Lee decides to walk 10000 steps every day to combat the effect that lockdown has had on his body's agility, mobility, flexibility and strength. Consider the following data from fitness tracker over a period of 10 days

Day number	Steps walked
1	6012
2	7079
3	6886
4	7230
5	4598
6	5564
7	6971
8	7763
9	8032
10	9569

In [28]:
```
#Represent the above data in a 10x2 array. In each row, the first element should contain day number and second
import numpy as np
#Creating a 2D array
Day_number = np.arange(1,11)
Steps_walked = [6012,7079,6886,7230,4598,5564,6971,7763,8032,9569]
arr = np.array([Day_number, Steps_walked])
arr = arr.T
arr
```

Out[28]:
```
array([[   1, 6012],
       [   2, 7079],
       [   3, 6886],
       [   4, 7230],
       [   5, 4598],
       [   6, 5564],
       [   7, 6971],
       [   8, 7763],
       [   9, 8032],
       [  10, 9569]])
```

In [3]:
```
#Lee notices that the tracker's battery dies every day at 7 pm. Lee discovers that on an average, he walks 2000
new_arr= arr[:,1] + 2000
arr[:,1]=new_arr
arr
```

Out[3]:
```
array([[    1,  8012],
       [    2,  9079],
       [    3,  8886],
       [    4,  9230],
       [    5,  6598],
       [    6,  7564],
       [    7,  8971],
       [    8,  9763],
       [    9, 10032],
       [   10, 11569]])
```

In [5]:
```
#Write a program that returns the steps walked if the steps walked are more than 9000.
matched = arr[:,1]>9000
matched
new_arr = arr[matched]
new_arr
```

Out[5]:
```
array([[    2,  9079],
       [    4,  9230],
       [    8,  9763],
       [    9, 10032],
       [   10, 11569]])
```

In [7]:
```
#Print an array containing steps walked in sorted order.
sortedArr = arr[arr[:,1].argsort()]
print('Sorted 2D Numpy Array')
print(sortedArr)
```

```
Sorted 2D Numpy Array
[[    5  6598]
 [    6  7564]
 [    1  8012]
 [    3  8886]
 [    7  8971]
 [    2  9079]
 [    4  9230]
 [    8  9763]
 [    9 10032]
 [   10 11569]]
```

Create a dataframe as follows using Pandas

	Chemistry	Physics	Mathematics	English
Subodh	67	45	50	19
Ram	90	92	87	90
Abdul	66	72	81	72
John	32	40	12	68

In [10]:
```
#Create a dataframe as follows using Pandas
import pandas as pd
import numpy as np
marks = {'Chemistry': [67,90,66,32],
 'Physics': [45,92,72,40],
 'Mathematics': [50,87,81,12],
 'English': [19,90,72,68]}
marks_df = pd.DataFrame(marks, index = ['Subodh', 'Ram', 'Abdul', 'John'])
marks_df
```

Out[10]:

	Chemistry	Physics	Mathematics	English
Subodh	67	45	50	19
Ram	90	92	87	90
Abdul	66	72	81	72
John	32	40	12	68

In [12]:
```
#The teacher wants to create a new column called total and the value of each row in total column should be the s
marks_df['Total'] = marks_df['Chemistry'] + marks_df['Physics'] + marks_df['Mathematics'] + marks_df['English']
marks_df
```

Out[12]:

	Chemistry	Physics	Mathematics	English	Total
Subodh	67	45	50	19	181
Ram	90	92	87	90	359
Abdul	66	72	81	72	291
John	32	40	12	68	152

In [14]:
```
#Drop the Total column
marks_df.drop(columns = 'Total', inplace = True)
marks_df
```

Out[14]:

	Chemistry	Physics	Mathematics	English
Subodh	67	45	50	19
Ram	90	92	87	90
Abdul	66	72	81	72
John	32	40	12	68

In [16]:
```
#The teacher wants to award five bonus marks to all the students.
new_marks = marks_df + 5
new_marks
```

Out[16]:

	Chemistry	Physics	Mathematics	English
Subodh	72	50	55	24
Ram	95	97	92	95
Abdul	71	77	86	77
John	37	45	17	73

In [18]:
```
#The teacher wants to increase the marks of all the students as follows?Chemistry: + 5 Physics: + 10 Mathematic
new_marks = marks_df + [5,10,10,2]
new_marks
```

Out[18]:

	Chemistry	Physics	Mathematics	English
Subodh	72	55	60	21
Ram	95	102	97	92
Abdul	71	82	91	74
John	37	50	22	70

In [20]:
```
#The teacher wants to get the total marks scored in each subject
marks_df.apply(np.sum, axis = 0)
```

Out[20]:
```
Chemistry      255
Physics        249
Mathematics    230
English        249
dtype: int64
```

In [22]:
```
#The teacher wants to get the total marks scored by each student.
marks_df.apply(np.sum, axis = 1)
```

Out[22]:
```
Subodh    181
Ram       359
Abdul     291
John      152
dtype: int64
```

In [26]:
```
#The teacher wants to hide the marks of the students who scored less than 35 marks and display Fail in place of
f = marks_df < 35
marks_df.mask(f, 'Fail')
```

Out[26]:

	Chemistry	Physics	Mathematics	English
Subodh	67	45	50	Fail
Ram	90	92	87	90
Abdul	66	72	81	72
John	Fail	40	Fail	68

Perform the following operation on Autompg.csv of XYZ Custom Cars company using Pandas

In [30]:
```
#Importing libraries
import numpy as np
import pandas as pd
```

In [131…
```
#Read data from an existing file
import pandas as pd
import numpy as np
df = pd.read_csv('AUTOMPG.csv')
df.head()
```

Out[131…

	Unnamed: 0	displacement	cylinders	horsepower	weight	acceleration	model_year	origin	mpg
0	0	307.0	8	130.0	3504	12.0	70	1	18.0
1	1	350.0	8	165.0	3693	11.5	70	1	15.0
2	2	318.0	8	150.0	3436	11.0	70	1	18.0
3	3	304.0	8	150.0	3433	12.0	70	1	16.0
4	4	302.0	8	140.0	3449	10.5	70	1	17.0

In [133…
```
#Engineers at XYZ Custom Cars want to know how many cars are Fuel efficient MPG > 29, Horsepower < 93.5, Weight
df.loc[(df['mpg'] > 29) & (df['horsepower'] < 93.5) & (df['weight'] < 2500)]
```

Out[133…

	Unnamed: 0	displacement	cylinders	horsepower	weight	acceleration	model_year	origin	mpg
51	51	79.0	4	70.0	2074	19.5	71	2	30.0
52	52	88.0	4	76.0	2065	14.5	71	2	30.0
53	53	71.0	4	65.0	1773	19.0	71	3	31.0
54	54	72.0	4	69.0	1613	18.0	71	3	35.0
129	129	79.0	4	67.0	1950	19.0	74	3	31.0
...	...	...	...	...	...	...	...	...	...
384	384	91.0	4	67.0	1965	15.7	82	3	32.0
385	385	91.0	4	67.0	1995	16.2	82	3	38.0
391	391	135.0	4	84.0	2370	13.0	82	1	36.0
394	394	97.0	4	52.0	2130	24.6	82	2	44.0
395	395	135.0	4	84.0	2295	11.6	82	1	32.0

81 rows × 9 columns

In [135…

```
#Engineers at XYZ Custom Cars want to know how many cars are Muscle cars Displacement >262, Horsepower > 126, W
df.loc[(df['displacement'] > 262) & (df['horsepower'] > 126) & (df['weight'] >=2800) & (df['weight'] <= 3600)]
```

Out[135…

	Unnamed: 0	displacement	cylinders	horsepower	weight	acceleration	model_year	origin	mpg
0	0	307.0	8	130.0	3504	12.0	70	1	18.0
2	2	318.0	8	150.0	3436	11.0	70	1	18.0
3	3	304.0	8	150.0	3433	12.0	70	1	16.0
4	4	302.0	8	140.0	3449	10.5	70	1	17.0
10	10	383.0	8	170.0	3563	10.0	70	1	15.0
13	13	455.0	8	225.0	3086	10.0	70	1	14.0
121	121	318.0	8	150.0	3399	11.0	73	1	15.0
166	166	302.0	8	129.0	3169	12.0	75	1	13.0
251	251	302.0	8	139.0	3570	12.8	78	1	20.2
262	262	305.0	8	145.0	3425	13.2	78	1	19.2
264	264	302.0	8	139.0	3205	11.2	78	1	18.1

In [40]:

```
#Engineers at XYZ Custom Cars want to know how many cars are SUVs Horsepower > 140 , Weight > 4500
df.loc[(df['horsepower'] > 140) & (df['weight'] >=4500)]
```

Out[40]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year	origin	name
25	10.0	8	360.0	215.0	4615	14.0	70	usa	NaN
28	9.0	8	304.0	193.0	4732	18.5	70	NaN	NaN
42	12.0	8	383.0	180.0	4955	11.5	71	NaN	NaN
43	13.0	8	400.0	170.0	4746	12.0	71	NaN	NaN
44	13.0	8	400.0	175.0	5140	12.0	71	NaN	NaN
67	11.0	8	429.0	208.0	4633	11.0	72	NaN	NaN
68	13.0	8	350.0	155.0	4502	13.5	72	NaN	NaN
90	12.0	8	429.0	198.0	4952	11.5	73	NaN	NaN
94	13.0	8	440.0	215.0	4735	11.0	73	NaN	NaN
95	12.0	8	455.0	225.0	4951	11.0	73	NaN	NaN
103	11.0	8	400.0	150.0	4997	14.0	73	NaN	NaN
104	12.0	8	400.0	167.0	4906	12.5	73	NaN	NaN
105	13.0	8	360.0	170.0	4654	13.0	73	NaN	NaN
137	13.0	8	350.0	150.0	4699	14.5	74	NaN	NaN
156	16.0	8	400.0	170.0	4668	11.5	75	NaN	NaN
159	14.0	8	351.0	148.0	4657	13.5	75	NaN	NaN

In [137…

```
#Engineers at XYZ Custom Cars want to know how many cars are Racecars Weight <2223, acceleration > 17
df.loc[(df['acceleration'] > 17) & (df['weight'] < 2223)]
```

Out[137…

	Unnamed: 0	displacement	cylinders	horsepower	weight	acceleration	model_year	origin	mpg
19	19	97.0	4	46.0	1835	20.5	70	2	26.0
32	32	98.0	4	NaN	2046	19.0	71	1	25.0
51	51	79.0	4	70.0	2074	19.5	71	2	30.0
53	53	71.0	4	65.0	1773	19.0	71	3	31.0
54	54	72.0	4	69.0	1613	18.0	71	3	35.0
55	55	97.0	4	60.0	1834	19.0	71	2	27.0
56	56	91.0	4	70.0	1955	20.5	71	1	26.0
79	79	96.0	4	69.0	2189	18.0	72	2	26.0
102	102	97.0	4	46.0	1950	21.0	73	2	26.0
117	117	68.0	4	49.0	1867	19.5	73	2	29.0
129	129	79.0	4	67.0	1950	19.0	74	3	31.0
131	131	71.0	4	65.0	1836	21.0	74	3	32.0
145	145	83.0	4	61.0	2003	19.0	74	3	32.0
181	181	91.0	4	53.0	1795	17.5	75	3	33.0
195	195	85.0	4	52.0	2035	22.2	76	1	29.0
196	196	98.0	4	60.0	2164	22.1	76	1	24.5
198	198	91.0	4	53.0	1795	17.4	76	3	33.0
216	216	98.0	4	68.0	2045	18.5	77	3	31.5
218	218	79.0	4	58.0	1825	18.6	77	2	36.0
244	244	90.0	4	48.0	1985	21.5	78	2	43.1
246	246	78.0	4	52.0	1985	19.4	78	3	32.8
247	247	85.0	4	70.0	2070	18.6	78	3	39.4
303	303	85.0	4	65.0	2020	19.2	79	3	31.8
310	310	89.0	4	60.0	1968	18.8	80	3	38.1
322	322	86.0	4	65.0	2110	17.9	80	3	46.6
324	324	85.0	4	65.0	2110	19.2	80	3	40.8
325	325	90.0	4	48.0	2085	21.7	80	2	44.3
330	330	85.0	4	NaN	1835	17.3	80	2	40.9
331	331	97.0	4	67.0	2145	18.0	80	3	33.8
346	346	97.0	4	67.0	2065	17.8	81	3	32.3
347	347	85.0	4	65.0	1975	19.4	81	3	37.0
348	348	89.0	4	62.0	2050	17.3	81	3	37.7
376	376	91.0	4	68.0	2025	18.2	82	3	37.0
377	377	91.0	4	68.0	1970	17.6	82	3	31.0
379	379	98.0	4	70.0	2125	17.3	82	1	36.0
394	394	97.0	4	52.0	2130	24.6	82	2	44.0

