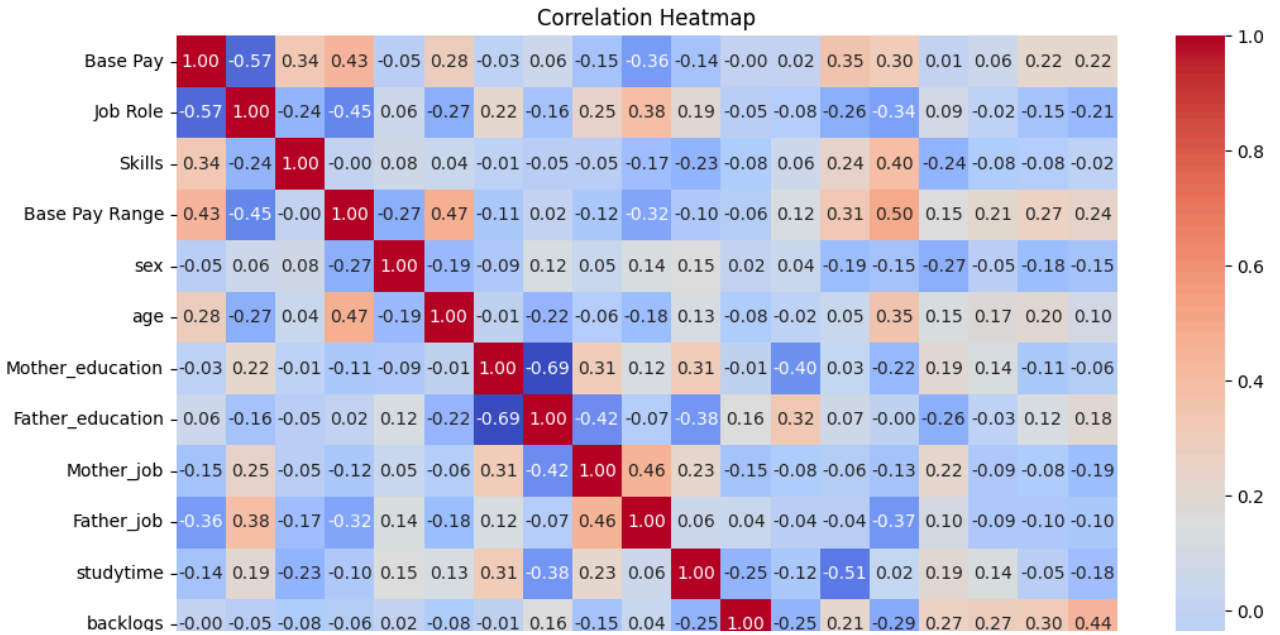
```
# Predict on the test set
y_pred = rf_classifier.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.00

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        82
           2       1.00      1.00      1.00        79
           3       1.00      1.00      1.00        93
           4       1.00      1.00      1.00        96
           5       1.00      1.00      1.00        29
           6       1.00      1.00      1.00        12

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400


Confusion Matrix:
[[ 9  0  0  0  0  0  0]
```

```
[ 0 82  0  0  0  0  0]
[ 0  0 79  0  0  0  0]
[ 0  0  0 93  0  0  0]
[ 0  0  0  0 96  0  0]
[ 0  0  0  0  0 29  0]
[ 0  0  0  0  0  0 12]]
```

```
corr_matrix = df.corr()

# Create heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

```




```
1 df.shape
```

```
(2000, 19)
```

PREDICTING THE MONTHLY INCOME BASED ON VARIOUS STUDENT DEMOGRAPHIES

```
1 df.columns
```

```
Index(['Base Pay', 'Job Role', 'Skills', 'Base Pay Range', 'sex', 'age',
```