p7_and_p8

September 15, 2023

p7

```
[ ]: import numpy as np
     dtypes = [('emp_id', int), ('last_name', 'U20'), ('first_name', 'U20'),␣
      ↪('gender', 'U20'), ('Title', 'U20')]

     employee_table = np.array([], dtype=dtypes)
     emp1 = (1000, "Trbati", "Yolanda", "F", "Programmer")
     emp2 = (1001, "Kleinn", "Joel", "M", "Programmer")
     emp3 = (1002, "Ginsburg", "Laura", "F", "President")
     emp4 = (1003, "Cox", "Jennifer", "F", "Programmer")
     emp5 = (1005, "Ziada", "Mauri", "M", "Product Designer")
     emp6 = (1006, "Keyser", "Cara", "F", "Account Executive")
     emp7 = (1010, "Smith", "Roxie", "M", "Programmer")
     emp8 = (1011, "Nelson", "Robert", "M", "Programmer")
     emp9 = (1012, "Sachsen", "Lars", "M", "Support Technician")
     emp10 = (1013, "Shannon", "Don", "M", "Product Designer")

     employee_table = np.append(employee_table, np.array([emp1, emp2,␣
      ↪emp3,emp4,emp5,emp6,emp7,emp8,emp9,emp10], dtype=dtypes))
     print(employee_table)
```

```
[(1000, 'Trbati', 'Yolanda', 'F', 'Programmer')
 (1001, 'Kleinn', 'Joel', 'M', 'Programmer')
 (1002, 'Ginsburg', 'Laura', 'F', 'President')
 (1003, 'Cox', 'Jennifer', 'F', 'Programmer')
 (1005, 'Ziada', 'Mauri', 'M', 'Product Designer')
 (1006, 'Keyser', 'Cara', 'F', 'Account Executive')
 (1010, 'Smith', 'Roxie', 'M', 'Programmer')
 (1011, 'Nelson', 'Robert', 'M', 'Programmer')
 (1012, 'Sachsen', 'Lars', 'M', 'Support Technician')
 (1013, 'Shannon', 'Don', 'M', 'Product Designer')]
```

1. How many Male employees are in a company?

```
[ ]: male_emp = employee_table[employee_table['gender'] == 'M']
     nm= len(male_emp)
     print("Number of Male employees in the Company are :",nm)
```

```
Number of Male employees in the Company are : 6
```

2.Display the details of employees whose Last_Name starts with S.

```
[ ]: s_emp = employee_table[np.char.startswith(employee_table['last_name'], 'S')]
     print(s_emp)
```

```
[(1010, 'Smith', 'Roxie', 'M', 'Programmer')
 (1012, 'Sachsen', 'Lars', 'M', 'Support Technician')
 (1013, 'Shannon', 'Don', 'M', 'Product Designer')]
```

3.Sort the Female Employee details in descending order based on First_Name

```
[ ]: sf_emp = np.sort(employee_table, order='first_name')[::-1]
     print(sf_emp)
```

```
[(1000, 'Trbati', 'Yolanda', 'F', 'Programmer')
 (1010, 'Smith', 'Roxie', 'M', 'Programmer')
 (1011, 'Nelson', 'Robert', 'M', 'Programmer')
 (1005, 'Ziada', 'Mauri', 'M', 'Product Designer')
 (1002, 'Ginsburg', 'Laura', 'F', 'President')
 (1012, 'Sachsen', 'Lars', 'M', 'Support Technician')
 (1001, 'Kleinn', 'Joel', 'M', 'Programmer')
 (1003, 'Cox', 'Jennifer', 'F', 'Programmer')
 (1013, 'Shannon', 'Don', 'M', 'Product Designer')
 (1006, 'Keyser', 'Cara', 'F', 'Account Executive')]
```

4.Extract 1D array and reshape it into 2D array.

```
[ ]: empid_1d = employee_table['emp_id']
     empid_2d = empid_1d.reshape(-1, 1)
     print("1-D array was ",empid_1d)
     print("2-D array is ",empid_2d)
```

```
1-D array was  [1000 1001 1002 1003 1005 1006 1010 1011 1012 1013]
2-D array was  [[1000]
 [1001]
 [1002]
 [1003]
 [1005]
 [1006]
 [1010]
 [1011]
 [1012]
 [1013]]
```

5.Extract the below matrix using Boolean and Fancy indexing.

```
[ ]: start_emp_id = 1002
     end_emp_id = 1012
     mask = (employee_table['emp_id'] >= start_emp_id) & (employee_table['emp_id']␣
      ↪<= end_emp_id)
```

```
columns_to_extract = ['emp_id', 'last_name', 'Title']
filtered_data = employee_table[mask][columns_to_extract]
print(filtered_data)
```

```
[(1002, 'Ginsburg', 'President') (1003, 'Cox', 'Programmer')
 (1005, 'Ziada', 'Product Designer') (1006, 'Keyser', 'Account Executive')
 (1010, 'Smith', 'Programmer') (1011, 'Nelson', 'Programmer')
 (1012, 'Sachsen', 'Support Technician')]
```

p8

1. Import the domain dataset that you identiied with missing values and perform the following.

For each output, give the interpretation with respect to the imported dataset.

[ ]:
```
import numpy as np
import pandas as pd
d1=pd.read_csv("C:/Users/Hp/Downloads/archive (2)/student-mat.csv")
```

2.Read the csv file and create and understand the data frame using describe(), shape, info().

[ ]:
```
print(d1)
print("DataFrame Description:")
print(d1.describe())
print("\nDataFrame Shape:")
print(d1.shape)
print("\nDataFrame Info:")
print(d1.info())
```

```
    school sex  age address famsize Pstatus  Medu  Fedu      Mjob      Fjob
0       GP   F   18       U     GT3       A     4     4   at_home   teacher  \
1       GP   F   17       U     GT3       T     1     1   at_home     other
2       GP   F   15       U     LE3       T     1     1   at_home     other
3       GP   F   15       U     GT3       T     4     2    health  services
4       GP   F   16       U     GT3       T     3     3     other     other
..     … ..  …     …     …     …   …   …       …       …
390     MS   M   20       U     LE3       A     2     2  services  services
391     MS   M   17       U     LE3       T     3     1  services  services
392     MS   M   21       R     GT3       T     1     1     other     other
393     MS   M   18       R     LE3       T     3     2  services     other
394     MS   M   19       U     LE3       T     1     1     other   at_home

     … famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0    …      4        3      4     1     1      3        6   5   6   6
1    …      5        3      3     1     1      3        4   5   5   6
2    …      4        3      2     2     3      3       10   7   8  10
3    …      3        2      2     1     1      5        2  15  14  15
4    …      4        3      2     1     2      5        4   6  10  10
..   …    …      …    …   …   …    …      …  ..  ..  ..
390  …      5        5      4     4     5      4       11   9   9   9
```

```
391  …      2        4      5     3     4      2        3  14  16  16
392  …      5        5      3     3     3      3        3  10   8   7
393  …      4        4      1     3     4      5        0  11  12  10
394  …      3        2      3     3     3      5        5   8   9   9

[395 rows x 33 columns]
DataFrame Description:
              age        Medu        Fedu  traveltime   studytime    failures  
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000  \
mean    16.696203    2.749367    2.521519    1.448101    2.035443    0.334177   
std      1.276043    1.094735    1.088201    0.697505    0.839240    0.743651   
min     15.000000    0.000000    0.000000    1.000000    1.000000    0.000000   
25%     16.000000    2.000000    2.000000    1.000000    1.000000    0.000000   
50%     17.000000    3.000000    2.000000    1.000000    2.000000    0.000000   
75%     18.000000    4.000000    3.000000    2.000000    2.000000    0.000000   
max     22.000000    4.000000    4.000000    4.000000    4.000000    3.000000   

           famrel    freetime       goout        Dalc        Walc      health  
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000  \
mean     3.944304    3.235443    3.108861    1.481013    2.291139    3.554430   
std      0.896659    0.998862    1.113278    0.890741    1.287897    1.390303   
min      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000   
25%      4.000000    3.000000    2.000000    1.000000    1.000000    3.000000   
50%      4.000000    3.000000    3.000000    1.000000    2.000000    4.000000   
75%      5.000000    4.000000    4.000000    2.000000    3.000000    5.000000   
max      5.000000    5.000000    5.000000    5.000000    5.000000    5.000000   

         absences          G1          G2          G3  
count  395.000000  395.000000  395.000000  395.000000  
mean     5.708861   10.908861   10.713924   10.415190  
std      8.003096    3.319195    3.761505    4.581443  
min      0.000000    3.000000    0.000000    0.000000  
25%      0.000000    8.000000    9.000000    8.000000  
50%      4.000000   11.000000   11.000000   11.000000  
75%      8.000000   13.000000   13.000000   14.000000  
max     75.000000   19.000000   19.000000   20.000000  

DataFrame Shape:
(395, 33)

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
None
```

3.Find if any missing values (null values) are in the data, handle all the rows with missing data in four different ways (delete, replace, fill, bill), and print the data frame.

[ ]:

```
df_deleted = d1.dropna()
df_replaced = d1.fillna(-1)
df_filled_forward = d1.ffill()
df_filled_backward = d1.bfill()
```

4.Filter based on any column using groupby().

[ ]:

```
grouped_data =d1.groupby('Mjob')['age'].mean()
print(grouped_data)
```

```
Mjob
```

```
at_home     16.966102
health      16.352941
other       16.751773
services    16.679612
teacher     16.517241
Name: age, dtype: float64
```

5.Select 20 samples randomly and Create a data frame with Hiraricle Index

[ ]:

```
random_samples = d1.sample(n=20)
hierarchical_df = random_samples.set_index(['school', 'sex'])
print("\nDataFrame with Hierarchical Index:")
print(hierarchical_df)
```

```
DataFrame with Hierarchical Index:
            age address famsize Pstatus  Medu  Fedu      Mjob      Fjob
school sex
GP     F     15       U     LE3       A     4     3     other     other  \
       F     16       U     LE3       T     2     4     other    health
MS     F     19       R     GT3       T     2     3  services     other
GP     M     16       U     LE3       T     4     3    health     other
       F     16       R     GT3       T     3     3  services     other
       M     16       U     LE3       T     2     1     other     other
MS     F     20       U     GT3       T     4     2    health     other
GP     M     17       R     LE3       T     1     1     other  services
       M     17       U     LE3       T     4     4     other   teacher
       F     15       U     GT3       T     1     2   at_home     other
MS     M     17       U     LE3       T     3     1  services  services
GP     M     17       R     GT3       T     1     2   at_home     other
       F     15       U     GT3       A     3     3     other    health
       F     16       U     LE3       T     4     4    health    health
       M     16       U     GT3       T     3     2   at_home     other
       F     18       R     LE3       T     1     1   at_home     other
       M     15       U     GT3       T     4     4   teacher    health
       F     15       U     LE3       A     3     4     other     other
       F     17       U     LE3       T     0     2   at_home   at_home
MS     F     18       R     GT3       T     1     1     other     other

                reason guardian  …  famrel  freetime  goout Dalc Walc
school sex                       …
GP     F        course   mother  …       5         2      2    1    1  \
       F        course   father  …       4         2      2    1    2
MS     F        course   mother  …       5         4      2    1    2
GP     M          home   father  …       3         1      3    1    3
       F    reputation   father  …       4         1      2    1    1
       M        course   mother  …       4         2      3    1    2
MS     F        course    other  …       5         4      3    1    1
GP     M        course   mother  …       5         3      5    1    5
```

```
       M         home   father  …       4         1      1    2    2
       F       course   mother  …       4         3      2    1    1
MS     M       course   mother  …       2         4      5    3    4
GP     M         home   mother  …       3         1      3    1    5
       F   reputation   father  …       4         3      3    1    1
       F        other   mother  …       5         4      5    1    1
       M   reputation   mother  …       5         3      3    1    3
       F   reputation   mother  …       5         2      2    1    1
       M   reputation   mother  …       3         2      2    1    1
       F         home   mother  …       5         3      2    1    1
       F         home   father  …       3         3      3    2    3
MS     F         home   mother  …       4         3      2    1    2

           health absences  G1  G2  G3
school sex
GP     F        5        8   8   8   6
       F        5        2  13  13  13
MS     F        5        0   7   5   0
GP     M        5        4   8  10  10
       F        2        0   7  10  10
       M        5        0  15  15  15
MS     F        3        4  15  14  15
GP     M        5        0   5   8   7
       M        5        0  11  11  10
       F        5        2  10  11  11
MS     M        2        3  14  16  16
GP     M        3        4   8   9  10
       F        4       10  10  11  11
       F        4        4  14  15  16
       M        2       10  11   9  10
       F        3        1  12  12  12
       M        5        4  14  15  15
       F        1        0   7  10  11
       F        2        0  16  15  15
MS     F        4        2   8   8  10

[20 rows x 31 columns]
```