In [139…
```
#XYZ Custom cars want the data sorted according to the number of cylinders.
df.sort_values(by = 'cylinders')
```

Out[139…

	Unnamed: 0	displacement	cylinders	horsepower	weight	acceleration	model_year	origin	mpg
111	111	70.0	3	90.0	2124	13.5	73	3	18.0
71	71	70.0	3	97.0	2330	13.5	72	3	19.0
334	334	70.0	3	100.0	2420	12.5	80	3	23.7
243	243	80.0	3	110.0	2720	13.5	77	3	21.5
267	267	134.0	4	95.0	2560	14.2	78	3	27.5
...	...	...	...	...	...	...	...	...	...
86	86	304.0	8	150.0	3672	11.5	73	1	14.0
285	285	305.0	8	130.0	3840	15.4	79	1	17.0
286	286	302.0	8	129.0	3725	13.4	79	1	17.6
92	92	351.0	8	158.0	4363	13.0	73	1	13.0
0	0	307.0	8	130.0	3504	12.0	70	1	18.0

398 rows × 9 columns

In [141…
```
#There is a requirement in which the cars that have lowest acceleration must be assessed. It is also to be checi
df.sort_values(['acceleration', 'horsepower'], ascending = (1,0))
```

Out[141…

	Unnamed: 0	displacement	cylinders	horsepower	weight	acceleration	model_year	origin	mpg
11	11	340.0	8	160.0	3609	8.0	70	1	14.0
7	7	440.0	8	215.0	4312	8.5	70	1	14.0
9	9	390.0	8	190.0	3850	8.5	70	1	15.0
6	6	454.0	8	220.0	4354	9.0	70	1	14.0
116	116	400.0	8	230.0	4278	9.5	73	1	16.0
...	...	...	...	...	...	...	...	...	...
195	195	85.0	4	52.0	2035	22.2	76	1	29.0
59	59	97.0	4	54.0	2254	23.5	72	2	23.0
326	326	90.0	4	48.0	2335	23.7	80	2	43.4
394	394	97.0	4	52.0	2130	24.6	82	2	44.0
299	299	141.0	4	71.0	3190	24.8	79	2	27.2

398 rows × 9 columns

In [48]:
```
#The board of XYZ custom cars wants to know about minimum and maximum sum, mean and median of all the numerical

#Using list comprehension to get the numerical columns
list1 = [col for col in df.columns if df[col].dtype in ['float', 'int64']]
df[list1].agg(['min', 'max','sum','mean','median'])
```

Out[48]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
min	9.000000	3.000000	68.000000	46.000000	1.613000e+03	8.00000	70.00000
max	46.600000	8.000000	455.000000	230.000000	5.140000e+03	24.80000	82.00000
sum	9358.800000	2171.000000	76983.500000	41259.000000	1.182229e+06	6196.10000	30252.00000
mean	23.514573	5.454774	193.425879	104.189394	2.970425e+03	15.56809	76.01005
median	23.000000	4.000000	148.500000	92.000000	2.803500e+03	15.50000	76.00000

In [143…
```
#XYZ custom cars want to know the number of cars manufactured in each year.
df.groupby(['model_year']).count()['horsepower'] #all values will be same, so display any one column
```

Out[143…
```
model_year
70    29
71    27
72    28
73    40
74    26
75    30
76    34
77    28
78    36
79    29
80    27
81    28
82    30
Name: horsepower, dtype: int64
```

In [147…
```
#The engineers at XYZ Custom Cars want to know about the relationship between model year and acceleration of ca
df.groupby(['model_year']).mean()[['acceleration']]
```

Out[147…

model_year	acceleration
70	12.948276
71	15.142857
72	15.125000
73	14.312500
74	16.203704
75	16.050000
76	15.941176
77	15.435714
78	15.805556
79	15.813793
80	16.934483
81	16.306897
82	16.638710

In [70]:
```
#The engineers at XYZ Custom Cars want to know the frequency distribution of different number of cylinders acro
pd.crosstab(df['model-year'], df['cylinders'])
```

Out[70]:

cylinders	3	4	5	6	8
model-year					
70	0	7	0	4	18
71	0	13	0	8	7
72	1	14	0	0	13
73	1	11	0	8	20
74	0	15	0	7	5
75	0	12	0	12	6
76	0	15	0	10	9
77	1	14	0	5	8
78	0	17	1	12	6
79	0	12	1	6	10
80	1	25	1	2	0
81	0	21	0	7	1
82	0	28	0	3	0

In [155…
```
#The engineers at XYZ custom cars want to know the mean of all the numerical attributes of cars for each year
Pivot1 = pd.pivot_table(df, index = 'model_year', aggfunc=np.mean)
Pivot1
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_15752\1000159008.py:2: FutureWarning: The provided callable
<function mean at 0x0000012AD6769580> is currently using DataFrameGroupBy.mean. In a future version of pandas, t
he provided callable will be used directly. To keep current behavior pass the string "mean" instead.
  Pivot1 = pd.pivot_table(df, index = 'model_year', aggfunc=np.mean)
```

Out[155…

model_year	Unnamed: 0	acceleration	cylinders	displacement	horsepower	mpg	origin	weight
70	14.0	12.948276	6.758621	281.413793	147.827586	17.689655	1.310345	3372.793103
71	42.5	15.142857	5.571429	209.750000	107.037037	21.250000	1.428571	2995.428571
72	70.5	15.125000	5.821429	218.375000	120.178571	18.714286	1.535714	3237.714286
73	104.5	14.312500	6.375000	256.875000	130.475000	17.100000	1.375000	3419.025000
74	138.0	16.203704	5.259259	171.740741	94.230769	22.703704	1.666667	2877.925926
75	166.5	16.050000	5.600000	205.533333	101.066667	20.266667	1.466667	3176.800000
76	198.5	15.941176	5.647059	197.794118	101.117647	21.573529	1.470588	3078.735294
77	229.5	15.435714	5.464286	191.392857	105.071429	23.375000	1.571429	2997.357143
78	261.5	15.805556	5.361111	177.805556	99.694444	24.061111	1.611111	2861.805556
79	294.0	15.813793	5.827586	206.689655	101.206897	25.093103	1.275862	3055.344828
80	323.0	16.934483	4.137931	115.827586	77.481481	33.696552	2.206897	2436.655172
81	352.0	16.306897	4.620690	135.310345	81.035714	30.334483	1.965517	2522.931034
82	382.0	16.638710	4.193548	128.870968	81.466667	31.709677	1.645161	2453.548387

In [ ]:

In [ ]: