In [1]:
```
import pandas as pd
import numpy as np
import matplotlib
```

In [2]:
```
#how to import data into python

df=pd.read_csv(r"C:\Users\USER\OneDrive\Documentos\student_habits_performance.csv")
```

In [4]:
```
df
```

Out[4]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	
...	...	...	...	...	...	...	
995	S1995	21	Female	2.6	0.5	1.6	
996	S1996	17	Female	2.9	1.0	2.4	
997	S1997	20	Male	3.0	2.6	1.3	
998	S1998	24	Male	5.4	4.1	1.1	
999	S1999	19	Female	4.3	2.9	1.9	

1000 rows × 16 columns

In [ ]:

In [ ]:

In [ ]:

In [31]:
```
# first step to take after importing your data is to inspect your data

###Data inspection using head,tail,info,describe,columns,index,datatype,shape

#to check the first 5 rows of the data

df.head()
```

Out[31]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_ti
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	

In [32]:
```
#to check the last 5 rows of the data

df.tail()
```

Out[32]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part
995	S1995	21	Female	2.6	0.5	1.6	
996	S1996	17	Female	2.9	1.0	2.4	
997	S1997	20	Male	3.0	2.6	1.3	
998	S1998	24	Male	5.4	4.1	1.1	
999	S1999	19	Female	4.3	2.9	1.9	

In [33]:
```
#to check the columns of the data

df.columns
```

Out[33]:
```
Index(['student_id', 'age', 'gender', 'study_hours_per_day',
       'social_media_hours', 'netflix_hours', 'part_time_job',
       'attendance_percentage', 'sleep_hours', 'diet_quality',
       'exercise_frequency', 'parental_education_level', 'internet_quality',
       'mental_health_rating', 'extracurricular_participation', 'exam_score'],
      dtype='object')
```

In [34]:
```
#to check the shape of the data

df.shape
```

Out[34]:
```
(1000, 16)
```

In [35]:
```
#to check the datas information

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   student_id                     1000 non-null   object 
 1   age                            1000 non-null   int64  
 2   gender                         1000 non-null   object 
 3   study_hours_per_day            1000 non-null   float64
 4   social_media_hours             1000 non-null   float64
 5   netflix_hours                  1000 non-null   float64
 6   part_time_job                  1000 non-null   object 
 7   attendance_percentage          1000 non-null   float64
 8   sleep_hours                    1000 non-null   float64
 9   diet_quality                   1000 non-null   object 
 10  exercise_frequency             1000 non-null   int64  
 11  parental_education_level       909 non-null    object 
 12  internet_quality               1000 non-null   object 
 13  mental_health_rating           1000 non-null   int64  
 14  extracurricular_participation  1000 non-null   object 
 15  exam_score                     1000 non-null   float64
dtypes: float64(6), int64(3), object(7)
memory usage: 125.1+ KB
```

In [36]:
```
#to get describe information of the data

df.describe()
```

Out[36]:

	age	study_hours_per_day	social_media_hours	netflix_hours	attendance_perce
count	1000.0000	1000.00000	1000.000000	1000.000000	1000.0
mean	20.4980	3.55010	2.505500	1.819700	84.1
std	2.3081	1.46889	1.172422	1.075118	9.3
min	17.0000	0.00000	0.000000	0.000000	56.0
25%	18.7500	2.60000	1.700000	1.000000	78.0
50%	20.0000	3.50000	2.500000	1.800000	84.4
75%	23.0000	4.50000	3.300000	2.525000	91.0
max	24.0000	8.30000	7.200000	5.400000	100.0

In [ ]:

In [ ]:

In [ ]:

In [37]:
```
# Second Step To Take is to clean your data

##check for null,duplicates,change of columns,replace values,drop duplicates,fillna
```

```
#to check for duplicates

df.duplicated()
```

Out[37]:

```
0      False
1      False
2      False
3      False
4      False
       ...  
995    False
996    False
997    False
998    False
999    False
Length: 1000, dtype: bool
```

In [38]:

```
#to check for duplicates

df[df.duplicated()]
```

Out[38]:

student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_tim

In [39]:

```
#to drop duplicates from the data sets

df.drop_duplicates()
```

Out[39]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	
...	...	...	...	...	...	...	
995	S1995	21	Female	2.6	0.5	1.6	
996	S1996	17	Female	2.9	1.0	2.4	
997	S1997	20	Male	3.0	2.6	1.3	
998	S1998	24	Male	5.4	4.1	1.1	
999	S1999	19	Female	4.3	2.9	1.9	

1000 rows × 16 columns

In [40]:
```
#to drop duplicates in a column
df["gender"].drop_duplicates()
```

Out[40]:
```
0     Female
2       Male
18     Other
Name: gender, dtype: object
```

In [41]:
```
# to check for null values

df.isnull()
```

Out[41]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	par
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
995	False	False	False	False	False	False	
996	False	False	False	False	False	False	
997	False	False	False	False	False	False	
998	False	False	False	False	False	False	
999	False	False	False	False	False	False	

1000 rows × 16 columns

In [42]:
```
#to drop null values

df.dropna()
```

Out[42]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	
...	...	...	...	...	...	...	
995	S1995	21	Female	2.6	0.5	1.6	
996	S1996	17	Female	2.9	1.0	2.4	
997	S1997	20	Male	3.0	2.6	1.3	
998	S1998	24	Male	5.4	4.1	1.1	
999	S1999	19	Female	4.3	2.9	1.9	

909 rows × 16 columns

In [43]:

```
# to replace null in the data with zero 0

df.fillna(0)
```

Out[43]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	
...	...	...	...	...	...	...	
995	S1995	21	Female	2.6	0.5	1.6	
996	S1996	17	Female	2.9	1.0	2.4	
997	S1997	20	Male	3.0	2.6	1.3	
998	S1998	24	Male	5.4	4.1	1.1	
999	S1999	19	Female	4.3	2.9	1.9	

1000 rows × 16 columns

In [44]:
```
# to replace null value in a specific column with zero 0

df["age"].fillna(0)
```

Out[44]:
```
0      23
1      20
2      21
3      23
4      19
       ..
995    21
996    17
997    20
998    24
999    19
Name: age, Length: 1000, dtype: int64
```

In [45]:
```
#to create a new column and replace a value into the new column from the old column
#you can then go ahead to drop the replaced column

df["Gender"]=df["gender"].replace("Female","F")
```

In [46]:
```
df
```

Out[46]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	
...	...	...	...	...	...	...	
995	S1995	21	Female	2.6	0.5	1.6	
996	S1996	17	Female	2.9	1.0	2.4	
997	S1997	20	Male	3.0	2.6	1.3	
998	S1998	24	Male	5.4	4.1	1.1	
999	S1999	19	Female	4.3	2.9	1.9	

1000 rows × 17 columns

In [ ]:

In [47]:
```
#to drop a column from the data set
#after dropping use INPLACE to parmanent the change
```

```python
df.drop(["New_Column"],axis= 1)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[47], line 4
      1 #to drop a column from the data set
      2 #after dropping use INPLACE to parmanent the change
----> 4 df.drop(["New_Column"],axis= 1)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:5581, in DataFrame.drop(sel
f, labels, axis, index, columns, level, inplace, errors)
   5433 def drop(
   5434     self,
   5435     labels: IndexLabel | None = None,
   (...)
   5442     errors: IgnoreRaise = "raise",
   5443 ) -> DataFrame | None:
   5444     """
   5445     Drop specified labels from rows or columns.
   5446 
   (...)
   5579             weight  1.0     0.8
   5580     """
-> 5581     return super().drop(
   5582         labels=labels,
   5583         axis=axis,
   5584         index=index,
   5585         columns=columns,
   5586         level=level,
   5587         inplace=inplace,
   5588         errors=errors,
   5589     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4788, in NDFrame.drop(sel
f, labels, axis, index, columns, level, inplace, errors)
   4786 for axis, labels in axes.items():
   4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4790 if inplace:
   4791     self._update_inplace(obj)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4830, in NDFrame._drop_axi
s(self, labels, axis, level, errors, only_slice)
   4828         new_axis = axis.drop(labels, level=level, errors=errors)
   4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
   4831     indexer = axis.get_indexer(new_axis)
   4833 # Case for non-unique axis
   4834 else:

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:7070, in Index.drop(s
elf, labels, errors)
   7068 if mask.any():
   7069     if errors != "ignore":
-> 7070         raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7071     indexer = indexer[~mask]
   7072 return self.delete(indexer)
```

```
KeyError: "['New_Column'] not found in axis"
```

In [ ]:
```
#to drop a table permanently you introduce INPLACE=True

df.drop(["New_Column"],axis=1,inplace=True)
```

In [ ]:
```
df
```

In [ ]:
```
#to sort values
df.sort_values("age",ascending=True)
```

In [ ]:
```
#to change data type of a particular column if need be

df["Newtype"]=df["study_hours_per_day"].astype(int)
```

In [ ]:
```
df
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```
#Assignment 1
#drop the "newcolumn" used to change data type

df.drop(["Newtype"],axis=1)
```

In [ ]:
```
df
```

In [ ]:
```
df.drop(["Newtype"],axis=1,inplace=True)
```

In [ ]:
```
df
```

In [ ]:

In [ ]:

In [ ]:

In [48]:
```
# Third Step To Take is Analysing your Data Based On Giving Measures

## data analysis, statistical analysis

df.head(3)
```

Out[48]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_ti
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	

In [49]:
```
# How many students have partime jobs?

df["part_time_job"].value_counts()
```

Out[49]:
```
part_time_job
No     785
Yes    215
Name: count, dtype: int64
```

In [50]:
```
# How many female and males are in the data

df["gender"].value_counts()
```

Out[50]:
```
gender
Female    481
Male      477
Other      42
Name: count, dtype: int64
```

In [51]:
```
# what is the maximum study hours per day

df["study_hours_per_day"].max()
```

Out[51]:
```
8.3
```

In [52]:
```
# what is the minimum study hour per day

df["study_hours_per_day"].min()
```

Out[52]:
```
0.0
```

In [53]:
```
# what is the mean age

df["age"].mean()
```

Out[53]:
```
20.498
```

In [ ]:

In [ ]:

In [ ]:

In [54]:
```
#ASSIGNMENT 2
#SHOW VALUE COUNT OF DIET QUALITY FOR ALL STUDENTS
#AVERAGE ATTENDANCE PERCENTAGE FOR STUDENTS
#CORRELATION ANALYSIS OF NUMERICS

df.head()
```

Out[54]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_ti
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	

In [55]:
```
#Value Count Of Diet For All Students

df[["diet_quality"]].value_counts()
```

Out[55]:
```
diet_quality
Fair            437
Good            378
Poor            185
Name: count, dtype: int64
```

In [56]:
```
# Average attendance percentage of all students

df[["attendance_percentage"]].mean()
```

Out[56]:
```
attendance_percentage    84.1317
dtype: float64
```

In [57]:
```
# Average attendance percentage of all gender

df.groupby(["gender"])[["attendance_percentage"]].mean()
```

Out[57]:

	attendance_percentage
gender	
Female	84.371518
Male	83.894549
Other	84.078571

In [58]:
```
# Average attendance percentage of all students

df.groupby(["gender","age"])[["attendance_percentage"]].mean()
```

Out[58]:

		attendance_percentage
gender	**age**	
Female	**17**	85.758621
	18	84.340678
	19	84.922951
	20	83.929730
	21	84.360317
	22	83.068333
	23	84.476471
	24	84.261818
Male	**17**	83.256522
	18	86.331481
	19	82.710204
	20	83.435385
	21	83.023636
	22	83.488235
	23	85.308065
	24	83.634722
Other	**17**	88.666667
	18	89.575000
	19	82.766667
	20	80.871429
	21	78.228571
	22	90.600000
	23	87.300000
	24	82.000000

In [59]:

```
#CORRELATION ANALYSIS OF NUMERICS
## To reveal all columns that contain numerical data

df.select_dtypes(include=['int64', 'float64']).corr()
```

Out[59]:

	age	study_hours_per_day	social_media_hours	netflix_hours
age	1.000000	0.003971	-0.009151	-0.001174
study_hours_per_day	0.003971	1.000000	0.020282	-0.031158
social_media_hours	-0.009151	0.020282	1.000000	0.011477
netflix_hours	-0.001174	-0.031158	0.011477	1.000000
attendance_percentage	-0.026055	0.026264	0.040479	-0.002092
sleep_hours	0.037482	-0.027757	0.018236	-0.000935
exercise_frequency	-0.003836	-0.028701	-0.037319	-0.006448
mental_health_rating	-0.045101	-0.003768	0.001496	0.008034
exam_score	-0.008907	0.825419	-0.166733	-0.171779

In [ ]:

In [60]:
```
df.head(2)
```

Out[60]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_ti
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	

In [61]:
```
# Sort
## total nextflix hours of all students sort by hours

df.groupby("gender")[["netflix_hours"]].sum().sort_values("netflix_hours",ascending
```

Out[61]:

	netflix_hours
gender	
Male	868.5
Female	864.8
Other	86.4

In [62]:
```
# check exam score for all gender and partime job

df.groupby(["gender","part_time_job"])[["exam_score"]].sum()
```

Out[62]:

gender	part_time_job	exam_score
Female	No	26806.6
	Yes	6739.0
Male	No	25692.7
	Yes	7396.0
Other	No	2322.4
	Yes	644.8

In [63]:

```
# check exam score for all gender

df.groupby(["gender","age"])[["exam_score"]].sum()
```

Out[63]:

		exam_score
gender	**age**	
Female	**17**	4140.5
	18	4279.9
	19	4203.5
	20	5273.1
	21	4329.2
	22	4142.7
	23	3505.1
	24	3671.6
Male	**17**	4829.9
	18	3706.4
	19	3325.3
	20	4471.2
	21	3639.9
	22	3530.2
	23	4444.4
	24	5141.4
Other	**17**	385.2
	18	313.6
	19	188.4
	20	464.0
	21	473.9
	22	135.7
	23	487.8
	24	518.6

In [64]:

```
df.head(2)
```

Out[64]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_ti
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	

In [65]:
```
df.pivot_table(index=["gender"],columns=["part_time_job"],values=["exam_score"],agg
```

Out[65]:

	sum	
	exam_score	
part_time_job	**No**	**Yes**
gender		
Female	26806.6	6739.0
Male	25692.7	7396.0
Other	2322.4	644.8

In [ ]:

In [ ]:

In [ ]:

In [66]:
```
#ASSIGNMENT 3
#CALCULATE THE AVERAGE SLEEP HOURS FOR THOSE ON DIET QUALITY
#CALCULATE THE TOTAL STUDENT UNDER DIET QUALITY
#WHICH EDUCATION LEVEL HAD THE HIGHEST AGE OF ALL STUDENT

df.head(2)
```

Out[66]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_ti
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	

In [67]:
```
# Average sleep hours for those on diet
df.groupby("diet_quality")[["sleep_hours"]].mean().sort_values("sleep_hours",ascend
```

Out[67]:

	sleep_hours
diet_quality	
Poor	6.559459
Fair	6.465217
Good	6.432011

In [68]:
```
# Total student under diet quality

df[["gender","diet_quality"]].value_counts()
```

Out[68]:
```
gender  diet_quality
Female  Fair            210
Male    Fair            209
        Good            185
Female  Good            179
        Poor             92
Male    Poor             83
Other   Fair             18
        Good             14
        Poor             10
Name: count, dtype: int64
```

In [33]:
```
# which education level had the highest age for all students

df.groupby("parental_education_level")[["age"]].max()
```

Out[33]:

	age
parental_education_level	
Bachelor	24
High School	24
Master	24

In [35]:
```
# which gender had the maximum age in parental education level

df.groupby(["gender","parental_education_level"])[["age"]].max()
```

Out[35]:

gender	parental_education_level	age
Female	Bachelor	24
	High School	24
	Master	24
Male	Bachelor	24
	High School	24
	Master	24
Other	Bachelor	23
	High School	24
	Master	24

In [40]:
```
df.groupby(["gender","parental_education_level"])[["study_hours_per_day"]].max()
```

Out[40]:

gender	parental_education_level	study_hours_per_day
Female	Bachelor	7.5
	High School	7.6
	Master	6.7
Male	Bachelor	6.8
	High School	8.3
	Master	8.2
Other	Bachelor	4.7
	High School	5.1
	Master	5.6

In [41]:
```
# describe age for the parental educational level

df.groupby("parental_education_level")["age"].describe()
```

Out[41]:

parental_education_level	count	mean	std	min	25%	50%	75%	max
Bachelor	350.0	20.600000	2.323420	17.0	19.0	20.0	23.0	24.0
High School	392.0	20.364796	2.256468	17.0	18.0	20.0	22.0	24.0
Master	167.0	20.473054	2.366468	17.0	18.0	20.0	23.0	24.0

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```python
df.head(4)
```

In [16]:
```python
# To apply upper case to a specific columnn

df["applylamnda"]=df["gender"].apply(lambda x:x.upper())
```

In [23]:
```python
df
```

Out[23]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	
...	...	...	...	...	...	...	
995	S1995	21	Female	2.6	0.5	1.6	
996	S1996	17	Female	2.9	1.0	2.4	
997	S1997	20	Male	3.0	2.6	1.3	
998	S1998	24	Male	5.4	4.1	1.1	
999	S1999	19	Female	4.3	2.9	1.9	

1000 rows × 17 columns

In [14]:
```python
# Apply a transformation to the mental_health_rating column by multiplying each val
# The apply() method executes the lambda function on each element of the column
# The result is stored in a new column called "Numericapply"
```

```
df["Numericapply"] = df["mental_health_rating"].apply(lambda x: x*2)
```

In [15]:
```
df.head()
```

Out[15]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_ti
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	

In [9]:
```
# Group the dataframe by 'parental_education_level' column
# Then count the occurrences of each gender within each education level group

df.groupby("parental_education_level")["gender"].count()
```

Out[9]:
```
parental_education_level
Bachelor       350
High School    392
Master         167
Name: gender, dtype: int64
```

In [73]:
```
df.head()
```

Out[73]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part_ti
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	

In [8]:
```
# Calculate the mean of the 'age' column and count the number of entries in the 'ge
# This aggregation returns a new DataFrame with one row containing these summary st

df.agg({"age":"mean","gender":"count"})
```

Out[8]:
```
age         20.498
gender    1000.000
dtype: float64
```

In [7]:
```
# Add 10 to each value in the 'age' column using transform
# transform applies the function to each element and returns a Series with the same
```

```
df["age"].transform(lambda x:x +10)
```

Out[7]:
```
0      33
1      30
2      31
3      33
4      29
       ..
995    31
996    27
997    30
998    34
999    29
Name: age, Length: 1000, dtype: int64
```

In [6]:
```
#Correlation coefficient between 'age',and 'exercise_frequency' columns

df["age"].corr(df["exercise_frequency"])
```

Out[6]:
```
-0.0038362358530908297
```

In [32]:
```
# Convert all values in the 'diet_quality' column to lowercase for consistency in d

df["diet_quality"]=df["diet_quality"].apply(lambda x:x.lower())
```

In [44]:
```
df
```

Out[44]:

	student_id	age	gender	study_hours_per_day	social_media_hours	netflix_hours	part
0	S1000	23	Female	0.0	1.2	1.1	
1	S1001	20	Female	6.9	2.8	2.3	
2	S1002	21	Male	1.4	3.1	1.3	
3	S1003	23	Female	1.0	3.9	1.0	
4	S1004	19	Female	5.0	4.4	0.5	
...	...	...	...	...	...	...	
995	S1995	21	Female	2.6	0.5	1.6	
996	S1996	17	Female	2.9	1.0	2.4	
997	S1997	20	Male	3.0	2.6	1.3	
998	S1998	24	Male	5.4	4.1	1.1	
999	S1999	19	Female	4.3	2.9	1.9	

1000 rows × 18 columns

In [ ]: