Step 4. See the first 10 entries

order_id quantity item_name \ 0 1 1 Chips and Fresh Tomato Salsa 1 1 1 Izze 2 1 1 Nantucket Nectar 3 1 1 Chips and Tomatillo-Green Chili Salsa 4 2 2 Chicken Bowl 5 3 1 Chicken Bowl 6 3 1 Side of Chips 7 4 1 Steak Burrito 8 4 1 Steak Soft Tacos 9 5 1 Steak Burrito

choice_description item_price 0 NaN $2.39 1 [Clementine] $3.39 2 [Apple] $3.39 3 NaN $2.39 4 [Tomatillo-Red Chili Salsa (Hot), [Black Beans... $16.98 5 [Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou... $10.98 6 NaN $1.69 7 [Tomatillo Red Chili Salsa, [Fajita Vegetables... $11.75 8 [Tomatillo Green Chili Salsa, [Pinto Beans, Ch... $9.25 9 [Fresh Tomato Salsa, [Rice, Black Beans, Pinto... $9.25

Step 5. What is the number of observations in the dataset?

4622

Step 6. What is the number of columns in the dataset?

5

Step 7. Print the name of all the columns.

```
Index([u'order_id', u'quantity', u'item_name', u'choice_description',  u'item_price'],
 dtype='object')
```

Step 8. How is the dataset indexed?

```
RangeIndex(start=0, stop=4622, step=1)
```

Step 9. Which was the most-ordered item?

```
 order_id quantity
item_name
Chicken Bowl 713926 761
```

Step 10. For the most-ordered item, how many items were ordered?

```
 order_id quantity
item_name
Chicken Bowl 713926 761
```

Step 11. What was the most ordered item in the choice_description column?

```
 order_id quantity
choice_description
 [Diet Coke] 123455 159
```

Step 12. How many items were orderd in total?

```
4972
```

Step 13. Turn the item price into a float

Step 13.a. Check the item price type

```
dtype('O')
```

Step 13.b. Create a lambda function and change the type of item price

Step 13.c. Check the item price type

```
dtype('float64')
```

Step 14. How much was the revenue for the period in the dataset?

```
Revenue was: $39237.02
```

Step 15. How many orders were made in the period?

```
1834
```

Step 16. What is the average revenue amount per order?

```
21.394231188658654
```

Step 17. How many different items are sold?

50

Step 17. How many products cost more than $10.00?

12

Step 18. Sort by the name of the item

```
 order_id quantity item_name \
3389 1360 2 6 Pack Soft Drink
341 148 1 6 Pack Soft Drink
1849 749 1 6 Pack Soft Drink
1860 754 1 6 Pack Soft Drink
2713 1076 1 6 Pack Soft Drink
3422 1373 1 6 Pack Soft Drink
553 230 1 6 Pack Soft Drink
1916 774 1 6 Pack Soft Drink
1922 776 1 6 Pack Soft Drink
1937 784 1 6 Pack Soft Drink
3836 1537 1 6 Pack Soft Drink
298 129 1 6 Pack Soft Drink
1976 798 1 6 Pack Soft Drink
1167 481 1 6 Pack Soft Drink
3875 1554 1 6 Pack Soft Drink
1124 465 1 6 Pack Soft Drink
3886 1558 1 6 Pack Soft Drink
2108 849 1 6 Pack Soft Drink
3010 1196 1 6 Pack Soft Drink
4535 1803 1 6 Pack Soft Drink
4169 1664 1 6 Pack Soft Drink
4174 1666 1 6 Pack Soft Drink
4527 1800 1 6 Pack Soft Drink
4522 1798 1 6 Pack Soft Drink
3806 1525 1 6 Pack Soft Drink
2389 949 1 6 Pack Soft Drink
3132 1248 1 6 Pack Soft Drink
3141 1253 1 6 Pack Soft Drink
639 264 1 6 Pack Soft Drink
1026 422 1 6 Pack Soft Drink
... ... ... ...
2996 1192 1 Veggie Salad
3163 1263 1 Veggie Salad
4084 1635 1 Veggie Salad
1694 686 1 Veggie Salad
2756 1094 1 Veggie Salad
4201 1677 1 Veggie Salad Bowl
1884 760 1 Veggie Salad Bowl
455 195 1 Veggie Salad Bowl
```

```
3223 1289 1 Veggie Salad Bowl
2223 896 1 Veggie Salad Bowl
2269 913 1 Veggie Salad Bowl
4541 1805 1 Veggie Salad Bowl
```

3293 1321 1 Veggie Salad Bowl
186 83 1 Veggie Salad Bowl
960 394 1 Veggie Salad Bowl
1316 536 1 Veggie Salad Bowl
2156 869 1 Veggie Salad Bowl
4261 1700 1 Veggie Salad Bowl
295 128 1 Veggie Salad Bowl
4573 1818 1 Veggie Salad Bowl
2683 1066 1 Veggie Salad Bowl
496 207 1 Veggie Salad Bowl
4109 1646 1 Veggie Salad Bowl
738 304 1 Veggie Soft Tacos
3889 1559 2 Veggie Soft Tacos
2384 948 1 Veggie Soft Tacos
781 322 1 Veggie Soft Tacos
2851 1132 1 Veggie Soft Tacos
1699 688 1 Veggie Soft Tacos
1395 567 1 Veggie Soft Tacos

choice_description item_price 3389 [Diet Coke] 12.98 341 [Diet Coke] 6.49 1849 [Coke] 6.49 1860 [Diet Coke] 6.49 2713 [Coke] 6.49 3422 [Coke] 6.49 553 [Diet Coke] 6.49 1916 [Diet Coke] 6.49 1922 [Coke] 6.49 1937 [Diet Coke] 6.49 3836 [Coke] 6.49 298 [Sprite] 6.49 1976 [Diet Coke] 6.49 1167 [Coke] 6.49 3875 [Diet Coke] 6.49 1124 [Coke] 6.49 3886 [Diet Coke] 6.49 2108 [Coke] 6.49 3010 [Diet Coke] 6.49 4535 [Lemonade] 6.49 4169 [Diet Coke] 6.49 4174 [Coke] 6.49 4527 [Diet Coke] 6.49 4522 [Diet Coke] 6.49 3806 [Sprite] 6.49

2389 [Coke] 6.49 3132 [Diet Coke] 6.49 3141 [Lemonade] 6.49 639 [Diet Coke] 6.49 1026 [Sprite] 6.49 ... ... 2996 [Roasted Chili Corn Salsa (Medium), [Black Bea... 8.49 3163 [[Fresh Tomato Salsa (Mild), Roasted Chili Cor... 8.49 4084 [[Fresh Tomato Salsa (Mild), Roasted Chili Cor... 8.49 1694 [[Fresh Tomato Salsa (Mild), Roasted Chili Cor... 8.49 2756 [[Tomatillo-Green Chili Salsa (Medium), Roaste... 8.49 4201 [Fresh Tomato Salsa, [Fajita Vegetables, Black... 11.25 1884 [Fresh Tomato Salsa, [Fajita Vegetables, Rice,... 11.25 455 [Fresh Tomato Salsa, [Fajita Vegetables, Rice,... 11.25 3223 [Tomatillo Red Chili Salsa, [Fajita Vegetables... 11.25 2223 [Roasted Chili Corn Salsa, Fajita Vegetables] 8.75 2269 [Fresh Tomato Salsa, [Fajita Vegetables, Rice,... 8.75 4541 [Tomatillo Green Chili Salsa, [Fajita Vegetabl... 8.75 3293 [Fresh Tomato Salsa, [Rice, Black Beans, Chees... 8.75 186 [Fresh Tomato Salsa, [Fajita Vegetables, Rice,... 11.25 960 [Fresh Tomato Salsa, [Fajita Vegetables, Lettu... 8.75 1316 [Fresh Tomato Salsa, [Fajita Vegetables, Rice,... 8.75 2156 [Tomatillo Red Chili Salsa, [Fajita Vegetables... 11.25 4261 [Fresh Tomato Salsa, [Fajita Vegetables, Rice,... 11.25 295 [Fresh Tomato Salsa, [Fajita Vegetables, Lettu... 11.25 4573 [Fresh Tomato Salsa, [Fajita Vegetables, Pinto... 8.75 2683 [Roasted Chili Corn Salsa, [Fajita Vegetables,... 8.75 496 [Fresh Tomato Salsa, [Rice, Lettuce, Guacamole... 11.25 4109 [Tomatillo Red Chili Salsa, [Fajita Vegetables... 11.25 738 [Tomatillo Red Chili Salsa, [Fajita Vegetables... 11.25 3889 [Fresh Tomato Salsa (Mild), [Black Beans, Rice... 16.98 2384 [Roasted Chili Corn Salsa, [Fajita Vegetables,... 8.75 781 [Fresh Tomato Salsa, [Black Beans, Cheese, Sou... 8.75 2851 [Roasted Chili Corn Salsa (Medium), [Black Bea... 8.49 1699 [Fresh Tomato Salsa, [Fajita Vegetables, Rice,... 11.25 1395 [Fresh Tomato Salsa (Mild), [Pinto Beans, Rice... 8.49

[4622 rows x 5 columns]

Step 19. What was the quantity of the most expensive item ordered?

```
 order_id quantity item_name  choice_description \
3598 1443 15 Chips and Fresh Tomato Salsa  NaN

 item_price
3598 44.25
```

Step 20. How many times did someone order more than one Canned Soda?

```
20
```

Step 4. Discover what is the mean age per occupation

```
occupation
administrator 38.746835
artist 31.392857
doctor 43.571429
educator 42.010526
engineer 36.388060
entertainment 29.222222
executive 38.718750
healthcare 41.562500
homemaker 32.571429
lawyer 36.750000
librarian 40.000000
marketing 37.615385
none 26.555556
other 34.523810
programmer 33.121212
retired 63.071429
salesman 35.666667
scientist 35.548387
student 22.081633
technician 33.148148
writer 36.311111
Name: age, dtype: float64
```

Step 5. Discover the Male ratio per occupation and sort it from the most to the least

```
doctor 100.000000
engineer 97.014925
technician 96.296296
retired 92.857143
programmer 90.909091
executive 90.625000
scientist 90.322581
entertainment 88.888889
lawyer 83.333333
salesman 75.000000
educator 72.631579
student 69.387755
other 65.714286
marketing 61.538462
writer 57.777778
none 55.555556
```

```
administrator 54.430380
artist 53.571429
librarian 43.137255
healthcare 31.250000
homemaker 14.285714
dtype: float64
```

Step 6. For each occupation, calculate the minimum and maximum ages

```
 min max
occupation
administrator 21 70
artist 19 48
doctor 28 64
educator 23 63
engineer 22 70
entertainment 15 50
executive 22 69
healthcare 22 62
homemaker 20 50
lawyer 21 53
librarian 23 69
marketing 24 55
none 11 55
other 13 64
programmer 20 63
retired 51 73
salesman 18 66
scientist 23 55
student 7 42
technician 21 55
writer 18 60
```

Step 7. For each combination of occupation and gender, calculate the mean age

```
occupation gender
administrator F 40.638889
 M 37.162791
artist F 30.307692
 M 32.333333
doctor M 43.571429
educator F 39.115385
 M 43.101449
```

```
engineer F 29.500000
 M 36.600000
entertainment F 31.000000
 M 29.000000
executive F 44.000000
 M 38.172414
healthcare F 39.818182
 M 45.400000
homemaker F 34.166667
 M 23.000000
lawyer F 39.500000
 M 36.200000
librarian F 40.000000
 M 40.000000
marketing F 37.200000
 M 37.875000
none F 36.500000
 M 18.600000
other F 35.472222
 M 34.028986
programmer F 32.166667
 M 33.216667
retired F 70.000000
 M 62.538462
salesman F 27.000000
 M 38.555556
scientist F 28.333333
 M 36.321429
student F 20.750000
 M 22.669118
technician F 38.000000
 M 32.961538
writer F 37.631579
 M 35.346154
Name: age, dtype: float64
```

Step 8. For each occupation present the percentage of women and men

```
occupation gender
administrator F 45.569620
 M 54.430380
artist F 46.428571
 M 53.571429
doctor M 100.000000
educator F 27.368421
 M 72.631579
```

```
engineer F 2.985075
 M 97.014925
entertainment F 11.111111
 M 88.888889
executive F 9.375000
 M 90.625000
healthcare F 68.750000
 M 31.250000
homemaker F 85.714286
 M 14.285714
lawyer F 16.666667
 M 83.333333
librarian F 56.862745
 M 43.137255
marketing F 38.461538
 M 61.538462
none F 44.444444
 M 55.555556
other F 34.285714
 M 65.714286
programmer F 9.090909
 M 90.909091
retired F 7.142857
 M 92.857143
salesman F 25.000000
 M 75.000000
scientist F 9.677419
 M 90.322581
student F 30.612245
 M 69.387755
technician F 3.703704
 M 96.296296
writer F 42.222222
 M 57.777778
```

```
Name: gender, dtype: float64
```

```
 Year Population Total Violent Property Murder Forcible_Rape \
0 1960 179323175 3384200 288460 3095700 9110 17190 1 1961 182992000 3488000
289390 3198600 8740 17220 2 1962 185771000 3752200 301510 3450700 8530 17550 3
1963 188483000 4109500 316970 3792500 8640 17650 4 1964 191141000 4564600
364220 4200400 9360 21420

  Robbery Aggravated_assault Burglary Larceny_Theft Vehicle_Theft 0 107840 154320
912100 1855400 328200 1 106670 156760 949600 1913000 336000 2 110860 164570
994300 2089600 366800 3 116470 174210 1086400 2297800 408300 4 130390 203050
1213200 2514400 472800
```

Step 4. What is the type of the columns?

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55 entries, 0 to 54
Data columns (total 12 columns):
Year 55 non-null int64
Population 55 non-null int64
Total 55 non-null int64
Violent 55 non-null int64
Property 55 non-null int64
Murder 55 non-null int64
Forcible_Rape 55 non-null int64
Robbery 55 non-null int64
Aggravated_assault 55 non-null int64
Burglary 55 non-null int64
Larceny_Theft 55 non-null int64
Vehicle_Theft 55 non-null int64
dtypes: int64(12)
memory usage: 5.2 KB
```

Have you noticed that the type of Year is int64. But pandas has a different type to work with Time Series. Let's see it now.

Step 5. Convert the type of the column Year to datetime64

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55 entries, 0 to 54
Data columns (total 12 columns):
Year 55 non-null datetime64[ns]
Population 55 non-null int64
Total 55 non-null int64
Violent 55 non-null int64
Property 55 non-null int64
Murder 55 non-null int64
Forcible_Rape 55 non-null int64
Robbery 55 non-null int64
Aggravated_assault 55 non-null int64
Burglary 55 non-null int64
Larceny_Theft 55 non-null int64
Vehicle_Theft 55 non-null int64
dtypes: datetime64[ns](1), int64(11)
memory usage: 5.2 KB
```

Step 6. Set the Year column as the index of the dataframe

```
 Population Total Violent Property Murder  Forcible_Rape \
Year

1960-01-01 179323175 3384200 288460 3095700 9110  17190
1961-01-01 182992000 3488000 289390 3198600 8740  17220
1962-01-01 185771000 3752200 301510 3450700 8530  17550
1963-01-01 188483000 4109500 316970 3792500 8640  17650
1964-01-01 191141000 4564600 364220 4200400 9360  21420

 Robbery Aggravated_assault Burglary Larceny_Theft \ Year
1960-01-01  107840  154320  912100  1855400   1961-01-01  106670  156760  949600
1913000   1962-01-01  110860  164570  994300  2089600   1963-01-01  116470  174210
1086400 2297800  1964-01-01  130390  203050  1213200  2514400
```

```
 Vehicle_Theft
Year
1960-01-01 328200
1961-01-01 336000
1962-01-01 366800
1963-01-01 408300
1964-01-01 472800
```

Step 7. Delete the Total column

```
 Population Violent Property Murder Forcible_Rape  Robbery \
Year

1960-01-01 179323175 288460 3095700 9110 17190  107840
1961-01-01 182992000 289390 3198600 8740 17220  106670
1962-01-01 185771000 301510 3450700 8530 17550  110860
1963-01-01 188483000 316970 3792500 8640 17650  116470
1964-01-01 191141000 364220 4200400 9360 21420  130390

 Aggravated_assault Burglary Larceny_Theft Vehicle_Theft Year

1960-01-01 154320 912100 1855400 328200 1961-01-01 156760 949600 1913000 336000

1962-01-01 164570 994300 2089600 366800 1963-01-01 174210 1086400 2297800 408300

1964-01-01 203050 1213200 2514400 472800
```

Step 8. Group the year by decades and sum the values

Pay attention to the Population column number, summing this column is a mistake

```
 Population Violent Property Murder Forcible_Rape Robbery \
                    1960 201385000 4134930 45160900 106180 236720 1633510
```

```
1970 220099000 9607930 91383800 192230 554570 4159020  1980 248239000 14074328
117048900  206439  865639  5383109   1990  272690813  17527048  119053499  211664
998827 5748930  2000 307006550 13968056 100944369 163068 922499 4230366  2010
318857056 6072017 44095950 72867 421059 1749809

 Aggravated_assault Burglary Larceny_Theft Vehicle_Theft  1960 2158520 13321100
26547700  5292100   1970  4702120  28486000  53157800  9739900   1980  7619130
33073494  72040253  11935411   1990  10568963  26750015  77679366  14624418
2000  8652124  21565176  67970291  11412834   2010  3764142  10125170  30401698
3569080
```

Step 9. What is the most dangerous decade to live in the US?

```
Population 2010
Violent 1990
Property 1990
Murder 1990
Forcible_Rape 1990
Robbery 1990
Aggravated_assault 1990
Burglary 1980
Larceny_Theft 1990
Vehicle_Theft 1990
dtype: int64
```

Step 2. Create 3 differents Series, each of length 100, as follows:

1. The first a random number from 1 to 4
2. The second a random number from 1 to 3
3. The third a random number from 10,000 to 30,000

```
0 2
1 2
2 4
3 2
4 1
5 1
6 2
7 3
8 3
9 2
10 1
11 2
12 4
13 1
14 2
15 3
16 4
17 4
18 4
19 3
20 2
21 1
22 4
23 1
24 3
25 2
26 3
27 1
28 3
29 4
..
70 4
71 2
72 2
73 4
74 2
75 1
76 2
77 4
78 3
79 2
80 2
```

```
81 2
82 4
```

```
83 2
84 2
85 2
86 1
87 3
88 1
89 1
90 1
91 3
92 1
93 2
94 3
95 4
96 4
97 2
98 1
99 3
dtype: int64 0 2 1 3
2 2
3 3
4 3
5 1
6 2
7 1
8 2
9 2
 10 2
 11 3
 12 3
 13 1
 14 3
 15 3
 16 3
 17 1
 18 3
 19 3
20 3
21 3
22 1
23 2
24 3
25 2
26 2
27 1
28 3
29 3
 ..
```

70 3
71 2
72 2
73 2
74 3
75 2
76 3
77 1
78 1
79 1
80 2
81 1
82 1
83 3
84 1
85 3
86 1
87 2
88 3
89 2
90 2
91 3
92 2
93 2
94 2
95 2
96 2
97 3
98 1
99 1
dtype: int64 0 16957 1 24571
2 28303
3 14153
4 23445
5 21444
6 16179
7 22696
8 18595
9 27145
10 14406
11 15011
12 17444
13 26236
14 23808
15 21417
16 15079
17 13100

```
18 21470
19 17082
20 21935
21 26770
22 10059
23 11095
24 25916
25 17137
26 22023
27 21612
28 11446
29 29281
 ...
70 23963
71 26782
72 11199
73 23600
74 26935
75 27365
76 23084
77 19052
78 19922
79 17088
80 25468
81 10924
82 10243
83 19834
84 21288
85 22410
86 22348
87 18812
88 29522
89 20838
90 28695
91 23000
92 21684
93 26316
94 10866
95 12337
96 13480
97 25158
98 25585
99 26142
dtype: int64
```

Step 3. Let's create a DataFrame by joinning the Series by column

```
 0 1 2
0 2 2 16957
 1 2 3 24571
2 4 2 28303
3 2 3 14153
4 1 3 23445
```

Step 4. Change the name of the columns to bedrs, bathrs, price_sqr_meter

```
 bedrs bathrs price_sqr_meter
0 2 2 16957
 1 2 3 24571
2 4 2 28303
3 2 3 14153
4 1 3 23445
```

Step 5. Create a one column DataFrame with the values of the 3 Series and assign it to 'bigcolumn'

```
<class 'pandas.core.frame.DataFrame'>

 0
0 2
1 2
2 4
3 2
4 1
5 1
6 2
7 3
8 3
9 2
10 1
11 2
12 4
13 1
14 2
15 3
16 4
17 4
18 4
19 3
20 2
21 1
```

```
22 4
23 1
24 3
25 2
26 3
27 1
28 3
29 4
 .. ...
70 23963
71 26782
72 11199
73 23600
74 26935
75 27365
76 23084
77 19052
78 19922
79 17088
80 25468
81 10924
82 10243
83 19834
84 21288
85 22410
86 22348
87 18812
88 29522
89 20838
90 28695
91 23000
92 21684
93 26316
94 10866
95 12337
96 13480
97 25158
98 25585
99 26142

 [300 rows x 1 columns]
```

Step 7. Reindex the DataFrame so it goes from 0 to 299

```
 0
0 2
1 2
2 4
```

3 2 4 1 5 1 6
2 7 3 8 3 9 2
10 1 11 2 12
4 13 1 14 2
15 3 16 4 17
4 18 4 19 3
20 2 21 1 22
4 23 1 24 3
25 2 26 3 27
1 28 3 29 4 ..
... 270 23963
271 26782
272 11199
273 23600
274 26935
275 27365
276 23084
277 19052
278 19922
279 17088
280 25468
281 10924
282 10243
283 19834
284 21288
285 22410
286 22348
287 18812
288 29522
289 20838
290 28695

```
291 23000
292 21684
293 26316
294 10866
295 12337
296 13480
297 25158
298 25585
299 26142

[300 rows x 1 columns]
```