Asg 19.9

Pandas GroupBy Object

(Coding)

Files needed for this assignment:

menu.csv
food_prices.txt
marketing_campaign.csv

In [5]:
```
# 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"

For Questions 1-6: We will be using the **'marketing_campaign.csv' dataset** and the **customers DataFrame**

Question 1:

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

See the links below for access to the dataset and for information about the dataset.

Customer Personality Analysis 1

Customer Personality Analysis 2

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

- head()
- tail()
- info()
- index
- columns
- shape
- dtypes

In [6]:
```
customers = pd.read_csv('marketing_campaign.csv',sep=';')
print(customers.head())
print(customers.tail())
print(customers.info())
print(customers.index)
print(customers.columns)
print(menu.shape)
print(menu.dtypes)
```

```
     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
RangeIndex(start=0, stop=2240, step=1)
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response'],
      dtype='object')
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [6], in <cell line: 7>()
      5 print(customers.index)
      6 print(customers.columns)
----> 7 print(menu.shape)
      8 print(menu.dtypes)

NameError: name 'menu' is not defined
```

Question 2:

Calculate the mean **Income** across the entire dataset.

In [14]:
```
print(f"Income Mean: {customers.Income.mean():.2f}")
```

```
Income Mean: 52247.25
```

Question 16:

Calculate the mean **Income** just for Single people.

In [15]:
```
print(f"Single Income Mean: {customers.loc[(customers['Marital_Status']=='Single'),
```

```
Single Income Mean: 50995.35
```

Question 3:

Calculate the mean **Income** for each **Marital_Status**.

Documentation for **'groupby'**

In [17]:
```
customers.groupby(['Marital_Status']).Income.mean()
```

Out[17]:
```
Marital_Status
Absurd      72365.500000
Alone       43789.000000
Divorced    52834.228448
Married     51724.978996
Single      50995.350318
Together    53245.534031
Widow       56481.552632
YOLO        48432.000000
Name: Income, dtype: float64
```

Question 4:

a. Calculate the maximum **Income** for each **Marital_Status**.

b. Calculate the minimum **Income** for each **Marital_Status.**

In [21]:
```
print(customers.groupby(['Marital_Status']).Income.min())
customers.groupby(['Marital_Status']).Income.max()
```

```
Marital_Status
Absurd      65487.0
Alone       34176.0
Divorced     1730.0
Married      2447.0
Single       3502.0
Together     5648.0
Widow       22123.0
YOLO        48432.0
Name: Income, dtype: float64
```

Out[21]:
```
Marital_Status
Absurd       79244.0
Alone        61331.0
Divorced    153924.0
Married     160803.0
Single      113734.0
Together    666666.0
Widow        85620.0
YOLO         48432.0
Name: Income, dtype: float64
```

Question 5:

Calculate the count, mean, median, maximum, and minimum for the **Income** Series for each **Marital Status** ... i.e., apply multiple aggregation functions simultaneously.

In [23]:
```
customers.groupby(['Marital_Status']).Income.agg(['mean','median','max','min'])
```

Out[23]:

Marital_Status	mean	median	max	min
Absurd	72365.500000	72365.5	79244.0	65487.0
Alone	43789.000000	35860.0	61331.0	34176.0
Divorced	52834.228448	52683.0	153924.0	1730.0
Married	51724.978996	51876.0	160803.0	2447.0
Single	50995.350318	48904.0	113734.0	3502.0
Together	53245.534031	51369.0	666666.0	5648.0
Widow	56481.552632	56551.0	85620.0	22123.0
YOLO	48432.000000	48432.0	48432.0	48432.0

Question 6:

Calculate the mean for all numeric columns for each **Marital Status**.

In [24]:
```
customers.groupby(['Marital_Status']).mean(numeric_only=True)
```

Out[24]:

Marital_Status	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	Mnt
Absurd	6051.500000	1975.000000	72365.500000	0.000000	0.000000	53.000000	355.5
Alone	2728.333333	1973.000000	43789.000000	1.000000	0.666667	30.333333	184.6
Divorced	5427.060345	1966.275862	52834.228448	0.413793	0.590517	49.487069	324.8
Married	5633.152778	1969.579861	51724.978996	0.456019	0.511574	48.277778	299.4
Single	5489.241667	1971.489583	50995.350318	0.464583	0.406250	49.506250	288.3
Together	5644.674138	1967.746552	53245.534031	0.450000	0.529310	50.106897	306.8
Widow	5969.558442	1958.558442	56481.552632	0.233766	0.636364	49.142857	369.2
YOLO	5812.500000	1973.000000	48432.000000	0.000000	1.000000	3.000000	322.0

8 rows × 26 columns

We will now do some data visualization ... This part is optional, but recommended.

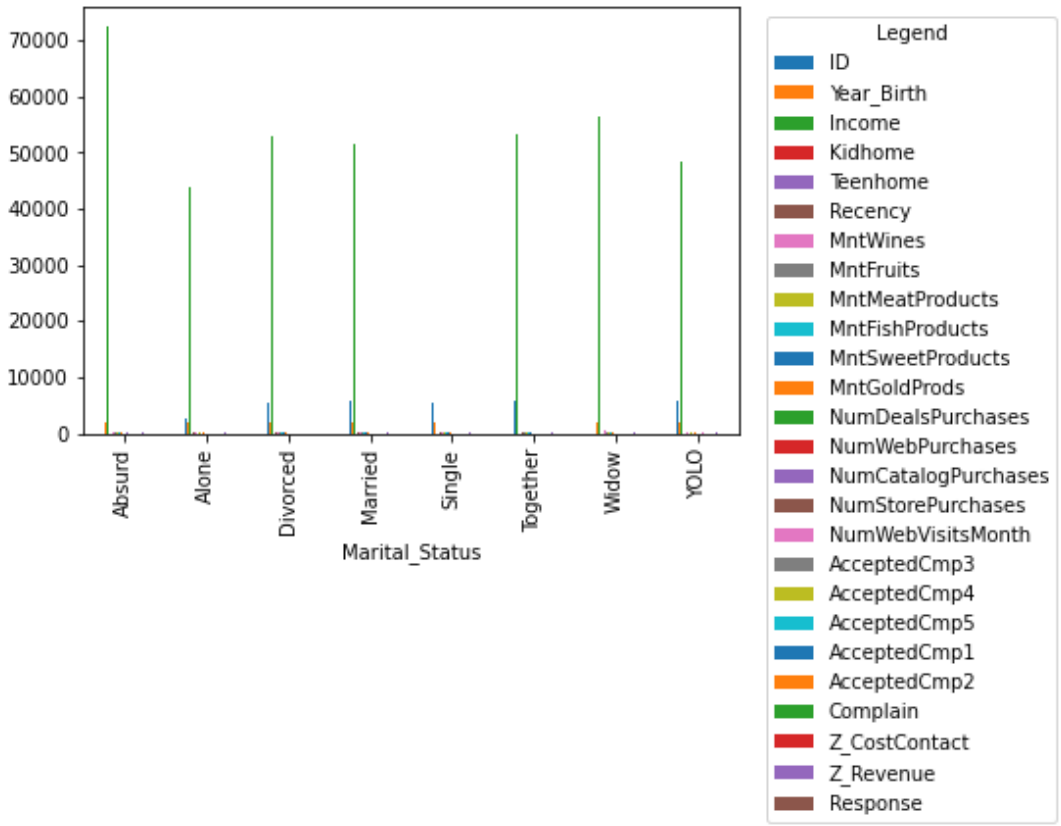
In [29]:
```
# allow plots to appear in the notebook
import matplotlib.pyplot as plt
%matplotlib inline
```

Question 7:

Construct side-by-side bar graphs for the DataFrame from Question 6.

In [42]:
```
customers.groupby(['Marital_Status']).mean(numeric_only=True).plot(kind='bar
plt.legend(bbox_to_anchor=(1.5, 1),loc='upper right', title='Legend')
```

Out[42]:
```
<matplotlib.legend.Legend at 0x1bbaac26ee0>
```



Question 8:

Calculate the mean for all numeric columns for each level of **Education**.

In [40]:
```
customers.groupby(['Education']).mean(numeric_only=True)
```

Out[40]:

Education	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	Mnt
2n Cycle	5588.211823	1972.024631	47633.190000	0.477833	0.408867	48.418719	198.
Basic	5396.407407	1977.462963	20306.259259	0.629630	0.092593	48.444444	7.
Graduation	5652.523514	1969.635315	52720.373656	0.444543	0.494232	50.035492	284.
Master	5403.648649	1966.878378	52917.534247	0.454054	0.535135	47.586486	333.
PhD	5619.096708	1966.043210	56145.313929	0.401235	0.598765	48.483539	404.

5 rows × 26 columns

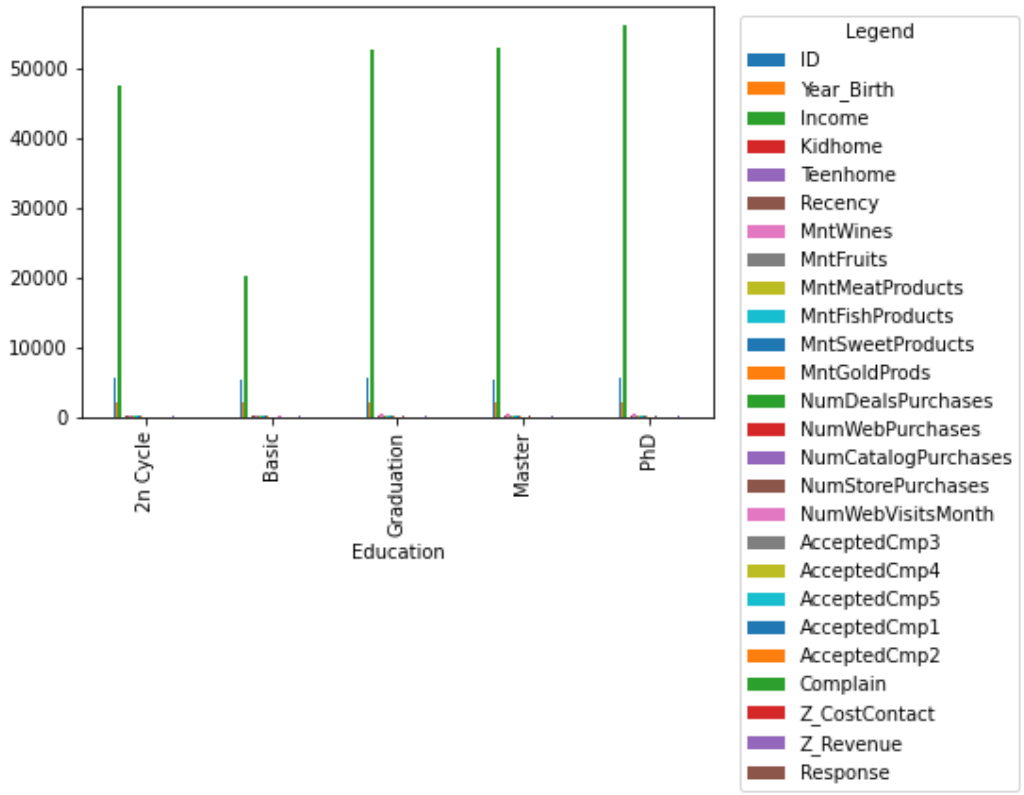
Question 9:

Construct side-by-side bar graphs for the DataFrame from Question 9.

In [43]:
```
customers.groupby(['Education']).mean(numeric_only=True).plot(kind='bar')
```

```
plt.legend(bbox_to_anchor=(1.5, 1),loc='upper right', title='Legend')
```

Out[43]:
```
<matplotlib.legend.Legend at 0x1bbab1849d0>
```



Question 10:

Find the mean, median, and count for the **Income**e Series grouped first by level of **Education** and then by **Marital_Status**.

In [45]:
```
customers.groupby(['Education','Marital_Status']).Income.agg(['mean','me
```

Out[45]:

Education	Marital_Status	mean	median	count
2n Cycle	Divorced	49395.130435	49118.0	23
	Married	46201.100000	46462.5	80
	Single	53673.944444	48668.5	36
	Together	44736.410714	45774.0	56
	Widow	51392.200000	47682.0	5
Basic	Divorced	9548.000000	9548.0	1
	Married	21960.500000	22352.0	20
	Single	18238.666667	16383.0	18
	Together	21240.071429	23179.0	14
	Widow	22123.000000	22123.0	1
Graduation	Absurd	79244.000000	79244.0	1
	Alone	34176.000000	34176.0	1
	Divorced	54526.042017	55635.0	119
	Married	50800.258741	50737.0	429
	Single	51322.182927	49973.5	246
	Together	55758.480702	53977.0	285
	Widow	54976.657143	58275.0	35
Master	Absurd	65487.000000	65487.0	1
	Alone	61331.000000	61331.0	1
	Divorced	50331.945946	49476.0	37
	Married	53286.028986	53088.5	138
	Single	53530.560000	49494.0	75
	Together	52109.009804	49736.0	102
	Widow	58401.545455	51529.0	11
PhD	Alone	35860.000000	35860.0	1
	Divorced	53096.615385	50613.5	52
	Married	58138.031579	57081.5	190
	Single	53314.614583	50198.0	96
	Together	56041.422414	56756.0	116
	Widow	60288.083333	57032.0	24
	YOLO	48432.000000	48432.0	2

Question 11:

Find the mean, maximum, minimum, and count for the **NumWebVisitsMonth** Series grouped first by **Marital_Status** and then by level of **Education**.

In [46]:
```
customers.groupby(['Marital_Status','Education']).NumWebVisitsMonth.ag
```

Out[46]:

Marital_Status	Education	mean	max	min
Absurd	**Graduation**	1.000000	1	1
	Master	2.000000	2	2
Alone	**Graduation**	6.000000	6	6
	Master	8.000000	8	8
	PhD	5.000000	5	5
Divorced	**2n Cycle**	6.217391	8	2
	Basic	8.000000	8	8
	Graduation	5.344538	20	0
	Master	5.405405	9	1
	PhD	5.500000	9	1
Married	**2n Cycle**	5.333333	9	1
	Basic	7.050000	9	3
	Graduation	5.390300	13	0
	Master	5.224638	9	1
	PhD	5.244792	19	0
Single	**2n Cycle**	5.270270	9	1
	Basic	6.666667	9	3
	Graduation	5.289683	19	0
	Master	5.146667	17	1
	PhD	5.122449	9	0
Together	**2n Cycle**	5.473684	9	0
	Basic	6.928571	9	5
	Graduation	5.178322	10	1
	Master	5.094340	10	1
	PhD	5.307692	20	0
Widow	**2n Cycle**	5.000000	8	3
	Basic	5.000000	5	5
	Graduation	4.800000	9	1
	Master	4.916667	9	1
	PhD	4.958333	9	0
YOLO	**PhD**	8.000000	8	8

In [ ]: