```
In [6]: !pip install pyreadstat
        import pyreadstat as pyd
```

```
Requirement already satisfied: pyreadstat in c:\users\ryan\anaconda3\lib\site-packages (1.2.8)
Requirement already satisfied: pandas>=1.2.0 in c:\users\ryan\anaconda3\lib\site-packages (from pyreadstat) (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\ryan\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (1.2
6.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\ryan\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreads
tat) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\ryan\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (202
4.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\ryan\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (20
23.3)
Requirement already satisfied: six>=1.5 in c:\users\ryan\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.2.
0->pyreadstat) (1.16.0)
```

```
In [10]: df1, meta = pyd.read_sas7bdat('./retention.sas7bdat')
```

Q1

```
In [13]: # Display the first few rows
         print(df1.head())
```

```
   Att_hrs_fall GENDER  HIGH_SCHOOL_PERCENTILE        AGE  Att_hrs_spr  \
0          13.0      F                    97.0  18.910335         15.0   
1          14.0      M                    97.0  18.036961         15.0   
2          15.0      M                    83.0  18.162902         15.0   
3          14.0      M                    92.0  18.590007         12.0   
4          16.0      F                    80.0  18.880219         16.0   

   Avg_income   Distance Dropped_course  Major_rate     SAT  ...  Legacynum  \
0     30573.0   5.083094              0    0.821782  1160.0  ...          0   
1     27305.0   2.198764              0    0.795455  1050.0  ...          0   
2     30573.0   5.083094              0    0.843750  1140.0  ...          1   
3     35865.0   3.245574              0         NaN  1090.0  ...          0   
4     40125.0  19.486585              0    0.840391  1150.0  ...          0   

  Stu_worker_ind Need_pct_met  Perc_hrs_comp_fall   Hs_rate  Dorm_rate  \
0              0     1.000000                 1.0  0.812500        0.8   
1              0     0.977366                 1.0  1.000000        0.8   
2              0     0.913965                 1.0  0.812500        0.8   
3              1     1.000000                 1.0  0.847826        0.8   
4              1     1.000000                 1.0  0.875000        0.8   

   Instate Transcrip  Fall_GPA  Target  
0        1         0  3.307692       0  
1        1         1  2.821429       0  
2        1         0  3.133333       0  
3        1         0  2.678571       0  
4        1         0  2.500000       0  

[5 rows x 21 columns]
```

In [15]:

```
# Display metadata
print("Column Names:", meta.column_names)
print("Column Labels:", meta.column_labels)
```

```
Column Names: ['Att_hrs_fall', 'GENDER', 'HIGH_SCHOOL_PERCENTILE', 'AGE', 'Att_hrs_spr', 'Avg_income', 'Distance', 'Dropped_cou
rse', 'Major_rate', 'SAT', 'Extra_curr', 'Legacynum', 'Stu_worker_ind', 'Need_pct_met', 'Perc_hrs_comp_fall', 'Hs_rate', 'Dorm_
rate', 'Instate', 'Transcrip', 'Fall_GPA', 'Target']
Column Labels: ['Att_hrs_fall', 'GENDER', 'HIGH_SCHOOL_PERCENTILE', 'AGE', 'Att_hrs_spr', 'Avg_income', 'Distance', 'Dropped_co
urse', 'Major_rate', 'SAT', 'Extra_curr', 'Legacynum', 'Stu_worker_ind', 'Need_pct_met', 'Perc_hrs_comp_fall', 'Hs_rate', 'Dorm
_rate', 'Instate', 'Transcrip', 'Fall_GPA', 'Target']
```

In [17]:

```
#Q1a - How many observations (rows) exist in the Retention table?

print(f"Number of observations (rows): {df1.shape[0]}")
```

```
Number of observations (rows): 2626
```

In [19]:

```
#Q1b - •        How many variables (columns) exist in the Retention table?

print(f"Number of variables (columns): {df1.shape[1]}")
```

```
Number of variables (columns): 21
```

Q1c - How are the initial roles and levels of the variables determined? Can we change the default settings? Explain by example.

