Data Science Practical

March 7, 2024

1 Initialisation

[1]:
```
import pandas as pd
import numpy  as np
from IPython.display import HTML, Markdown, Latex


def display_df(tp_df=None, index=False):
    tp_df = tp_df if tp_df is not None else df
    display(Markdown(tp_df.to_markdown(index=index)))
```

2 22/02/24

Create a dataframe to store data of 10 students, with the columns being "Name", "Age", "Semester I marks out of 600", "Semester II marks out of 500", and "Attendance"

1) Display details of students who scored more than 560 marks in sem I
2) Display details of students who scored less than 250 marks in sem II
3) Display details of student who scored minimum marks in sem II
4) Display details of student who scored maximum marks in sem II
5) Display details of students whose attendance is more than 75
6) Display details of students whose attendance is less than 50
7) Insert 2 new records
8) Add a column corresponding to percentage of marks of both semesters
9) Add a new column corresponding to grades:

Percentage	Grade
>=90	O
>=75 and <90	A+
>=60 and <75	A
>=50 and <60	B+
>=40 and <50	B
>40	F

[2]:
```
data = {
    'Name':                          ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',␣
 ↪'I', 'J'],
```

```
    'Age':                          [20, 21, 20, 22, 23, 20, 21, 22, 20, 21],
    'Semester I marks out of 600':  [213, 31, 57, 406, 417, 45, 217, 200, 588,␣
 ↪319],
    'Semester II marks out of 500': [198, 378, 133, 450, 283, 485, 193, 283,␣
 ↪236, 191],
    'Attendance':                   [76, 26, 53, 32, 50, 67, 92, 62, 44, 85]
}
df = pd.DataFrame(data)

print('\nOriginal data:')
display_df()

print('\n1) Students who scored more than 560 marks in sem I:')
ans = df.query('`Semester I marks out of 600` > 560')
display_df(ans, index=True)

print('\n2) Students who scored less than 250 marks in sem II:')
ans = df.query('`Semester II marks out of 500` < 250')
display_df(ans, index=True)

print('\n3) Student who scored minimum marks in sem II:')
min_marks = min(df['Semester II marks out of 500'])
ans       = df.query('`Semester II marks out of 500` == @min_marks')
display_df( ans , index=True )

print('\n4) Student who scored maximum marks in sem II:')
ans = df.sort_values(by='Semester II marks out of 500',ascending=False).head(1)
display_df(ans, index=True)

print('\n5) Students whose attendance is more than 75:')
ans = df.query('Attendance > 75')
display_df(ans, index=True)

print('\n6) Students whose attendance is less than 50:')
ans = df.query('Attendance < 50')
display_df(ans, index=True)

print('\n7) Inserted two new records:')
new_data = {
    'Name':                         ['K', 'L'],
    'Age':                          [22,  23],
    'Semester I marks out of 600':  [300, 400],
    'Semester II marks out of 500': [400, 300],
    'Attendance':                   [80,  40]
}
new_df = pd.DataFrame(new_data)
df     = pd.concat([df,new_df], ignore_index=True)
```

```
display_df(index=True)

print('\n8) Added the percentage column:')
df['Percentage'] = (df['Semester I marks out of 600'] + df['Semester II marks␣
 ↪out of 500']) / 11
df['Percentage'] = df['Percentage'].apply(lambda x: round(x,2))
display_df()

print('\n9) Added the grade column:')
def get_grade(x: float):
    if   x >= 90: return 'O'
    elif x >= 75: return 'A+'
    elif x >= 60: return 'A'
    elif x >= 50: return 'B+'
    elif x >= 40: return 'B'
    else:         return 'F'
df['Grade'] = df['Percentage'].apply(get_grade)
display_df()
```

Original data:

Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance
A	20	213	198	76
B	21	31	378	26
C	20	57	133	53
D	22	406	450	32
E	23	417	283	50
F	20	45	485	67
G	21	217	193	92
H	22	200	283	62
I	20	588	236	44
J	21	319	191	85

1) Students who scored more than 560 marks in sem I:

	Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance
8	I	20	588	236	44

2) Students who scored less than 250 marks in sem II:

	Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance
0	A	20	213	198	76
2	C	20	57	133	53
6	G	21	217	193	92
8	I	20	588	236	44
9	J	21	319	191	85

3) Student who scored minimum marks in sem II:

	Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance
2	C	20	57	133	53

4) Student who scored maximum marks in sem II:

	Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance
5	F	20	45	485	67

5) Students whose attendance is more than 75:

	Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance
0	A	20	213	198	76
6	G	21	217	193	92
9	J	21	319	191	85

6) Students whose attendance is less than 50:

	Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance
1	B	21	31	378	26
3	D	22	406	450	32
8	I	20	588	236	44

7) Inserted two new records:

	Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance
0	A	20	213	198	76
1	B	21	31	378	26
2	C	20	57	133	53
3	D	22	406	450	32
4	E	23	417	283	50
5	F	20	45	485	67
6	G	21	217	193	92
7	H	22	200	283	62
8	I	20	588	236	44
9	J	21	319	191	85
10	K	22	300	400	80
11	L	23	400	300	40

`8) Added the percentage column:`

Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance	Percentage
A	20	213	198	76	37.36
B	21	31	378	26	37.18
C	20	57	133	53	17.27
D	22	406	450	32	77.82
E	23	417	283	50	63.64
F	20	45	485	67	48.18
G	21	217	193	92	37.27
H	22	200	283	62	43.91
I	20	588	236	44	74.91
J	21	319	191	85	46.36
K	22	300	400	80	63.64
L	23	400	300	40	63.64

`9) Added the grade column:`

Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance	Percentage	Grade
A	20	213	198	76	37.36	F
B	21	31	378	26	37.18	F
C	20	57	133	53	17.27	F
D	22	406	450	32	77.82	A+
E	23	417	283	50	63.64	A
F	20	45	485	67	48.18	B
G	21	217	193	92	37.27	F

Name	Age	Semester I marks out of 600	Semester II marks out of 500	Attendance	Percentage	Grade
H	22	200	283	62	43.91	B
I	20	588	236	44	74.91	A
J	21	319	191	85	46.36	B
K	22	300	400	80	63.64	A
L	23	400	300	40	63.64	A

3 29/02/24

1. Create a DataFrame based on E-Commerce data and generate mean, mode, and median
2. Write a program to implement pivot() and pivot-table() on a DataFrame
3. Write a Program to read a CSV file and create its DataFrame
4. Consider the DataFrame QtrSales where each row contains the item category, item name and expenditure and group the rows by category, and print the average expenditure per category
5. Create a DataFrame having age, name, weight of five students. Write a program to display only the weight of first and fourth rows
6. Write a program to create a DataFrame to store weight, age and name of three people. Print the DataFrame and its transpose

[3]:
```
# 1
data = {
    'Order_ID':     ['101', '102', '103', '104', '105'],
    'Price':        [50, 20, 40, 50, 45],
    'Quantity':     [2, 3, 1, 2, 1]
}
df = pd.DataFrame(data)
print('\n\n\n1.1) Original dataframe:')
display_df(df)

mean_df   = df.mean(numeric_only=True)
mode_df   = df.mode(numeric_only=True)
median_df = df.median(numeric_only=True)

print('\n1.2) Means: ')
display_df(mean_df, index=True)
print('\n1.3) Modes: ')
display_df(mode_df, index=True)
print('\n1.4) Medians: ')
display_df(median_df, index=True)



# 2
data = {
```

```
    'Day':         ['Monday', 'Monday', 'Tuesday', 'Tuesday', 'Wednesday',␣
 ↪'Wednesday'],
    'City':        ['Delhi', 'Mumbai', 'Delhi', 'Mumbai', 'Delhi', 'Mumbai'],
    'Temperature': [32, 34, 33, 35, 34, 36],
}
df = pd.DataFrame(data)
print('\n\n\n2.1) Dataframe for pivot(): ')
display_df(df)

pivot_df = df.pivot(index='Day', columns='City', values='Temperature')
print('\n2.2) pivot(): ')
display_df(pivot_df, index=True)

data = {
    'Day':         ['Monday', 'Monday', 'Monday', 'Tuesday', 'Tuesday',␣
 ↪'Tuesday'],
    'City':        ['Delhi', 'Delhi', 'Mumbai', 'Delhi', 'Mumbai', 'Mumbai'],
    'Temperature': [32, 33, 36, 33, 36, 37],
}
df = pd.DataFrame(data)
print('\n2.3) Dataframe for pivot_table(): ')
display_df(df)

pivot_table_df = df.pivot_table(index='Day', columns='City',␣
 ↪values='Temperature', aggfunc='count')
print('\n2.4) pivot_table(): ')
display_df(pivot_table_df, index=True)



# 3
filename = 'data.csv'
print(f'\n\n\n3.1) Contents of {filename}:')
with open(filename) as f:
    print(f.read())

df = pd.read_csv(filename)
print('\n3.2) Dataframe:')
display_df(df)



# 4
QtrSales = pd.DataFrame({
    'category':    ['Electronics', 'Electronics', 'Fashion', 'Fashion',␣
 ↪'Electronics', 'Fashion'],
```

```
    'item_name':   ['Laptop', 'Headphones', 'T-Shirt', 'Jeans', 'Smartphone',␣
 ↪'Shoes'],
    'expenditure': [1200, 100, 31, 50, 800, 60]
})
print('\n\n\n4.1) Original dataframe:')
display_df(QtrSales)

grouped = QtrSales.groupby(by='category')['expenditure']
mean_df = grouped.mean()
print('\n4.2) Average expenditure per category:')
display_df(mean_df, index=True)



# 5
data = {
    'name':   ['John', 'Emma', 'Michael', 'Sophia', 'William'],
    'age':    [20, 21, 22, 20, 23],
    'weight': [70, 65, 75, 68, 72]
}
df = pd.DataFrame(data)
print('\n\n\n5.1) Original dataframe:')
display_df(df, index=True)

weight_df = df.loc[ [0,3] , ['weight'] ]
print('\n5.2) Weight of the first and fourth rows:')
display_df(weight_df, index=True)



# 6
data = {
    'name':   ['John', 'Emma', 'Michael', 'Sophia', 'William'],
    'age':    [20, 21, 22, 20, 23],
    'weight': [70, 65, 75, 68, 72]
}
df = pd.DataFrame(data)
print('\n\n\n6.1) Original dataframe:')
display_df(df, index=True)

print('\n6.2) Transpose of the dataframe:')
display_df(df.T, index=True)
```

```
1.1) Original dataframe:
```

Order_ID	Price	Quantity
101	50	2
102	20	3
103	40	1
104	50	2
105	45	1

1.2) Means:

	0
Price	41
Quantity	1.8

1.3) Modes:

	Price	Quantity
0	50	1
1	nan	2

1.4) Medians:

	0
Price	45
Quantity	2

2.1) Dataframe for pivot():

Day	City	Temperature
Monday	Delhi	32
Monday	Mumbai	34
Tuesday	Delhi	33
Tuesday	Mumbai	35
Wednesday	Delhi	34
Wednesday	Mumbai	36

2.2) pivot():

Day	Delhi	Mumbai
Monday	32	34
Tuesday	33	35
Wednesday	34	36

2.3) Dataframe for pivot_table():

Day	City	Temperature
Monday	Delhi	32
Monday	Delhi	33
Monday	Mumbai	36
Tuesday	Delhi	33
Tuesday	Mumbai	36
Tuesday	Mumbai	37

2.4) pivot_table():

Day	Delhi	Mumbai
Monday	2	1
Tuesday	1	2

3.1) Contents of data.csv:

```
Name,Age,Gender
Ram,16,M
Manish,18,M
Sahil,15,M
Amrit,20,F
Mark,19,M
```

3.2) Dataframe:

Name	Age	Gender
Ram	16	M
Manish	18	M
Sahil	15	M
Amrit	20	F
Mark	19	M

4.1) Original dataframe:

category	item_name	expenditure
Electronics	Laptop	1200
Electronics	Headphones	100
Fashion	T-Shirt	31
Fashion	Jeans	50
Electronics	Smartphone	800
Fashion	Shoes	60

4.2) Average expenditure per category:

category	expenditure
Electronics	700
Fashion	47

5.1) Original dataframe:

	name	age	weight
0	John	20	70
1	Emma	21	65
2	Michael	22	75
3	Sophia	20	68
4	William	23	72

5.2) Weight of the first and fourth rows:

	weight
0	70
3	68

6.1) Original dataframe:

	name	age	weight
0	John	20	70
1	Emma	21	65
2	Michael	22	75
3	Sophia	20	68
4	William	23	72

6.2) Transpose of the dataframe:

	0	1	2	3	4
name	John	Emma	Michael	Sophia	William
age	20	21	22	20	23
weight	70	65	75	68	72

4 7/3/24

1. Create a pandas series from a dictionary of values and an ndarray
2. Perform sorting on Series data and DataFrames
3. Two Series object, Population stores the details of four metro cities of India and another object AvgIncome stores the total average income reported in four years in these cities. Calculate income per capita for each of these metro cities
4. Series objects temp1, temp2, temp3, and temp 4 stores the temperature of days of week 1, week 2, week 3, week 4. Write a script to:
   1) Print average temperature per week
   2) Print average temperature of entire month

[4]:
```
data = {
    'Name':   np.array([ 'Ram', 'Manish', 'Sahil', 'Amrit', 'Mark' ]),
    'Age':    np.array([ 16, 18, 15, 20, 19 ]),
    'Gender': np.array([ 'M', 'M', 'M', 'F', 'M' ])
}
df = pd.DataFrame(data)
print('1) Dataframe created from a dictionary of values and nd array:')
display_df(df)



data = {
    'name':    ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
    'year':    [2012, 2012, 2013, 2014, 2014],
    'reports': [4, 24, 31, 2, 3]
}
df        = pd.DataFrame(data)
print('\n\n\n2.1) Original dataframe:')
display_df(df)
```

```
sorted_df = df.sort_values(by='reports', ascending=False)
print('\n2.2) Sorted dataframe (based on reports) in descending order:')
display_df(sorted_df)

s = pd.Series([3, 1, 2, 3, 4])
print('\n2.3) Original series:')
display_df(s)
s.sort_values(inplace=True)
print('\n2.4) Sorted series in ascending order:')
display_df(s)



Population = pd.Series({'Delhi': 20000000, 'Mumbai': 22000000, 'Bangalore':␣
 ↪12000000, 'Kolkata': 15000000})
AvgIncome  = pd.Series({'Delhi': 350000,   'Mumbai': 320000,   'Bangalore':␣
 ↪300000,   'Kolkata': 280000})
print('\n\n\n3.1) Population:')
display_df(Population, index=True)
print('\n3.2) AvgIncome:')
display_df(AvgIncome, index=True)

income_per_capita = AvgIncome / Population
print("\n3.3) Income per capita for each metro city:")
display_df(income_per_capita, index=True)



Temp1 = pd.Series([20, 22, 21, 23, 25], index=['Monday', 'Tuesday',␣
 ↪'Wednesday', 'Thursday', 'Friday'])
Temp2 = pd.Series([24, 23, 22, 21, 20], index=['Monday', 'Tuesday',␣
 ↪'Wednesday', 'Thursday', 'Friday'])
Temp3 = pd.Series([22, 23, 24, 25, 26], index=['Monday', 'Tuesday',␣
 ↪'Wednesday', 'Thursday', 'Friday'])
Temp4 = pd.Series([25, 24, 23, 22, 21], index=['Monday', 'Tuesday',␣
 ↪'Wednesday', 'Thursday', 'Friday'])
temp_df = pd.DataFrame({'Week 1': Temp1, 'Week 2': Temp2, 'Week 3': Temp3,␣
 ↪'Week 4': Temp4})
print('\n\n\n4.1) Original data:')
display_df(temp_df, index=True)

mean_df = temp_df.mean()
print("\n4.2) Average temperature per week:")
display_df(mean_df, index=True)
```

```
print("\n4.3) Average temperature of the entire month: ", temp_df.values.mean())
```

```
1) Dataframe created from a dictionary of values and nd array:
```

Name	Age	Gender
Ram	16	M
Manish	18	M
Sahil	15	M
Amrit	20	F
Mark	19	M

```
2.1) Original dataframe:
```

name	year	reports
Jason	2012	4
Molly	2012	24
Tina	2013	31
Jake	2014	2
Amy	2014	3

```
2.2) Sorted dataframe (based on reports) in descending order:
```

name	year	reports
Tina	2013	31
Molly	2012	24
Jason	2012	4
Amy	2014	3
Jake	2014	2

```
2.3) Original series:
```

0
3
1
2
3
4

2.4) Sorted series in ascending order:

0
1
2
3
3
4

3.1) Population:

	0
Delhi	2e+07
Mumbai	2.2e+07
Bangalore	1.2e+07
Kolkata	1.5e+07

3.2) AvgIncome:

	0
Delhi	350000
Mumbai	320000
Bangalore	300000
Kolkata	280000

3.3) Income per capita for each metro city:

	0
Delhi	0.0175
Mumbai	0.0145455
Bangalore	0.025
Kolkata	0.0186667

4.1) Original data:

	Week 1	Week 2	Week 3	Week 4
Monday	20	24	22	25
Tuesday	22	23	23	24
Wednesday	21	22	24	23
Thursday	23	21	25	22
Friday	25	20	26	21

```
4.2) Average temperature per week:
```

	0
Week 1	22.2
Week 2	22
Week 3	24
Week 4	23

```
4.3) Average temperature of the entire month:  22.8
```