Asg 19.6

Working with Pandas DataFrames -- The Essentials (Part Four)

(Coding)

Files needed for this assignment:

marketing_campaign.csv

baseball_data.csv

student.txt

cereals.txt

```
In [1]: # 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 [2]: # conventional way to import pandas and numpy

        import pandas as pd
        import numpy as np
```

PART ONE

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

For Questions 1-9: We will be using the **'marketing_campaign.csv' dataset** and the **customers DataFrame** </div>

Question 1:

a. Read in the dataset **'marketing_campaign.csv'** and store the results in a DataFrame named **customers**.

b. Use appropriate attributes and methods to inspect the **customers** DataFrame.

In [4]:
```
customers = pd.read_csv('marketing_campaign.csv', sep=';')
print(customers.head())
print(customers.tail())
print(customers.info())
```

```
     ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0  5524        1957  Graduation         Single  58138.0        0         0
1  2174        1954  Graduation         Single  46344.0        1         1
2  4141        1965  Graduation       Together  71613.0        0         0
3  6182        1984  Graduation       Together  26646.0        1         0
4  5324        1981         PhD        Married  58293.0        1         0

  Dt_Customer  Recency  MntWines  ...  NumWebVisitsMonth  AcceptedCmp3  \
0  2012-09-04       58       635  ...                  7             0
1  2014-03-08       38        11  ...                  5             0
2  2013-08-21       26       426  ...                  4             0
3  2014-02-10       26        11  ...                  6             0
4  2014-01-19       94       173  ...                  5             0

   AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  \
0             0             0             0             0         0
1             0             0             0             0         0
2             0             0             0             0         0
3             0             0             0             0         0
4             0             0             0             0         0

   Z_CostContact  Z_Revenue  Response
0              3         11         1
1              3         11         0
2              3         11         0
3              3         11         0
4              3         11         0

[5 rows x 29 columns]
         ID  Year_Birth   Education Marital_Status   Income  Kidhome  \
2235  10870        1967  Graduation        Married  61223.0        0
2236   4001        1946         PhD       Together  64014.0        2
2237   7270        1981  Graduation       Divorced  56981.0        0
2238   8235        1956      Master       Together  69245.0        0
2239   9405        1954         PhD        Married  52869.0        1

      Teenhome Dt_Customer  Recency  MntWines  ...  NumWebVisitsMonth  \
2235         1  2013-06-13       46       709  ...                  5
2236         1  2014-06-10       56       406  ...                  7
2237         0  2014-01-25       91       908  ...                  6
2238         1  2014-01-24        8       428  ...                  3
2239         1  2012-10-15       40        84  ...                  7

      AcceptedCmp3  AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  \
2235             0             0             0             0             0
2236             0             0             0             1             0
2237             0             1             0             0             0
2238             0             0             0             0             0
2239             0             0             0             0             0

      Complain  Z_CostContact  Z_Revenue  Response
2235         0              3         11         0
2236         0              3         11         0
2237         0              3         11         0
2238         0              3         11         0
2239         0              3         11         1

[5 rows x 29 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
```

```
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   2240 non-null   int64  
 1   Year_Birth           2240 non-null   int64  
 2   Education            2240 non-null   object 
 3   Marital_Status       2240 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64  
 6   Teenhome             2240 non-null   int64  
 7   Dt_Customer          2240 non-null   object 
 8   Recency              2240 non-null   int64  
 9   MntWines             2240 non-null   int64  
 10  MntFruits            2240 non-null   int64  
 11  MntMeatProducts      2240 non-null   int64  
 12  MntFishProducts      2240 non-null   int64  
 13  MntSweetProducts     2240 non-null   int64  
 14  MntGoldProds         2240 non-null   int64  
 15  NumDealsPurchases    2240 non-null   int64  
 16  NumWebPurchases      2240 non-null   int64  
 17  NumCatalogPurchases  2240 non-null   int64  
 18  NumStorePurchases    2240 non-null   int64  
 19  NumWebVisitsMonth    2240 non-null   int64  
 20  AcceptedCmp3         2240 non-null   int64  
 21  AcceptedCmp4         2240 non-null   int64  
 22  AcceptedCmp5         2240 non-null   int64  
 23  AcceptedCmp1         2240 non-null   int64  
 24  AcceptedCmp2         2240 non-null   int64  
 25  Complain             2240 non-null   int64  
 26  Z_CostContact        2240 non-null   int64  
 27  Z_Revenue            2240 non-null   int64  
 28  Response             2240 non-null   int64  
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
None
```

Question 2:

Filter the **customers** DataFrame rows to show only customers with an **'Income'** of at least 100000.

In [6]:
```
customers.loc[customers['Income']>=1000,:]
```

Out[6]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	R
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	
...	...	...	...	...	...	...	...	...	
2235	10870	1967	Graduation	Married	61223.0	0	1	2013-06-13	
2236	4001	1946	PhD	Together	64014.0	2	1	2014-06-10	
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014-01-25	
2238	8235	1956	Master	Together	69245.0	0	1	2014-01-24	
2239	9405	1954	PhD	Married	52869.0	1	1	2012-10-15	

2216 rows × 29 columns

Question 3:
Select the **'Education'** Series from the Question 2 filtered DataFrame using the .loc label-based indexing property.

Note: See the image below for information on selecting data via the Pandas .loc (label-based indexing) and .iloc (position-based indexing) indexing properties: ![loc%20and%20iloc.jpg](attachment:loc%20and%20iloc.jpg)

Documentation for **'loc'**
Documentation for **'iloc'**

In [10]:
```
customers.loc[:,'Education']
```

Out[10]:
```
0       Graduation
1       Graduation
2       Graduation
3       Graduation
4              PhD
           ...
2235    Graduation
2236           PhD
2237    Graduation
2238        Master
2239           PhD
Name: Education, Length: 2240, dtype: object
```

Question 4:

Select the **'Marital_Status'** Series from the Question 2 filtered DataFrame without using the .loc label-based indexing attribute.

In [11]:
```
customers['Marital_Status']
```

Out[11]:
```
0         Single
1         Single
2       Together
3       Together
4        Married
          ...   
2235     Married
2236    Together
2237    Divorced
2238    Together
2239     Married
Name: Marital_Status, Length: 2240, dtype: object
```

Question 5:

Filter the **customers** DataFrame rows to show only customers with an **'Income'** of at least 80000 and a **'Marital_Status'** of Single.

In [21]:
```
customers.loc[(customers['Income']>=80000) & (customers['Marital_Status']=='Single
```

Out[21]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custome
15	2114	1946	PhD	Single	82800.0	0	0	2012-11-2
67	9369	1979	2n Cycle	Single	88194.0	0	1	2014-03-1
113	999	1991	Graduation	Single	86037.0	0	0	2013-01-0
124	7215	1983	Graduation	Single	101970.0	0	0	2013-03-1
140	821	1992	Master	Single	92859.0	0	0	2012-10-1
159	2730	1955	Graduation	Single	80317.0	0	0	2013-08-2
418	4216	1981	Graduation	Single	91065.0	0	0	2013-02-2
430	3725	1961	PhD	Single	84865.0	0	0	2013-05-0
447	1137	1964	Graduation	Single	81246.0	0	0	2013-12-2
456	4947	1966	2n Cycle	Single	89572.0	0	0	2012-09-1
507	6071	1989	Graduation	Single	81217.0	0	0	2013-07-1
515	203	1975	Master	Single	81169.0	0	0	2013-04-1
561	3179	1980	Graduation	Single	81741.0	0	0	2013-06-1
626	10156	1975	Graduation	Single	84196.0	0	1	2013-06-0
634	8923	1973	Graduation	Single	83917.0	0	0	2013-04-1
636	6945	1952	Graduation	Single	84574.0	0	0	2013-06-0
650	4248	1960	Master	Single	98777.0	0	0	2014-02-1
686	9826	1972	PhD	Single	86857.0	0	0	2012-09-1
703	8029	1988	Master	Single	90247.0	0	0	2014-04-2
803	9930	1944	PhD	Single	82716.0	0	0	2013-11-0
878	1446	1956	Master	Single	86424.0	0	0	2014-04-0
884	5830	1972	PhD	Single	86857.0	0	0	2012-09-1
905	11074	1977	Graduation	Single	85072.0	0	0	2014-04-0
906	10150	1961	Graduation	Single	86429.0	0	0	2013-11-2
914	10619	1994	Graduation	Single	95529.0	0	0	2012-12-0
1001	7962	1987	PhD	Single	95169.0	0	0	2013-10-0
1031	9220	1971	Graduation	Single	91700.0	0	0	2013-01-1
1066	3005	1992	Graduation	Single	83528.0	0	0	2014-05-0
1097	10245	1986	2n Cycle	Single	80910.0	0	0	2012-10-3
1111	1524	1983	2n Cycle	Single	81698.0	0	0	2013-03-0
1113	7451	1960	Master	Single	98777.0	0	0	2014-02-1
1126	6749	1966	Graduation	Single	86358.0	1	1	2012-08-0
1169	2410	1969	Graduation	Single	81657.0	0	0	2014-01-2
1179	5735	1991	Master	Single	90638.0	0	0	2014-02-1
1269	9400	1958	2n Cycle	Single	85485.0	0	0	2014-06-2

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custome
1333	5147	1948	Graduation	Single	90842.0	0	0	2013-07-2
1358	1065	1963	Graduation	Single	80695.0	0	0	2013-12-2
1458	10133	1970	Graduation	Single	93790.0	0	0	2014-02-1
1520	4278	1983	PhD	Single	87188.0	0	0	2013-06-0
1570	4261	1946	PhD	Single	82800.0	0	0	2012-11-2
1572	5350	1991	Master	Single	90638.0	0	0	2014-02-1
1651	8395	1961	Graduation	Single	82014.0	0	0	2012-08-2
1689	295	1989	Graduation	Single	81217.0	0	0	2013-07-1
1722	569	1991	Graduation	Single	90273.0	0	0	2013-12-1
1850	4427	1995	2n Cycle	Single	83257.0	0	0	2012-09-1
1854	10163	1984	PhD	Single	82733.0	0	0	2013-09-1
1897	5558	1954	PhD	Single	90933.0	0	0	2014-03-3
1898	4619	1945	PhD	Single	113734.0	0	0	2014-05-2
1922	3138	1956	Graduation	Single	91249.0	0	0	2012-10-2
1924	7966	1959	Graduation	Single	80982.0	1	1	2013-01-0
1952	3434	1951	Graduation	Single	80872.0	0	0	2014-05-1
1958	2109	1990	Graduation	Single	96843.0	0	0	2013-04-2
1992	6248	1947	Master	Single	91712.0	0	0	2013-10-1
1993	10164	1958	Graduation	Single	94472.0	0	1	2014-04-0
2041	4974	1970	Graduation	Single	83273.0	1	2	2012-09-2
2109	3104	1961	Graduation	Single	82332.0	0	0	2012-09-1
2167	3520	1990	Master	Single	91172.0	0	0	2013-03-2
2190	4418	1983	Master	Single	89616.0	0	0	2013-02-2
2213	3661	1995	2n Cycle	Single	80617.0	0	0	2012-10-1

59 rows × 29 columns

Question 6:

Filter the **customers** DataFrame rows to show only customers with an **'Income'** greater than 80000, a **'Marital_Status'** of Divorced, and an '**'Education'** level of Graduation.

In [20]:
```python
customers.loc[(customers['Income']>=80000) & (customers['Marital_Status']=='Divo
```

Out[20]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custom
53	2225	1977	Graduation	Divorced	82582.0	0	0	2014-06-
84	535	1987	Graduation	Divorced	81361.0	0	0	2014-02-
252	10089	1974	Graduation	Divorced	102692.0	0	0	2013-04-
345	1411	1952	Graduation	Divorced	82623.0	0	0	2013-11-
347	1826	1970	Graduation	Divorced	84835.0	0	0	2014-06-
655	5555	1975	Graduation	Divorced	153924.0	0	0	2014-02-
734	10430	1973	Graduation	Divorced	89694.0	1	1	2013-10-
837	5687	1980	Graduation	Divorced	81702.0	0	0	2012-09-
942	6810	1983	Graduation	Divorced	82025.0	0	0	2013-05-
990	8545	1954	Graduation	Divorced	85683.0	0	0	2014-03-
1100	5538	1975	Graduation	Divorced	83829.0	0	0	2013-10-
1265	3910	1975	Graduation	Divorced	83829.0	0	0	2013-10-
1582	3503	1950	Graduation	Divorced	82460.0	0	0	2013-12-
1584	4608	1987	Graduation	Divorced	81361.0	0	0	2014-02-

14 rows × 29 columns

Question 7:

Filter the **customers** DataFrame rows to show only customers with a **'Marital_Status'** of Married or Together or an ''**Education**' level of PhD.

In [22]:
```
customers.loc[(customers['Education']=='PhD') | (customers['Marital_Status']=='
```

Out[22]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custo
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02
4	5324	1981	PhD	Married	58293.0	1	0	2014-01
5	7446	1967	Master	Together	62513.0	0	1	2013-09
7	6177	1985	PhD	Married	33454.0	1	0	2013-05
...	...	...	...	...	...	...	...	
2234	8372	1974	Graduation	Married	34421.0	1	0	2013-07
2235	10870	1967	Graduation	Married	61223.0	0	1	2013-06
2236	4001	1946	PhD	Together	64014.0	2	1	2014-06
2238	8235	1956	Master	Together	69245.0	0	1	2014-01
2239	9405	1954	PhD	Married	52869.0	1	1	2012-10

1621 rows × 29 columns

Question 8:

Filter the **customers** DataFrame rows to show only customers with an **Income** between 40000 and 60000 or an **''Education'** level of Master or PhD.

In [24]:
```
customers.loc[((customers['Income']>=40000) & (customers['Income']<=60000)) |
```

Out[24]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custo
0	5524	1957	Graduation	Single	58138.0	0	0	2012-0
1	2174	1954	Graduation	Single	46344.0	1	1	2014-0
4	5324	1981	PhD	Married	58293.0	1	0	2014-0
5	7446	1967	Master	Together	62513.0	0	1	2013-0
6	965	1971	Graduation	Divorced	55635.0	0	1	2012-1
...	...	...	...	...	...	...	...	
2227	4201	1962	Graduation	Single	57967.0	0	1	2013-0
2231	9817	1970	Master	Single	44802.0	0	0	2012-0
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014-0
2238	8235	1956	Master	Together	69245.0	0	1	2014-0
2239	9405	1954	PhD	Married	52869.0	1	1	2012-1

888 rows × 29 columns

Question 9:

Use the **"isin() method"** to filter the **customers** DataFrame to show only customers with a **Marital_Status** of Single, Married, or Together.

Documentation for **'isin'**

In [31]:
```
specific_statuses = customers['Marital_Status'].isin(['Single','Married','To
customers[specific_statuses]
```

Out[31]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cu
0	5524	1957	Graduation	Single	58138.0	0	0	201
1	2174	1954	Graduation	Single	46344.0	1	1	201
2	4141	1965	Graduation	Together	71613.0	0	0	201
3	6182	1984	Graduation	Together	26646.0	1	0	201
4	5324	1981	PhD	Married	58293.0	1	0	201
...	...	...	...	...	...	...	...	
2234	8372	1974	Graduation	Married	34421.0	1	0	201
2235	10870	1967	Graduation	Married	61223.0	0	1	201
2236	4001	1946	PhD	Together	64014.0	2	1	201
2238	8235	1956	Master	Together	69245.0	0	1	201
2239	9405	1954	PhD	Married	52869.0	1	1	201

1924 rows × 29 columns

PART TWO

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

For Questions 10-14: We will be using the **'baseball_data.csv' dataset** and the **baseball DataFrame** </div>

Question 10:

a. Read in the dataset **'baseball_data.csv'** and store the results in a DataFrame named **baseball**.

See the link below for access to the dataset.

baseball_data.csv

b. Use appropriate attributes and methods to inspect the **baseball** DataFrame.

In [34]:
```
baseball = pd.read_csv('baseball_data.csv')
print(baseball.head())
print(baseball.tail())
print(baseball.info())
```

```
    playerID  yearID  stint teamID lgID   G   AB   R   H  2B  ...   RBI
SB  \
0  abercda01    1871      1    TRO  NaN   1    4   0   0   0  ...   0.0
0.0
1   addybo01    1871      1    RC1  NaN  25  118  30  32   6  ...  13.0
8.0
2  allisar01    1871      1    CL1  NaN  29  137  28  40   4  ...  19.0
3.0
3  allisdo01    1871      1    WS3  NaN  27  133  28  44  10  ...  27.0
1.0
4  ansonca01    1871      1    RC1  NaN  25  120  29  39  11  ...  16.0
6.0

    CS  BB   SO  IBB  HBP  SH  SF  GIDP
0  0.0   0  0.0  NaN  NaN NaN NaN   0.0
1  1.0   4  0.0  NaN  NaN NaN NaN   0.0
2  1.0   2  5.0  NaN  NaN NaN NaN   1.0
3  1.0   0  2.0  NaN  NaN NaN NaN   0.0
4  2.0   2  1.0  NaN  NaN NaN NaN   0.0

[5 rows x 22 columns]
         playerID  yearID  stint teamID lgID   G  AB  R   H  2B  ...   RBI
\
108784  zimmebr02    2020      1    BAL   AL   2   0  0   0   0  ...   0.0
108785  zimmejo02    2020      1    DET   AL   3   0  0   0   0  ...   0.0
108786  zimmeky01    2020      1    KCA   AL  16   0  0   0   0  ...   0.0
108787  zuberty01    2020      1    KCA   AL  23   0  0   0   0  ...   0.0
108788  zuninmi01    2020      1    TBA   AL  28  75  8  11   4  ...  10.0

         SB   CS  BB    SO  IBB  HBP   SH   SF  GIDP
108784  0.0  0.0   0   0.0  0.0  0.0  0.0  0.0   0.0
108785  0.0  0.0   0   0.0  0.0  0.0  0.0  0.0   0.0
108786  0.0  0.0   0   0.0  0.0  0.0  0.0  0.0   0.0
108787  0.0  0.0   0   0.0  0.0  0.0  0.0  0.0   0.0
108788  0.0  0.0   6  37.0  0.0  3.0  0.0  0.0   0.0

[5 rows x 22 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108789 entries, 0 to 108788
Data columns (total 22 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   playerID  108789 non-null  object
 1   yearID    108789 non-null  int64
 2   stint     108789 non-null  int64
 3   teamID    108789 non-null  object
 4   lgID      108052 non-null  object
 5   G         108789 non-null  int64
 6   AB        108789 non-null  int64
 7   R         108789 non-null  int64
 8   H         108789 non-null  int64
 9   2B        108789 non-null  int64
 10  3B        108789 non-null  int64
 11  HR        108789 non-null  int64
 12  RBI       108033 non-null  float64
 13  SB        106421 non-null  float64
 14  CS        85248 non-null   float64
 15  BB        108789 non-null  int64
 16  SO        106689 non-null  float64
 17  IBB       72139 non-null   float64
 18  HBP       105973 non-null  float64
 19  SH        102721 non-null  float64
 20  SF        72686 non-null   float64
 21  GIDP      83348 non-null   float64
dtypes: float64(9), int64(10), object(3)
memory usage: 18.3+ MB
None
```

Question 11:

a. Read in the dataset **'baseball_data.csv'** again, but this time specify by **name** that the only columns you want to include are the 'R', 'H', 'HR', and 'RBI' columns. Store the results in a DataFrame named **baseball_2**.

b. Use appropriate attributes or methods to check that you read in the correct columns.

In [37]:
```python
baseball_2 = pd.read_csv('baseball_data.csv',usecols=['R','H','HR','RBI']
baseball_2
```

Out[37]:

	R	H	HR	RBI
0	0	0	0	0.0
1	30	32	0	13.0
2	28	40	0	19.0
3	28	44	2	27.0
4	29	39	0	16.0
...	...	...	...	...
108784	0	0	0	0.0
108785	0	0	0	0.0
108786	0	0	0	0.0
108787	0	0	0	0.0
108788	8	11	4	10.0

108789 rows × 4 columns

Question 12:

a. Read in the dataset **'baseball_data.csv'** again, but this time specify by **position** that the only columns you want to include are the 'R', 'H', 'HR', and 'RBI' columns. Store the results in a DataFrame named **baseball_3**.

b. Use appropriate attributes or methods to check that you read in the correct columns.

```
In [42]: baseball_3 = pd.read_csv('baseball_data.csv',usecols=[7,8,11,12])
         baseball_3
```

Out[42]:

	R	H	HR	RBI
0	0	0	0	0.0
1	30	32	0	13.0
2	28	40	0	19.0
3	28	44	2	27.0
4	29	39	0	16.0
...	...	...	...	...
108784	0	0	0	0.0
108785	0	0	0	0.0
108786	0	0	0	0.0
108787	0	0	0	0.0
108788	8	11	4	10.0

108789 rows × 4 columns

Question 13:

a. Read in the dataset **'baseball_data.csv'** again, but this time specify that you only want to read in the first 7 rows. Store the results in a DataFrame named **baseball_4**.

b. Use appropriate attributes or methods to check that you read in the correct rows.

```
In [44]: baseball_4 = pd.read_csv('baseball_data.csv',nrows=7)
         baseball_4
```

Out[44]:

	playerID	yearID	stint	teamID	lgID	G	AB	R	H	2B	...	RBI	SB	CS
0	abercda01	1871	1	TRO	NaN	1	4	0	0	0	...	0	0	0
1	addybo01	1871	1	RC1	NaN	25	118	30	32	6	...	13	8	1
2	allisar01	1871	1	CL1	NaN	29	137	28	40	4	...	19	3	1
3	allisdo01	1871	1	WS3	NaN	27	133	28	44	10	...	27	1	1
4	ansonca01	1871	1	RC1	NaN	25	120	29	39	11	...	16	6	2
5	armstbo01	1871	1	FW1	NaN	12	49	9	11	2	...	5	0	1
6	barkeal01	1871	1	RC1	NaN	1	4	0	1	0	...	2	0	0

7 rows × 22 columns

Question 13:

Print out the teamID column by iterating through the **'teamID'** Series of the **baseball_4** DataFrame.

In [45]:
```
baseball_4['teamID']
```

Out[45]:
```
0    TRO
1    RC1
2    CL1
3    WS3
4    RC1
5    FW1
6    RC1
Name: teamID, dtype: object
```

Question 14:

Use the **iterrows() method** to print out each row of the **baseball_4** DataFrame as a Series.

In [52]:
```
gen = (baseball_4.iterrows())
for i in range(len(baseball_4)):
    print(next(gen)[1])
```

```
playerID    abercda01
yearID           1871
stint               1
teamID            TRO
lgID              NaN
G                   1
AB                  4
R                   0
H                   0
2B                  0
3B                  0
HR                  0
RBI                 0
SB                  0
CS                  0
BB                  0
SO                  0
IBB               NaN
HBP               NaN
SH                NaN
SF                NaN
GIDP                0
Name: 0, dtype: object
playerID    addybo01
yearID          1871
stint              1
teamID           RC1
lgID             NaN
G                 25
AB               118
R                 30
H                 32
2B                 6
3B                 0
HR                 0
RBI               13
SB                 8
CS                 1
BB                 4
SO                 0
IBB              NaN
HBP              NaN
SH               NaN
SF               NaN
GIDP               0
Name: 1, dtype: object
playerID    allisar01
yearID           1871
stint               1
teamID            CL1
lgID              NaN
G                  29
AB                137
R                  28
H                  40
2B                  4
3B                  5
HR                  0
RBI                19
SB                  3
```

```
CS                  1
BB                  2
SO                  5
IBB               NaN
HBP               NaN
SH                NaN
SF                NaN
GIDP                1
Name: 2, dtype: object
playerID    allisdo01
yearID           1871
stint               1
teamID            WS3
lgID              NaN
G                  27
AB                133
R                  28
H                  44
2B                 10
3B                  2
HR                  2
RBI                27
SB                  1
CS                  1
BB                  0
SO                  2
IBB               NaN
HBP               NaN
SH                NaN
SF                NaN
GIDP                0
Name: 3, dtype: object
playerID    ansonca01
yearID           1871
stint               1
teamID            RC1
lgID              NaN
G                  25
AB                120
R                  29
H                  39
2B                 11
3B                  3
HR                  0
RBI                16
SB                  6
CS                  2
BB                  2
SO                  1
IBB               NaN
HBP               NaN
SH                NaN
SF                NaN
GIDP                0
Name: 4, dtype: object
playerID    armstbo01
yearID           1871
stint               1
teamID            FW1
lgID              NaN
```

```
G                   12
AB                  49
R                    9
H                   11
2B                   2
3B                   1
HR                   0
RBI                  5
SB                   0
CS                   1
BB                   0
SO                   1
IBB                NaN
HBP                NaN
SH                 NaN
SF                 NaN
GIDP                 0
Name: 5, dtype: object
playerID    barkeal01
yearID           1871
stint               1
teamID            RC1
lgID              NaN
G                   1
AB                  4
R                   0
H                   1
2B                  0
3B                  0
HR                  0
RBI                 2
SB                  0
CS                  0
BB                  1
SO                  0
IBB               NaN
HBP               NaN
SH                NaN
SF                NaN
GIDP                0
Name: 6, dtype: object
```

Documentation for **iterrows**.

PART THREE

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

For Questions 15-16: We will be using the **'student.txt' dataset** and the **student DataFrame** </div>

Question 15:

a. Read in the dataset **'student.txt'** and store the results in a DataFrame named **student**.

See the link below for access to the dataset.

student.txt

b. Use appropriate attributes and methods to inspect the **student** DataFrame.

In [55]:
```
student = pd.read_table('student.txt',sep=';')
print(student.head())
print(student.tail())
print(student.info())
```

```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      
Fjob  ...  \
0     GP   F   18       U     GT3       A     4     4  at_home   t
eacher  ...
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  se
rvices  ...
4     GP   F   16       U     GT3       T     3     3    other    
other  ...

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4        3      4     1     1      3        4   0  11  11
1      5        3      3     1     1      3        2   9  11  11
2      4        3      2     2     3      3        6  12  13  12
3      3        2      2     1     1      5        0  14  14  14
4      4        3      2     1     2      5        0  11  13  13

[5 rows x 33 columns]
    school sex  age address famsize Pstatus  Medu  Fedu      Mjob  
Fjob  \
644     MS   F   19       R     GT3       T     2     3  services  
other   
645     MS   F   18       U     LE3       T     3     1   teacher  
services   
646     MS   F   18       U     GT3       T     1     1     other  
other   
647     MS   M   17       U     LE3       T     3     1  services  
services   
648     MS   M   18       R     LE3       T     3     2  services  
other   

     ... famrel freetime  goout  Dalc  Walc health absences  G1  G
2  G3
644  ...      5        4      2     1     2      5        4  10  1
1  10
645  ...      4        3      4     1     1      1        4  15  1
5  16
646  ...      1        1      1     1     1      5        6  11  1
2   9
647  ...      2        4      5     3     4      2        6  10  1
0  10
648  ...      4        4      1     3     4      5        4  10  1
1  11

[5 rows x 33 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   school      649 non-null    object
 1   sex         649 non-null    object
 2   age         649 non-null    int64 
 3   address     649 non-null    object
 4   famsize     649 non-null    object
 5   Pstatus     649 non-null    object
 6   Medu        649 non-null    int64 
 7   Fedu        649 non-null    int64
 8   Mjob        649 non-null    object
 9   Fjob        649 non-null    object
 10  reason      649 non-null    object
 11  guardian    649 non-null    object
 12  traveltime  649 non-null    int64
 13  studytime   649 non-null    int64
 14  failures    649 non-null    int64
 15  schoolsup   649 non-null    object
 16  famsup      649 non-null    object
 17  paid        649 non-null    object
 18  activities  649 non-null    object
 19  nursery     649 non-null    object
 20  higher      649 non-null    object
 21  internet    649 non-null    object
 22  romantic    649 non-null    object
 23  famrel      649 non-null    int64
 24  freetime    649 non-null    int64
 25  goout       649 non-null    int64
 26  Dalc        649 non-null    int64
 27  Walc        649 non-null    int64
 28  health      649 non-null    int64
 29  absences    649 non-null    int64
 30  G1          649 non-null    int64
 31  G2          649 non-null    int64
 32  G3          649 non-null    int64
dtypes: int64(16), object(17)
memory usage: 167.4+ KB
None
```

Question 16:

a. Read in the dataset **'student.txt'** again, but use the **select_dtypes() method** to include only the numeric columns. Store the results in a DataFrame named **student_2**.

b. Use appropriate attributes and methods to check that the **student_2** DataFrame contains the correct columns.

Documentation for **'select_dtypes'**

In [60]:
```
student_2 = pd.read_table('student.txt',sep=';')
student_2.select_dtypes(include=['int64'])
```

Out[60]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime
0	18	4	4	2	2	0	4	3
1	17	1	1	1	2	0	5	3
2	15	1	1	1	2	0	4	3
3	15	4	2	1	3	0	3	2
4	16	3	3	1	2	0	4	3
...	...	...	...	...	...	...	...	...
644	19	2	3	1	3	1	5	4
645	18	3	1	1	2	0	4	3
646	18	1	1	2	2	0	1	1
647	17	3	1	2	1	0	2	4
648	18	3	2	3	1	0	4	4

649 rows × 16 columns

PART FOUR

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

For Questions 17-20: We will be using the **'marketing_campaign.csv' dataset** and the **customers DataFrame**
</div>

Question 17:

a. Read in the dataset **'marketing_campaign.csv'** and store the results in a DataFrame named **customers**.

b. Use appropriate attributes and methods to inspect the **customers** DataFrame.

In [63]:
```
customers = pd.read_csv('marketing_campaign.csv',sep=';')
print(customers.head())
print(customers.tail())
print(customers.info())
```

```
     ID  Year_Birth   Education Marital_Status   Income  Kidhom
e  Teenhome  \
0  5524        1957  Graduation         Single  58138.0        
0         0   
1  2174        1954  Graduation         Single  46344.0        
1         1   
2  4141        1965  Graduation       Together  71613.0        
0         0   
3  6182        1984  Graduation       Together  26646.0        
1         0   
4  5324        1981         PhD        Married  58293.0        
1         0   

  Dt_Customer  Recency  MntWines  ...  NumWebVisitsMonth  Accep
tedCmp3  \
0  2012-09-04       58       635  ...                  7       
0   
1  2014-03-08       38        11  ...                  5       
0   
2  2013-08-21       26       426  ...                  4       
0   
3  2014-02-10       26        11  ...                  6       
0   
4  2014-01-19       94       173  ...                  5       
0   

   AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Comp
lain  \
0             0             0             0             0      
0   
1             0             0             0             0      
0   
2             0             0             0             0      
0   
3             0             0             0             0      
0   
4             0             0             0             0      
0   

   Z_CostContact  Z_Revenue  Response  
0              3         11         1  
1              3         11         0  
2              3         11         0  
3              3         11         0  
4              3         11         0  

[5 rows x 29 columns]
         ID  Year_Birth   Education Marital_Status   Income  Ki
dhome  \
2235  10870        1967  Graduation        Married  61223.0    
0   
2236   4001        1946         PhD       Together  64014.0    
2   
2237   7270        1981  Graduation       Divorced  56981.0    
0   
2238   8235        1956      Master       Together  69245.0    
0   
2239   9405        1954         PhD        Married  52869.0    
1   
```

```
      Teenhome Dt_Customer  Recency  MntWines  ...  NumWebVisit
sMonth  \
2235         1  2013-06-13       46       709  ...
5
2236         1  2014-06-10       56       406  ...
7
2237         0  2014-01-25       91       908  ...
6
2238         1  2014-01-24        8       428  ...
3
2239         1  2012-10-15       40        84  ...
7

      AcceptedCmp3  AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  A
cceptedCmp2  \
2235             0             0             0             0
0
2236             0             0             0             1
0
2237             0             1             0             0
0
2238             0             0             0             0
0
2239             0             0             0             0
0

      Complain  Z_CostContact  Z_Revenue  Response
2235         0              3         11         0
2236         0              3         11         0
2237         0              3         11         0
2238         0              3         11         0
2239         0              3         11         1

[5 rows x 29 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  AcceptedCmp3         2240 non-null   int64
 21  AcceptedCmp4         2240 non-null   int64
 22  AcceptedCmp5         2240 non-null   int64
 23  AcceptedCmp1         2240 non-null   int64
 24  AcceptedCmp2         2240 non-null   int64
 25  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
None
```

Documentation for describe()

Question 18:

Use the appropriate method to generate summary statistics for the numeric columns of the **customers** DataFrame.

In [64]:
```
customers.describe()
```

Out[64]:

	ID	Year_Birth	Income	Kidhome	Teenhome
count	2240.000000	2240.000000	2216.000000	2240.000000	2240.000000
mean	5592.159821	1968.805804	52247.251354	0.444196	0.506250
std	3246.662198	11.984069	25173.076661	0.538398	0.544538
min	0.000000	1893.000000	1730.000000	0.000000	0.000000
25%	2828.250000	1959.000000	35303.000000	0.000000	0.000000
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000
75%	8427.750000	1977.000000	68522.000000	1.000000	1.000000
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000

8 rows × 26 columns

Question 19:

Use the appropriate method to generate summary statistics for the non-numeric columns of the **customers** DataFrame.

In [70]:
```
customers.describe(include=['object'])
```

Out[70]:

	Education	Marital_Status	Dt_Customer
count	2240	2240	2240
unique	5	8	663
top	Graduation	Married	2012-08-31
freq	1127	864	12

Question 20:

Use the appropriate method to generate summary statistics for the all columns of the **customers** DataFrame.

In [72]:
```
customers.describe(include=['object',np.number])
```

Out[72]:

	ID	Year_Birth	Education	Marital_Status	In
count	2240.000000	2240.000000	2240	2240	2216.0
unique	NaN	NaN	5	8	
top	NaN	NaN	Graduation	Married	
freq	NaN	NaN	1127	864	
mean	5592.159821	1968.805804	NaN	NaN	52247.2
std	3246.662198	11.984069	NaN	NaN	25173.0
min	0.000000	1893.000000	NaN	NaN	1730.0
25%	2828.250000	1959.000000	NaN	NaN	35303.0
50%	5458.500000	1970.000000	NaN	NaN	51381.5
75%	8427.750000	1977.000000	NaN	NaN	68522.0
max	11191.000000	1996.000000	NaN	NaN	666666.0

11 rows × 29 columns

PART FIVE

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

For Questions 17-20: We will be using the **'cereals.txt' dataset** and the **cereals DataFrame** </div>

Question 21:

a. Read in the dataset **'cerals.txt'** and store the results in a DataFrame named **cereals**.

See the link below for access to the dataset.

cereals.txt

b. Use appropriate attributes and methods to inspect the **cerealss** DataFrame.

In [75]:
```
cereals = pd.read_table('cereals.txt', sep=',')
cereals
```

Out[75]:

	Name	Manufacturer	Type	Calories	Fiber	Sugars
0	100% Bran	Nabisco	Cold	70	10.0	6
1	100% Natural Bran	Quaker Oats	Cold	120	2.0	8
2	All-Bran	Kellogg's	Cold	70	9.0	5
3	All-Bran with Extra Fiber	Kellogg's	Cold	50	14.0	0
4	Almond Delight	Ralston Purina	Cold	110	1.0	8
...	...	...	...	...	...	...
72	Triples	General Mills	Cold	110	0.0	3
73	Trix	General Mills	Cold	110	0.0	12
74	Wheat Chex	Ralston Purina	Cold	100	3.0	3
75	Wheaties	General Mills	Cold	100	3.0	3
76	Wheaties Honey Gold	General Mills	Cold	110	1.0	8

77 rows × 6 columns

Question 22:

</b>
a. Use the **drop() method** to temporarily drop the "**Type** column of the **cereals** DataFrame.

</b> b. Run the **cereals.head()** command. What do you notice?

</b>

Documentation for **'drop'**

In [82]:
```
cereals.drop(["Type"],axis=1)
print("The type is still there because we did not use th
cereals.head()
```

```
The type is still there because we did not use the inpla
ce argument.
```

Out[82]:

	Name	Manufacturer	Type	Calories	Fiber	Sugars
0	100% Bran	Nabisco	Cold	70	10.0	6
1	100% Natural Bran	Quaker Oats	Cold	120	2.0	8
2	All-Bran	Kellogg's	Cold	70	9.0	5
3	All-Bran with Extra Fiber	Kellogg's	Cold	50	14.0	0
4	Almond Delight	Ralston Purina	Cold	110	1.0	8

Question 23:

</b>
a. Use the **drop() method** to temporarily drop the third row of the **cereals**

</b> b. Run the **cereals.head()** command. What do you notice?

</b>

Documentation for **'drop'**

In [84]:
```
cereals.drop(2,axis=0)
print("The third row is still there because we did not
cereals.head()
```

```
The type is still there because we did not use the inp
lace argument.
```

Out[84]:

	Name	Manufacturer	Type	Calories	Fiber	Sugars
0	100% Bran	Nabisco	Cold	70	10.0	6
1	100% Natural Bran	Quaker Oats	Cold	120	2.0	8
2	All-Bran	Kellogg's	Cold	70	9.0	5
3	All-Bran with Extra Fiber	Kellogg's	Cold	50	14.0	0
4	Almond Delight	Ralston Purina	Cold	110	1.0	8

Question 24:

Calculate the mean of each numeric column.

Documentation for **'mean'**

In [87]:
```
cereals.mean(numeric_only=True)
```

Out[87]:
```
Calories    106.883117
Fiber         2.151948
Sugars        6.922078
dtype: float64
```

Question 25:

Calculate the mean of each row.

Documentation for **'mean'**

In [90]:
```
cereals.mean(axis=1,numeric_only=True)
```

Out[90]:
```
0     28.666667
1     43.333333
2     28.000000
3     21.333333
4     39.666667
        ...
72    37.666667
73    40.666667
74    35.333333
75    35.333333
76    39.666667
Length: 77, dtype: float64
```

Question 26:

Calculate the mean of each numeric column using the alias "index" for axis=0.

In [91]:
```
cereals.mean(axis=0,numeric_only=True)
```

Out[91]:
```
Calories    106.883117
Fiber         2.151948
Sugars        6.922078
dtype: float64
```

Question 27:

Calculate the mean of each row using the alias "columns" for axis=1.

In [92]:
```
cereals.mean(axis=1,numeric_only=True)
```

Out[92]:
```
0     28.666667
1     43.333333
2     28.000000
3     21.333333
4     39.666667
        ...
72    37.666667
73    40.666667
74    35.333333
75    35.333333
76    39.666667
Length: 77, dtype: float64
```