Explanation of roles and levels

Variables roles are categorized as input or target based on their function in the analysis Input variables are the independent and used to determine the target while target variables are dependent and is the result

Varibale levels are nominal,ordinal and interval/ratio

We can override their default setting if required

In [22]:

```
# Example: Convert AGE into categorical age groups
import pandas as pd
# check if its numerical or categorical
print(df1['AGE'].dtype)
```

```
float64
```

In [24]:

```
# change it from numerical to categorical
df1['AGE'] = df1['AGE'].astype('category')
print(df1['AGE'].dtype)
```

```
category
```

Q2

In [27]:
```
#check for missing values
print(df1.isnull().sum())
```

```
Att_hrs_fall               0
GENDER                     0
HIGH_SCHOOL_PERCENTILE   263
AGE                        0
Att_hrs_spr                0
Avg_income               157
Distance                  98
Dropped_course             0
Major_rate                79
SAT                        0
Extra_curr                 0
Legacynum                  0
Stu_worker_ind             0
Need_pct_met               0
Perc_hrs_comp_fall         0
Hs_rate                  805
Dorm_rate                  0
Instate                    0
Transcrip                  0
Fall_GPA                   0
Target                     0
dtype: int64
```

No, variables with missing values are not rejected by Python Avg_income has 157 missing variables, HIGH_SCHOOL_PERCENTILE has 263 missing variables and Hs_rate has 805 missing variables.

In [30]:
```
# check which numeric variables are assigned as nominal variables
variables =df1.select_dtypes(include=['object'])
print(variables.columns)
```

```
Index(['GENDER', 'Dropped_course', 'Legacynum', 'Stu_worker_ind', 'Instate',
       'Transcrip', 'Target'],
      dtype='object')
```

Usually variables with the data type 'object' are typically treated as nominal variables, sometimes numeric-looking values can also be stored as objects sucha as (0,1) this can be the case if their data was stored as string format originally.

Is there a maximum number of levels for character variables

There is no maximum number of levels for character variables a column can have multiple unique variables, however using too many unique variables can lead to technical challenges and can make the variable less useful during analysis.

Q3

In [43]:

```
#How many input variables in the DataFrame? How many target variables? How many of the input variables are nominal? Are there

#Total input variables: 20
#Target variables: 1 (Target)
#Nominal (categorical) input variables: 2
#'Att_hrs_fall'
#'GENDER'
#Rejected variables: None (no columns were discarded or found irrelevant based on this analysis)
```

In [ ]:

```
#What is the sample size?
# Total records: 2626. 10% sample (commonly used in exploratory data analysis): 262 rows
# This is a typical and manageable size for quick testing, modeling, or visualization.
```

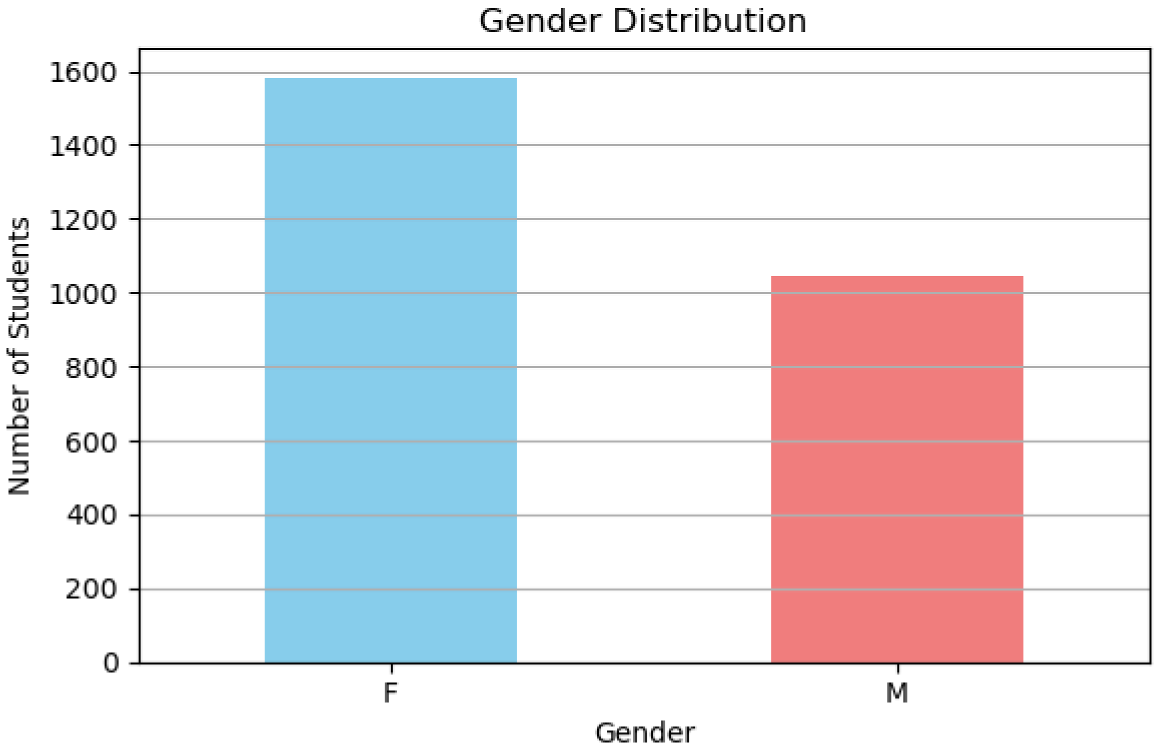
In [49]:

```
import matplotlib.pyplot as plt
```

In [51]:

```
# Step 2: Plot bar chart of GENDER
gender_counts = df1['GENDER'].value_counts()
gender_percentages = gender_counts / gender_counts.sum() * 100

# Bar plot
```

```
plt.figure(figsize=(6, 4))
gender_counts.plot(kind='bar', color=['skyblue', 'lightcoral'])
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Number of Students')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Step 3: Calculate percentage of female students
female_percentage = gender_percentages.get('F', 0)
print(f"Percentage of female students: {female_percentage:.2f}%")
```

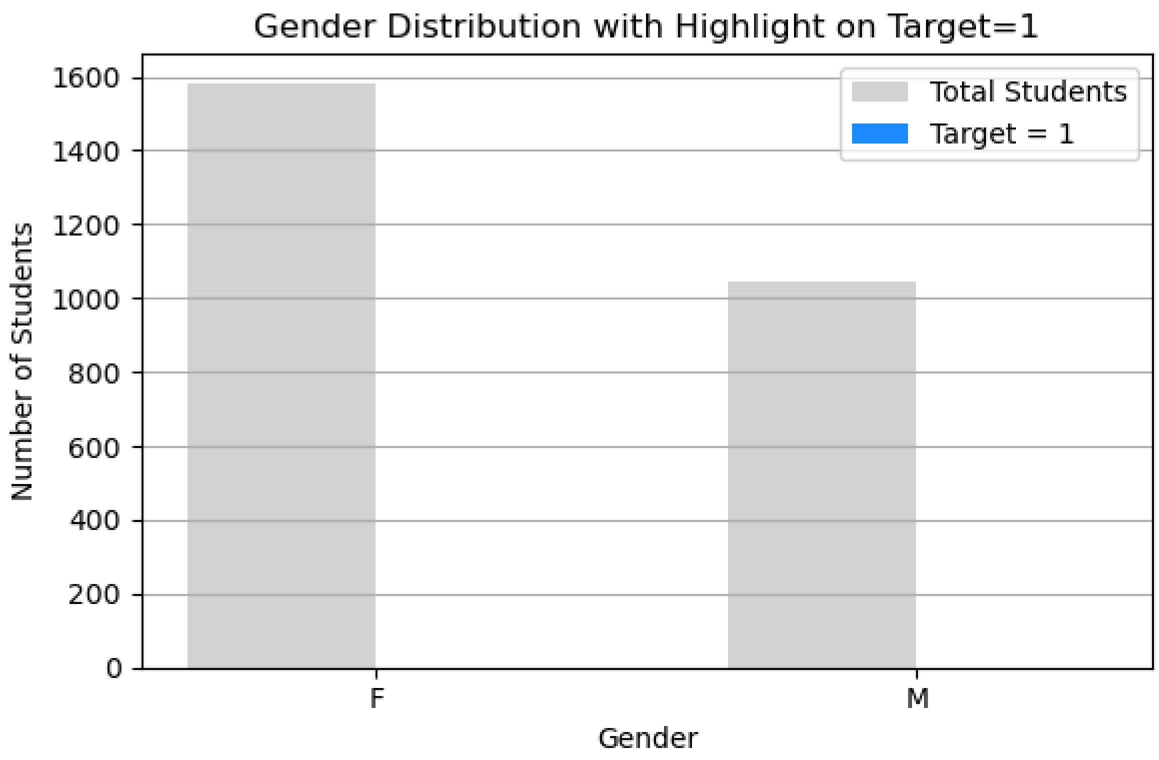


```
Percentage of female students: 60.24%
```

In [59]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

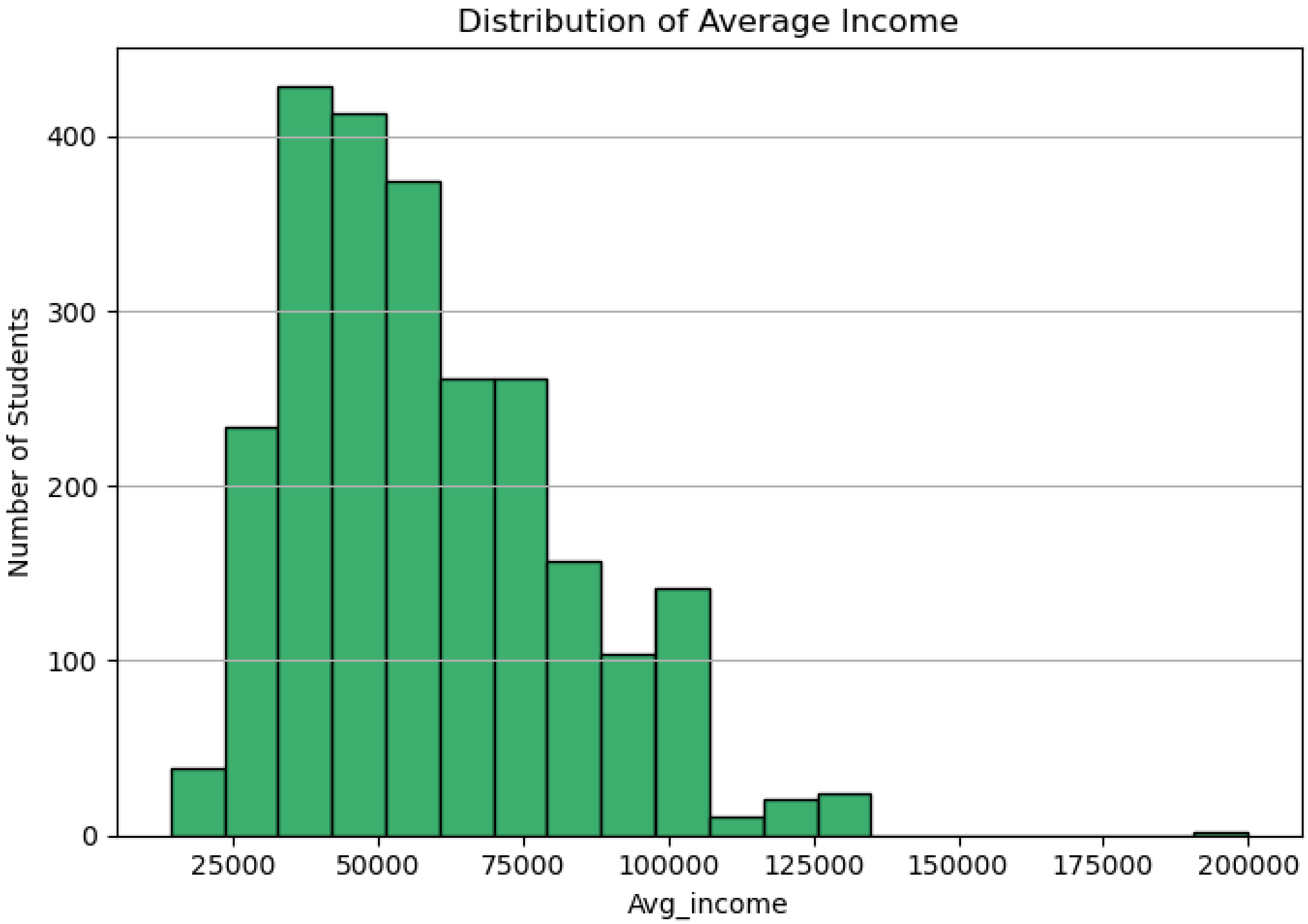
```python
import numpy as np

# Step 1: Load and parse the CSV
file_path = "retention.csv"
with open(file_path, 'r') as f:
    lines = f.readlines()

header = lines[0].strip().split(",")
data = [line.strip().split(",") for line in lines[1:]]
df = pd.DataFrame(data, columns=[col.strip('"') for col in header])

# Step 2: Count total and target=1 by gender
total_counts = df['GENDER'].value_counts()
target1_counts = df[df['Target'] == '1']['GENDER'].value_counts()

# Step 3: Align bars side-by-side
labels = sorted(total_counts.index)
x = np.arange(len(labels))  # label locations
width = 0.35  # width of the bars

# Step 4: Plot
fig, ax = plt.subplots(figsize=(6, 4))
bars1 = ax.bar(x - width/2, [total_counts.get(label, 0) for label in labels], width, label='Total Students', color='lightgray'
bars2 = ax.bar(x + width/2, [target1_counts.get(label, 0) for label in labels], width, label='Target = 1', color='dodgerblue')

# Labels and formatting
ax.set_xlabel('Gender')
ax.set_ylabel('Number of Students')
ax.set_title('Gender Distribution with Highlight on Target=1')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()
ax.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [61]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Load and parse CSV again if not already done
file_path = "retention.csv"
with open(file_path, 'r') as f:
    lines = f.readlines()

header = lines[0].strip().split(",")
data = [line.strip().split(",") for line in lines[1:]]
df = pd.DataFrame(data, columns=[col.strip('"') for col in header])

# Step 2: Convert Avg_income to numeric (handle non-numeric entries if any)
df['Avg_income'] = pd.to_numeric(df['Avg_income'], errors='coerce')

# Step 3: Plot histogram
```

```
plt.figure(figsize=(7, 5))
plt.hist(df['Avg_income'].dropna(), bins=20, color='mediumseagreen', edgecolor='black')
plt.title('Distribution of Average Income')
plt.xlabel('Avg_income')
plt.ylabel('Number of Students')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [65]:

```
import pandas as pd
```

```python
# Step 1: Load and parse CSV
file_path = "retention.csv"
with open(file_path, 'r') as f:
    lines = f.readlines()

header = lines[0].strip().split(",")
data = [line.strip().split(",") for line in lines[1:]]
df = pd.DataFrame(data, columns=[col.strip('"') for col in header])

# Step 2: Get the last 10 rows
last_10_rows = df.tail(10)

# Step 3: Display the result
print(last_10_rows)
```

```
     Att_hrs_fall GENDER HIGH_SCHOOL_PERCENTILE                 AGE  \
2616        "15.0      M                   59.0   18.60643394934976   
2617        "12.0      M                   13.0   18.88021902806297   
2618        "13.0      F                   51.0   18.48870636550308   
2619        "14.0      M                   62.0   18.28062970568104   
2620        "14.0      M                    0.6  18.967830253251197   
2621        "13.0      M                   42.0   18.15195071868583   
2622        "13.0      M                   74.0   18.650239561943874   
2623        "16.0      F                    0.6  18.102669404517453   
2624        "12.0      F                   71.0   18.31895961670089   
2625        "16.0      F                   95.0  18.666666666666668   

     Att_hrs_spr Avg_income           Distance Dropped_course  \
2616        16.0    36760.0  156.74223825933137              0   
2617        18.0    26725.0  167.68172698652128              0   
2618        13.0    53190.0  1186.8518671692395              0   
2619        16.0    60898.0   1154.751207131066              0   
2620        13.0                                             0   
2621        12.0    48581.0  179.44467807371302              0   
2622        18.0             110.99359173168243              0   
2623        12.0    73510.0   356.6742856734745              0   
2624        13.0                                             0   
2625        16.0    59599.0   91.02294960168159              0   

             Major_rate     SAT  ... Legacynum Stu_worker_ind  \
2616  0.8403908794788274  1290.0  ...         0              1   
2617  0.7954545454545454  1060.0  ...         0              0   
2618  0.8315847598012148   980.0  ...         0              0   
2619  0.8315109343936382  1070.0  ...         1              0   
2620  0.8292682926829268  1140.0  ...         0              0   
2621  0.8383838383838383  1080.0  ...         0              0   
2622  0.8315109343936382  1110.0  ...         0              1   
2623  0.8285714285714286  1150.0  ...         0              0   
2624  0.8292682926829268  1060.0  ...         0              0   
2625  0.8295148247978437  1140.0  ...         0              0   

           Need_pct_met   Perc_hrs_comp_fall   Hs_rate Dorm_rate Instate  \
2616  0.8166249293157767  0.06666666666666667   0.77381  0.817533       1   
2617  0.9697547419782209                  0.5       0.6  0.845717       0   
2618                 1.0                  0.7       0.6   0.82125       0   
2619                 0.0  0.21428571428571427     0.625  0.845717       0   
```

```
2620                 1.0   0.2857142857142857       0.7       0.7       1
2621                 1.0                  0.5       0.7       0.7       1
2622  0.9018721037998146   0.7692307692307693  0.666667       0.7       0
2623                 1.0               0.6875       0.6       0.7       0
2624                 1.0                  0.5       0.5       0.7       0
2625                 1.0                 0.25       0.5       0.6       1

     Transcrip     Fall_GPA Target
2616         1  0.071428571     1"
2617         1          0.7     1"
2618         1          0.7     1"
2619         1  0.692307692     1"
2620         0  0.285714286     1"
2621         1  1.346153846     1"
2622         1          1.0     1"
2623         1          0.7     1"
2624         1          1.0     1"
2625         0      0.78125     1"

[10 rows x 21 columns]
```

```
In [ ]:
```