2000257-datascience-pandas-lab5-1

April 23, 2024

1 NANDANI AGRAWAL

2 22000257

3 BATCH-A

DATA SCIENCE PANDAS LAB-5

4 TASK-1

[3]:
```
import pandas as pd
```

5 Create 3 list viz. productId, Product Name, Product Price with 5 elements each.Add this element to create a DataFrame.

[133]:
```
productId = [1, 2, 3, 4, 5]
productName = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']
productPrice = [10.5, 20.0, 15.75, 30.25, 25.5]
df = pd.DataFrame({'ProductID': productId, 'ProductName': productName,␣
 ↪'ProductPrice': productPrice})
df
```

[133]:

	ProductID	ProductName	ProductPrice
0	1	Product A	10.50
1	2	Product B	20.00
2	3	Product C	15.75
3	4	Product D	30.25
4	5	Product E	25.50

6 Extract one by one column and display separately from above dataframe.

[58]:
```
df['ProductID']
```

[58]:
```
0    1
1    2
2    3
3    4
4    5
Name: ProductID, dtype: int64
```

[59]:
```
df['ProductName']
```

[59]:
```
0    Product A
1    Product B
2    Product C
3    Product D
4    Product E
Name: ProductName, dtype: object
```

[60]:
```
df['ProductPrice']
```

[60]:
```
0    10.50
1    20.00
2    15.75
3    30.25
4    25.50
Name: ProductPrice, dtype: float64
```

7 Display separately each and every row from an above dataframe. (use loc or iloc function)

[62]:
```
df.loc[1]
```

[62]:
```
ProductID               2
ProductName     Product B
ProductPrice         20.0
Name: 1, dtype: object
```

[63]:
```
df.iloc[2]
```

[63]:
```
ProductID               3
ProductName     Product C
ProductPrice        15.75
Name: 2, dtype: object
```

[64]:
```
df.iloc[3]
```

[64]:
```
ProductID               4
ProductName     Product D
```

```
ProductPrice        30.25
Name: 3, dtype: object
```

8 Display product name from 3 row using loc function.

[66]:
```
df.loc[2, 'ProductName']
```

[66]:
```
'Product C'
```

9 Display price located in 2 row using iloc function

[68]:
```
df.iloc[1]['ProductPrice']
```

[68]:
```
20.0
```

10 Display 2nd to 4th row data excluding price

[70]:
```
df.iloc[1:4, :-1]
```

[70]:
```
   ProductID ProductName
1          2   Product B
2          3   Product C
3          4   Product D
```

11 Display 1st to 3rd row price data only

[72]:
```
df.loc[:2, 'ProductPrice']
```

[72]:
```
0    10.50
1    20.00
2    15.75
Name: ProductPrice, dtype: float64
```

12 Change marks in python column to 41 for students who scored less than 40

[74]:
```
data = {'Name': ['Slice', 'Bob', 'Charlie', 'David', 'Eva'],
        'Python': [35, 50, 75, 90, 40],
        'Data Science': [65, 80, 55, 70, 99],
        'Math': [85, 40, 60, 30, 75]}

df = pd.DataFrame(data)
```

```
df
```

[74]:

	Name	Python	Data Science	Math
0	Slice	35	65	85
1	Bob	50	80	40
2	Charlie	75	55	60
3	David	90	70	30
4	Eva	40	99	75

[75]:
```
df.loc[df['Python'] < 40, 'Python'] = 41
df
```

[75]:

	Name	Python	Data Science	Math
0	Slice	41	65	85
1	Bob	50	80	40
2	Charlie	75	55	60
3	David	90	70	30
4	Eva	40	99	75

13 Add 2 marks to all subjects, and limit to 100

[77]:
```
a=df[['Python','Data Science','Math']]+2
a
```

[77]:

	Python	Data Science	Math
0	43	67	87
1	52	82	42
2	77	57	62
3	92	72	32
4	42	101	77

[78]:
```
a[a>100]=100
a
```

[78]:

	Python	Data Science	Math
0	43	67	87
1	52	82	42
2	77	57	62
3	92	72	32
4	42	100	77

14 Display students with more than 50 in Python and more than 60 in Data Science

[80]:
```
a[(a['Python']>50) & (a['Data Science']>60)]
```

[80]:
```
   Python  Data Science  Math
1      52            82    42
3      92            72    32
```

15 Display students records whose name starts with letter S.

[82]:
```
df[df['Name'].str.startswith('S')]
```

[82]:
```
    Name  Python  Data Science  Math
0  Slice      41            65    85
```

16 . Display students records whose name ends with letter a.

[84]:
```
df[df['Name'].str.endswith('a')]
```

[84]:
```
   Name  Python  Data Science  Math
4  Eva       40            99    75
```

17 . Calculate total and percentage for all the students.

[88]:
```
df['total']=df[['Python', 'Data Science', 'Math']].sum(axis=1)
df['total']
```

[88]:
```
0    191
1    170
2    190
3    190
4    214
Name: total, dtype: int64
```

[89]:
```
df['Percentage'] = (df['total'] / 300) * 100
df['Percentage']
```

[89]:
```
0    63.666667
1    56.666667
2    63.333333
3    63.333333
4    71.333333
Name: Percentage, dtype: float64
```

18 Calculate Status (PASS / FAIL) for the Students.

[ ]:
```
import numpy as np
df['Status'] = np.where(df['Percentage'] >= 40, 'PASS', 'FAIL')
df['Status']
```

19 Calculate percentage-based grade value

[ ]:
```
def calculate_grade(percentage):
    if percentage >= 90:
        return 'A'
    elif percentage >= 80:
        return 'B'
    elif percentage >= 70:
        return 'C'
    elif percentage >= 60:
        return 'D'
    elif percentage >= 50:
        return 'E'
    else:
        return 'F'

df['Grade'] = df['Percentage'].apply(calculate_grade)
df['Grade']
```

20 Calculate overall columns total, maximum, minimum, and standard deviation

[ ]:
```
df.describe()
```

21 Convert DataFrame to Excel, JSON, and HTML

[ ]:
```
df.to_excel('output.xlsx', index=False)
```

[ ]:
```
df.to_json('output.json', orient='records')
```

[ ]:
```
df.to_html('output.html', index=False)
```

22 Display meta-data for each column

[37]:
```
df.dtypes
```

[37]:
```
Name            object
Python           int64
Data Science     int64
Math             int64
dtype: object
```

23 Convert DataFrame to dictionary and add 500 to each element in Product Price column

[53]:
```
df_dict = df.to_dict(orient='list')
df_dict['ProductPrice'] = [price + 500 for price in df_dict['ProductPrice']]
# Convert back to DataFrame
df_updated = pd.DataFrame(df_dict)
df_updated
```

[53]:

	ProductID	ProductName	ProductPrice
0	1	Product A	510.50
1	2	Product B	520.00
2	3	Product C	515.75
3	4	Product D	530.25
4	5	Product E	525.50

24 TASK-2

25 Import pandas as pd.

[41]:
```
import pandas as pd
```

26 Read Salaries.csv as a dataframe called sal

[43]:
```
sal = pd.read_csv('Salaries.csv')
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_7880\1748292466.py:1: DtypeWarning:
Columns (3,4,5,6,12) have mixed types. Specify dtype option on import or set
low_memory=False.
  sal = pd.read_csv('Salaries.csv')
```

27 Check the head of the DataFrame

[45]:
```
sal.head()
```

[45]:

	Id	EmployeeName	JobTitle	\
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	

```
1   2       GARY JIMENEZ                 CAPTAIN III (POLICE DEPARTMENT)
2   3     ALBERT PARDINI                 CAPTAIN III (POLICE DEPARTMENT)
3   4  CHRISTOPHER CHONG            WIRE ROPE CABLE MAINTENANCE MECHANIC
4   5    PATRICK GARDNER    DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)

     BasePay OvertimePay   OtherPay Benefits   TotalPay  TotalPayBenefits  \
0  167411.18         0.0  400184.25      NaN  567595.43         567595.43
1  155966.02   245131.88  137811.38      NaN  538909.28         538909.28
2  212739.13   106088.18    16452.6      NaN  335279.91         335279.91
3    77916.0    56120.71   198306.9      NaN  332343.61         332343.61
4   134401.6      9737.0  182234.59      NaN  326373.19         326373.19

   Year  Notes         Agency Status
0  2011    NaN  San Francisco    NaN
1  2011    NaN  San Francisco    NaN
2  2011    NaN  San Francisco    NaN
3  2011    NaN  San Francisco    NaN
4  2011    NaN  San Francisco    NaN
```

28 Use the .info() method to find out how many entries there are

[94]:
```
sal['BasePay'] = pd.to_numeric(sal['BasePay']).astype(float)
sal['OvertimePay'] = pd.to_numeric(sal['OvertimePay']).astype(float)
sal['OtherPay'] = pd.to_numeric(sal['OtherPay']).astype(float)
sal['Benefits'] = pd.to_numeric(sal['Benefits']).astype(float)
```

[95]:
```
sal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Id                148654 non-null  int64
 1   EmployeeName      148654 non-null  object
 2   JobTitle          148654 non-null  object
 3   BasePay           148045 non-null  float64
 4   OvertimePay       148650 non-null  float64
 5   OtherPay          148650 non-null  float64
 6   Benefits          112491 non-null  float64
 7   TotalPay          148654 non-null  float64
 8   TotalPayBenefits  148654 non-null  float64
 9   Year              148654 non-null  int64
 10  Notes             0 non-null       float64
 11  Agency            148654 non-null  object
 12  Status            38119 non-null   object
```

```
dtypes: float64(7), int64(2), object(4)
memory usage: 14.7+ MB
```

29 What is the average BasePay?

[97]:
```
sal['BasePay'].mean()
```

[97]:
```
66325.4488404877
```

30 What is the highest amount of OvertimePay in the dataset ?

[99]:
```
sal['OvertimePay'].max()
```

[99]:
```
245131.88
```

31 What is the job title of JOSEPH DRISCOLL ? Note: Use all caps, otherwise you may get an answer that doesn't match up (there is also a lowercase Joseph Driscoll).

[101]:
```
sal[sal['EmployeeName']=='JOSEPH DRISCOLL']['JobTitle']
```

[101]:
```
24    CAPTAIN, FIRE SUPPRESSION
Name: JobTitle, dtype: object
```

32 How much does JOSEPH DRISCOLL make (including benefits)?

[103]:
```
sal[sal['EmployeeName']=='JOSEPH DRISCOLL']['TotalPayBenefits']
```

[103]:
```
24    270324.91
Name: TotalPayBenefits, dtype: float64
```

33 What is the name of highest paid person (including benefits)?

[107]:
```
sal[sal['TotalPayBenefits']== sal['TotalPayBenefits'].max()]
```

[107]:
```
    Id    EmployeeName                                      JobTitle  \
0   1  NATHANIEL FORD  GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY

     BasePay  OvertimePay   OtherPay  Benefits   TotalPay  TotalPayBenefits  \
0  167411.18          0.0  400184.25       NaN  567595.43         567595.43
```

```
   Year  Notes         Agency Status
0  2011    NaN  San Francisco    NaN
```

[108]:
```
sal.loc[sal['TotalPayBenefits'].idxmax()]
```

[108]:
```
Id                                                          1
EmployeeName                                    NATHANIEL FORD
JobTitle            GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY
BasePay                                                167411.18
OvertimePay                                                  0.0
OtherPay                                               400184.25
Benefits                                                     NaN
TotalPay                                               567595.43
TotalPayBenefits                                       567595.43
Year                                                        2011
Notes                                                        NaN
Agency                                             San Francisco
Status                                                       NaN
Name: 0, dtype: object
```

34 What is the name of lowest paid person (including benefits)? Do you notice something strange about how much he or she is paid?

[112]:
```
sal[sal['TotalPayBenefits']== sal['TotalPayBenefits'].min()]
```

[112]:
```
            Id EmployeeName                    JobTitle  BasePay  OvertimePay  \
148653  148654    Joe Lopez  Counselor, Log Cabin Ranch      0.0          0.0

        OtherPay  Benefits  TotalPay  TotalPayBenefits  Year  Notes  \
148653   -618.13       0.0   -618.13           -618.13  2014    NaN

               Agency Status
148653  San Francisco     PT
```

[113]:
```
sal.loc[sal['TotalPayBenefits'].idxmax()]['EmployeeName']
```

[113]:
```
'NATHANIEL FORD'
```

35 What was the average (mean) BasePay of all employees per year? (2011-2014) ?

[118]:
```
sal.groupby('Year')['BasePay'].mean()
```

[118]:
```
Year
2011    63595.956517
2012    65436.406857
2013    69630.030216
2014    66564.421924
Name: BasePay, dtype: float64
```

36 How many unique job titles are there?

[120]:
```
sal['JobTitle'].nunique()
```

[120]:
```
2159
```

37 What are the top 5 most common jobs?

[122]:
```
sal['JobTitle'].value_counts().head(5)
```

[122]:
```
JobTitle
Transit Operator                7036
Special Nurse                   4389
Registered Nurse                3736
Public Svc Aide-Public Works    2518
Police Officer 3                2421
Name: count, dtype: int64
```

38 How many Job Titles were represented by only one person in 2013? (e.g. Job Titles with only one occurence in 2013?)

[124]:
```
sum(sal[sal['Year']==2013]['JobTitle'].value_counts() == 1)
```

[124]:
```
202
```

39 How many people have the word Chief in their job title?

[126]:
```
def chief_string(title):
    if 'chief' in title.lower():
        return True
```

```
    else:
        return False
```

[127]:
```
sum(sal['JobTitle'].apply(lambda x: chief_string(x)))
```

[127]:
```
627
```

40 Is there a correlation between the length of the Job Title string and Salary?

[130]:
```
sal['JobTitleLength'] = sal['JobTitle'].apply(len)
sal['JobTitleLength'].corr(sal['TotalPayBenefits'])
```

[130]:
```
-0.03687844593260676
```

[131]:
```
sal['title_len'] = sal['JobTitle'].apply(len)
```

[132]:
```
sal[['title_len','TotalPayBenefits']].corr() # No correlation.
```

[132]:

	title_len	TotalPayBenefits
title_len	1.000000	-0.036878
TotalPayBenefits	-0.036878	1.000000

[ ]:
```

```