Asg 19.2 Pandas DataFrames (Coding)

Files needed for this assignment:

cars

Note: You also must provide the appropriate code and run all of your output to receive full credit for this assignment.

In [4]:

```
# set up notebook to display multiple output in one cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

print('The notebook is set up to display multiple output in one cell.')
```

```
The notebook is set up to display multiple output in one cell.
```

In [3]:

```
# conventional way to import pandas and numpy

import pandas as pd
import numpy as np
```

PART ONE

<div class="alert alert-block alert-info"

Use the student data that is found below for Questions 1 - 16:

ID, State, Grade, Gender, Age, Height, Weight

105, CO, 12, Female, 17, 64, 121

197, SC. 11, Female, 18, 66, 118

203, IN, 11, Male, 18, 70, 185

321, NJ, 12, Male, 18, 74, 167

476, WI, 11, Male, 17, 68, 158

</div>

Question 1:

Create a dictionary of lists from the data found above. Name the dictionary **student_dict**.

In [2]:

```
student_dict = {
    'ID': [105,197,203,321,476],
    'State':['CO','SC','IN','NJ','WI'],
    'Grade':[12,11,11,12,11],
    'Gender':['Female','Female','Male','Male','Male'],
    'Age':[17,18,18,18,17],
    'Height':[64,66,70,74,68],
    'Weight':[121,118,185,167,158]
}
student_dict
```

Out[2]:

```
{'ID': [105, 197, 203, 321, 476],
 'State': ['CO', 'SC', 'IN', 'NJ', 'WI'],
 'Grade': [12, 11, 11, 12, 11],
 'Gender': ['Female', 'Female', 'Male', 'Male', 'Male'],
 'Age': [17, 18, 18, 18, 17],
 'Height': [64, 66, 70, 74, 68],
 'Weight': [121, 118, 185, 167, 158]}
```

Question 2:

Use the dictionary from Part (a) to create a DataFrame. Name the DataFrame **student_df**.

In [5]:

```
student_df = pd.DataFrame(student_dict)
student_df
```

Out[5]:

	ID	State	Grade	Gender	Age	Height	Weight
0	105	CO	12	Female	17	64	121
1	197	SC	11	Female	18	66	118
2	203	IN	11	Male	18	70	185
3	321	NJ	12	Male	18	74	167
4	476	WI	11	Male	17	68	158

Question 3:

Use the appropriate DataFrame attribute to access the index (i.e. to access the row labels) for the DataFrame that you created in Question 2.

In [6]:
```
student_df.index
```

Out[6]:
```
RangeIndex(start=0, stop=5, step=1)
```

Question 4:

Use the appropriate DataFrame attribute to access the column labels for the DataFrame that you created in Question 2.

In [7]:
```
student_df.columns
```

Out[7]:
```
Index(['ID', 'State', 'Grade', 'Gender', 'Age', 'Height', 'Weight'], dtype='obje
ct')
```

Question 5:

Use the appropriate DataFrame attribute to access the data for the DataFrame that you created in Question 2.

In [8]:
```
student_df.values
```

Out[8]:
```
array([[105, 'CO', 12, 'Female', 17, 64, 121],
       [197, 'SC', 11, 'Female', 18, 66, 118],
       [203, 'IN', 11, 'Male', 18, 70, 185],
       [321, 'NJ', 12, 'Male', 18, 74, 167],
       [476, 'WI', 11, 'Male', 17, 68, 158]], dtype=object)
```

Question 6:

Redo Question 2, but this time use the strings 'One', 'Two', 'Three', 'Four', 'Five' for the index.

In [10]:

```
arr = np.array(['One','Two','Three','Four','Five'])
student_df = pd.DataFrame(student_dict, index = arr)
student_df
```

Out[10]:

	ID	State	Grade	Gender	Age	Height	Weight
One	105	CO	12	Female	17	64	121
Two	197	SC	11	Female	18	66	118
Three	203	IN	11	Male	18	70	185
Four	321	NJ	12	Male	18	74	167
Five	476	WI	11	Male	17	68	158

Question 7:

Redo Question 2, but this time use a NumPy array to set the index to [01, 02, 11, 12, 13].

In [11]:

```
arr = np.array(['01','02','11','12','13'])
student_df = pd.DataFrame(student_dict, index = arr)
student_df
```

Out[11]:

	ID	State	Grade	Gender	Age	Height	Weight
01	105	CO	12	Female	17	64	121
02	197	SC	11	Female	18	66	118
11	203	IN	11	Male	18	70	185
12	321	NJ	12	Male	18	74	167
13	476	WI	11	Male	17	68	158

Question 8:

Redo Question 2, and then on the resulting DataFrame reset the index to be the ID column.

In [22]:

```python
student_df = pd.DataFrame(student_dict)
student_df.set_index("ID",inplace=True)
student_df
```

Out[22]:

	State	Grade	Gender	Age	Height	Weight
ID						
105	CO	12	Female	17	64	121
197	SC	11	Female	18	66	118
203	IN	11	Male	18	70	185
321	NJ	12	Male	18	74	167
476	WI	11	Male	17	68	158

Question 9:

Redo Question 2, and then on the resulting DataFrame, reset the index to be a multi-index using the ID and Gender columns.

In [23]:

```python
student_df = pd.DataFrame(student_dict)
student_df.set_index(["ID","Gender"],inplace=True)
student_df
```

Out[23]:

		State	Grade	Age	Height	Weight
ID	**Gender**					
105	**Female**	CO	12	17	64	121
197	**Female**	SC	11	18	66	118
203	**Male**	IN	11	18	70	185
321	**Male**	NJ	12	18	74	167
476	**Male**	WI	11	17	68	158

Question 10:

Redo Question 2,, but only include the columns ID, Grade, and Height in your DataFrame.

In [24]:

```python
student_df = pd.DataFrame(student_dict,columns=['ID','Grade','Height'])
student_df
```

Out[24]:

	ID	Grade	Height
0	105	12	64
1	197	11	66
2	203	11	70
3	321	12	74
4	476	11	68

Question 11:

Redo Question 2, but put the columns in alphabetical order.

In [41]:

```
student_df = pd.DataFrame(student_dict)
student_df = student_df.sort_values('State', ascending=True)
student_df
```

Out[41]:

	ID	State	Grade	Gender	Age	Height	Weight
0	105	CO	12	Female	17	64	121
2	203	IN	11	Male	18	70	185
3	321	NJ	12	Male	18	74	167
1	197	SC	11	Female	18	66	118
4	476	WI	11	Male	17	68	158

Question 12:

Redo Question 2, then use bracket notation to find the data type for the ID, State, and Grade columns.

In [48]:

```
student_df = pd.DataFrame(student_dict)
print(student_df)
print(f"Data type for ID: {student_df['ID'].dtype}")
print(f"Data type for State: {student_df['State'].dtype}")
print(f"Data type for Grade: {student_df['Grade'].dtype}")
```

```
    ID State  Grade  Gender  Age  Height  Weight
0  105    CO     12  Female   17      64     121
1  197    SC     11  Female   18      66     118
2  203    IN     11    Male   18      70     185
3  321    NJ     12    Male   18      74     167
4  476    WI     11    Male   17      68     158
Data type for ID: int64
Data type for State: object
Data type for Grade: int64
```

Question 13:

Redo Question 2, then use dot notation to find the data type for the Gender, Age, and Height columns.

In [49]:

```python
student_df = pd.DataFrame(student_dict)
print(student_df)
print(f"Data type for ID: {student_df.ID.dtype}")
print(f"Data type for State: {student_df.State.dtype}")
print(f"Data type for Grade: {student_df.Grade.dtype}")
```

```
    ID State  Grade  Gender  Age  Height  Weight
0  105    CO     12  Female   17      64     121
1  197    SC     11  Female   18      66     118
2  203    IN     11    Male   18      70     185
3  321    NJ     12    Male   18      74     167
4  476    WI     11    Male   17      68     158
Data type for ID: int64
Data type for State: object
Data type for Grade: int64
```

Question 14:

Redo Question 2, then find the data types for all of the columns using a single command.

In [51]:

```python
student_df.dtypes
```

Out[51]:

```
ID           int64
State       object
Grade        int64
Gender      object
Age          int64
Height       int64
Weight       int64
dtype: object
```

Question 15:

Redo Question 2, and then find the mean Height and Weight.

In [54]:

```
print(f"Student Mean Height: {student_df.Height.mean()}")
print(f"Student Mean Weight: {student_df.Weight.mean()}")
```

```
Student Mean Height: 68.4
Student Mean Weight: 149.8
```

Question 16:

Redo Question 2, and then find the unique elements from the Age column.

In [58]:

```
print(f"Unique elements from the Age column: {student_df.Age.un
```

```
Unique elements from the Age column: [17 18]
```

PART TWO

<div class="alert alert-block alert-info"

For Questions 17 - 26: Read in the following dataset. Name the DataFrame that you create **cars**:

cars </div>

In [115…

```
cars_df = pd.read_csv('cars.csv')
cars_df
```

Out[115]:

	Manufacturer	Year	Fuel	Transmission	Price
0	Acura	2012	Gas	Automatic	10299
1	Jaguar	2011	Gas	Automatic	9500
2	Honda	2004	Gas	Automatic	3995
3	Chevrolet	2016	Gas	Automatic	41988
4	Kia	2015	Gas	Automatic	12995
...	...	...	...	...	...
460461	Rover	2008	Gas	Automatic	7950
460462	Nissan	2016	Gas	Automatic	13995
460463	BMW	2010	Gas	Automatic	10995
460464	Dodge	2015	Other	Manual	6495
460465	GMC	2008	Gas	Automatic	8990

460466 rows × 5 columns

Question 17:

Inspect the **cars** DataFrame by viewing the first 5 rows of the DataFrame.

In [116…

```
cars_df.head(5)
```

Out[116]:

	Manufacturer	Year	Fuel	Transmission	Price
0	Acura	2012	Gas	Automatic	10299
1	Jaguar	2011	Gas	Automatic	9500
2	Honda	2004	Gas	Automatic	3995
3	Chevrolet	2016	Gas	Automatic	41988
4	Kia	2015	Gas	Automatic	12995

Question 18:

Inspect the cars DataFrame by viewing the first 7 rows of the DataFrame.

In [117…

```
cars_df.head(7)
```

Out[117]:

	Manufacturer	Year	Fuel	Transmission	Price
0	Acura	2012	Gas	Automatic	10299
1	Jaguar	2011	Gas	Automatic	9500
2	Honda	2004	Gas	Automatic	3995
3	Chevrolet	2016	Gas	Automatic	41988
4	Kia	2015	Gas	Automatic	12995
5	Chevrolet	2014	Gas	Automatic	10995
6	BMW	2011	Gas	Automatic	8995

Question 19:

Inspect the cars DataFrame by viewing the first 5 rows of the Year column.

In [120…

```
cars_df['Year'].head(5)
```

Out[120]:

```
0    2012
1    2011
2    2004
3    2016
4    2015
Name: Year, dtype: int64
```

Question 20:

Inspect the cars DataFrame by viewing the first 4 rows of the Manufacturer and Transmission columns.

In [121…

```
cars_df[['Manufacturer','Transmission']].head(4)
```

Out[121]:

	Manufacturer	Transmission
0	Acura	Automatic
1	Jaguar	Automatic
2	Honda	Automatic
3	Chevrolet	Automatic

Question 21:

Inspect the cars DataFrame by viewing the last 5 rows of the DataFrame.

In [122…

```
cars_df.tail(5)
```

Out[122]:

	Manufacturer	Year	Fuel	Transmission	Price
460461	Rover	2008	Gas	Automatic	7950
460462	Nissan	2016	Gas	Automatic	13995
460463	BMW	2010	Gas	Automatic	10995
460464	Dodge	2015	Other	Manual	6495
460465	GMC	2008	Gas	Automatic	8990

Question 22:

Inspect the cars DataFrame by viewing the last 8 rows of the DataFrame.

In [123…

```
cars_df.tail(8)
```

Out[123]:

	Manufacturer	Year	Fuel	Transmission	Price
460458	GMC	2013	Gas	Automatic	9995
460459	Audi	2006	Gas	Automatic	5295
460460	Mazda	2015	Gas	Automatic	12955
460461	Rover	2008	Gas	Automatic	7950
460462	Nissan	2016	Gas	Automatic	13995
460463	BMW	2010	Gas	Automatic	10995
460464	Dodge	2015	Other	Manual	6495
460465	GMC	2008	Gas	Automatic	8990

Question 23:

Use the appropriate DataFrame attribute to find the number of rows and columns in the DataFrame.

In [124…

```
rows, cols = cars_df.shape
print(f"Number of rows: {rows}")
print(f"Number of columns: {cols}")
```

```
Number of rows: 460466
Number of columns: 5
```

Question 24:

Find the number of non-null values in the Transmission column.

In [125…

```
cars_df.Transmission.notnull().sum()
```

Out[125]:

```
457183
```

Question 25:

Find the unique elements from the Manufacturer column.

In [126…

```
cars_df.Manufacturer.unique()
```

Out[126]:

```
array(['Acura', 'Jaguar', 'Honda', 'Chevrolet', 'K
ia', 'BMW', 'Toyota',
       'Nissan', 'Volkswagen', 'Ford', 'GMC', 'Sub
aru', 'Ram', 'Lexus',
       'Volvo', 'Buick', 'Jeep', 'Hyundai', 'Merce
des-Benz', 'Cadillac',
       'Audi', 'Infiniti', 'Dodge', 'Pontiac', 'Mi
ni', 'Chrysler',
       'Mazda', 'Mercury', nan, 'Fiat', 'Harley-Da
vidson', 'Saturn',
       'Mitsubishi', 'Lincoln', 'Rover', 'Tesla',
'Alfa-Romeo',
       'Aston-Martin', 'Ferrari', 'Land Rover', 'P
orche', 'Hennessey'],
      dtype=object)
```

Question 26:

Find the maximum and minimum price for these used cars.

In [127…

```
print(f"Maximum price: {cars_df['Price'].max()}")
print(f"Minimum price: {cars_df['Price'].min()}")
```

```
Maximum price: 4294967295
Minimum price: 1
```

PART THREE

Question 27:

Create a 3 x 7 NumPy array of random integers from 60 to 95.

In [78]:

```
arr = np.random.randint(65,96,(3,7))
print(arr)
```

```
[[70 90 91 68 72 88 79]
 [73 80 92 90 65 88 76]
 [87 79 70 88 69 79 69]]
```

Question 28:

Use the NumPy array that you created in Question 27 to create a DataFrame named **august_temps**.

In [80]:

```
august_temps = pd.DataFrame(arr)
print(august_temps)
```

```
    0   1   2   3   4   5   6
0  70  90  91  68  72  88  79
1  73  80  92  90  65  88  76
2  87  79  70  88  69  79  69
```

Question 29:

Manually set the column labels of the DataFrame that you created to the days of the week -- starting with Sunday and ending with Saturday; and manually set the row labels (i.e. the index) to 2019, 2020, and 2021.

In [86]:

```
august_temps.columns = ['Sunday','Monday','Tu
august_temps.index = [2019,2020,2021]
august_temps
```

Out[86]:

	Sunday	Monday	Tuesday	Wednesday	Thurs
2019	70	90	91	68	
2020	73	80	92	90	
2021	87	79	70	88	

PART FOUR

Question 30:

Create a 4 x 5 NumPy array of random numbers from a Normal distribution with a mean of 88 and a standard deviation of 8.2.

In [89]:

```
arr = np.random.normal(88,8.2,20).reshape(4,
arr
```

Out[89]:

```
array([[92.37920766, 99.0171765 , 75.084637
53, 93.73118182, 87.26794386],
       [99.72011568, 84.96436084, 84.814237
04, 88.3978203 , 90.68816952],
       [85.48257776, 75.28405122, 93.217894
98, 98.3358218 , 78.28140844],
       [72.62892292, 99.59683547, 75.440138
75, 80.40969655, 88.07189454]])
```

Question 31:

Use the NumPy array that you created in Question 30 to create a DataFrame named test_scores.

In [91]:

```
test_scores = pd.DataFrame(arr)
test_scores
```

Out[91]:

	0	1	2	3	
0	92.379208	99.017176	75.084638	93.731182	87.
1	99.720116	84.964361	84.814237	88.397820	90.
2	85.482578	75.284051	93.217895	98.335822	78.
3	72.628923	99.596835	75.440139	80.409697	88.

Question 32:

Manually set the column labels of the DataFrame that you created to Test_1, Test_2, Test_3, Test_4, and Test_5; and manually set the row labels (i.e. the index) to Student_1, Student_2, Student_3, and Student_4.

In [92]:

```
test_scores.columns = ['Test_1','Test_2',
test_scores.index = ['Student_1','Student
test_scores
```

Out[92]:

	Test_1	Test_2	Test_3	
Student_1	92.379208	99.017176	75.084638	93.7
Student_2	99.720116	84.964361	84.814237	88.3
Student_3	85.482578	75.284051	93.217895	98.3
Student_4	72.628923	99.596835	75.440139	80.4

In [ ]: