Performing Grouped Aggregations

Let us understand how to perform grouped or by key aggregations using Pandas.

- Here are the steps we need to follow:
  - Make sure data is read into Data Frame.
  - Identify the key on which data should be aggregated. If the data has to be aggregated on derived field which is not available as part of the Data Frame, then first we need to update data frame with the derived field.
  - Using the key group the values using `groupby` function on data frame. We can only pass column names from Data Frame as part of `groupby`.
  - Apply required aggregate functions to get aggregated results based up on the key.
- We can apply multiple aggregate functions at a time after creating grouped data frame.
- Pandas Data Frame exposes a function called as `rename` to provide aliases to the aggregated fields.

```
%run 06_csv_to_pandas_data_frame.ipynb
```

- Getting number of orders per day

```
orders
```

	order_id	order_date	order_customer_id	order_status
0	1	2013-07-25 00:00:00.0	11599	CLOSED
1	2	2013-07-25 00:00:00.0	256	PENDING_PAYMENT
2	3	2013-07-25 00:00:00.0	12111	COMPLETE
3	4	2013-07-25 00:00:00.0	8827	CLOSED
4	5	2013-07-25 00:00:00.0	11318	COMPLETE
...	...	...	...	...
68878	68879	2014-07-09 00:00:00.0	778	COMPLETE
68879	68880	2014-07-13 00:00:00.0	1117	COMPLETE
68880	68881	2014-07-19 00:00:00.0	2518	PENDING_PAYMENT
68881	68882	2014-07-22 00:00:00.0	10000	ON_HOLD
68882	68883	2014-07-23 00:00:00.0	5533	COMPLETE

68883 rows × 4 columns

```
orders.groupby(orders['order_date'])
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f1f811bb828>
```

```
list(orders.groupby(orders['order_date'])['order_id'])[:3]
```

```
[('2013-07-25 00:00:00.0',
  0            1
  1            2
  2            3
  3            4
  4            5
           ...  
  57786    57787
  57787    57788
  57788    57789
  67415    67416
  68690    68691
  Name: order_id, Length: 143, dtype: int64),
 ('2013-07-26 00:00:00.0',
  104        105
  105        106
  106        107
  107        108
  108        109
           ...  
  67418    67419
  67419    67420
  67420    67421
  67421    67422
  68691    68692
  Name: order_id, Length: 269, dtype: int64),
 ('2013-07-27 00:00:00.0',
  346        347
  347        348
  348        349
  349        350
  350        351
           ...  
  67422    67423
  67423    67424
  67424    67425
  67425    67426
  68692    68693
  Name: order_id, Length: 202, dtype: int64)]
```

```
orders.groupby(orders['order_date'])['order_id'].count()
```

```
order_date
2013-07-25 00:00:00.0    143
2013-07-26 00:00:00.0    269
2013-07-27 00:00:00.0    202
2013-07-28 00:00:00.0    187
2013-07-29 00:00:00.0    253
                        ... 
2014-07-20 00:00:00.0    285
2014-07-21 00:00:00.0    235
2014-07-22 00:00:00.0    138
2014-07-23 00:00:00.0    166
2014-07-24 00:00:00.0    185
Name: order_id, Length: 364, dtype: int64
```

- Getting number of orders per status

```
orders.groupby('order_status')['order_status'].count()
```

```
order_status
CANCELED            1428
CLOSED              7556
COMPLETE           22899
ON_HOLD             3798
PAYMENT_REVIEW       729
PENDING             7610
PENDING_PAYMENT    15030
PROCESSING          8275
SUSPECTED_FRAUD     1558
Name: order_status, dtype: int64
```

- Computing revenue per order

```
order_items
```

	order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order
0	1	1	957	1	
1	2	2	1073	1	
2	3	2	502	5	
3	4	2	403	1	
4	5	4	897	2	
...	...	...	...	...	
172193	172194	68881	403	1	
172194	172195	68882	365	1	
172195	172196	68882	502	1	
172196	172197	68883	208	1	
172197	172198	68883	502	3	

172198 rows × 6 columns

```
list(order_items. \
    groupby('order_item_order_id')['order_item_subtotal'])[:5]
```

```
[(1,
  0    299.98
  Name: order_item_subtotal, dtype: float64),
 (2,
  1    199.99
  2    250.00
  3    129.99
  Name: order_item_subtotal, dtype: float64),
 (4,
  4     49.98
  5    299.95
  6    150.00
  7    199.92
  Name: order_item_subtotal, dtype: float64),
 (5,
  8     299.98
  9     299.95
  10     99.96
  11    299.98
  12    129.99
  Name: order_item_subtotal, dtype: float64),
 (7,
  13    199.99
  14    299.98
  15     79.95
  Name: order_item_subtotal, dtype: float64)]
```

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    sum()
```

```
order_item_order_id
1         299.98
2         579.98
4         699.85
5        1129.86
7         579.92
          ...
68879    1259.97
68880     999.77
68881     129.99
68882     109.99
68883    2149.99
Name: order_item_subtotal, Length: 57431, dtype: float64
```

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    agg(['sum', 'min', 'max', 'count'])
```

	sum	min	max	count
order_item_order_id				
1	299.98	299.98	299.98	1
2	579.98	129.99	250.00	3
4	699.85	49.98	299.95	4
5	1129.86	99.96	299.98	5
7	579.92	79.95	299.98	3
...	...	...	...	...
68879	1259.97	129.99	999.99	3
68880	999.77	149.94	250.00	5
68881	129.99	129.99	129.99	1
68882	109.99	50.00	59.99	2
68883	2149.99	150.00	1999.99	2

57431 rows × 4 columns

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    agg(['sum', 'min', 'max', 'count']). \
    rename(columns={'count': 'item_count', 'sum': 'revenue'})
```

	revenue	min	max	item_count
order_item_order_id				
1	299.98	299.98	299.98	1
2	579.98	129.99	250.00	3
4	699.85	49.98	299.95	4
5	1129.86	99.96	299.98	5
7	579.92	79.95	299.98	3
...	...	...	...	...
68879	1259.97	129.99	999.99	3
68880	999.77	149.94	250.00	5
68881	129.99	129.99	129.99	1
68882	109.99	50.00	59.99	2
68883	2149.99	150.00	1999.99	2

57431 rows × 4 columns

```
order_items.rename(columns={'order_item_order_id': 'order_id'})
```

	order_item_id	order_id	order_item_product_id	order_item_quantity	order_item_subto
0	1	1	957	1	299.
1	2	2	1073	1	199.
2	3	2	502	5	250.
3	4	2	403	1	129.
4	5	4	897	2	49.
...	...	...	...	...	
172193	172194	68881	403	1	129.
172194	172195	68882	365	1	59.
172195	172196	68882	502	1	50.
172196	172197	68883	208	1	1999.
172197	172198	68883	502	3	150.

172198 rows × 6 columns

Task 1

Get order_item_count and order_revenue for each order_id.

```
order_items
```

	order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order
0	1	1	957	1	
1	2	2	1073	1	
2	3	2	502	5	
3	4	2	403	1	
4	5	4	897	2	
...	...	...	...	...	
172193	172194	68881	403	1	
172194	172195	68882	365	1	
172195	172196	68882	502	1	
172196	172197	68883	208	1	
172197	172198	68883	502	3	

172198 rows × 6 columns

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    agg(['sum', 'count']). \
    rename(columns={'sum': 'order_revenue', 'count': 'order_item_count'}). \
    reset_index()
```

	order_item_order_id	order_revenue	order_item_count
0	1	299.98	1
1	2	579.98	3
2	4	699.85	4
3	5	1129.86	5
4	7	579.92	3
...	...	...	...
57426	68879	1259.97	3
57427	68880	999.77	5
57428	68881	129.99	1
57429	68882	109.99	2
57430	68883	2149.99	2

57431 rows × 3 columns

Task 2

Get order count by month using orders data for specific order_status.

```
orders
```

	order_id	order_date	order_customer_id	order_status
0	1	2013-07-25 00:00:00.0	11599	CLOSED
1	2	2013-07-25 00:00:00.0	256	PENDING_PAYMENT
2	3	2013-07-25 00:00:00.0	12111	COMPLETE
3	4	2013-07-25 00:00:00.0	8827	CLOSED
4	5	2013-07-25 00:00:00.0	11318	COMPLETE
...	...	...	...	...
68878	68879	2014-07-09 00:00:00.0	778	COMPLETE
68879	68880	2014-07-13 00:00:00.0	1117	COMPLETE
68880	68881	2014-07-19 00:00:00.0	2518	PENDING_PAYMENT
68881	68882	2014-07-22 00:00:00.0	10000	ON_HOLD
68882	68883	2014-07-23 00:00:00.0	5533	COMPLETE

68883 rows × 4 columns

```
orders.order_date.str.slice(0, 7)
```

```
0        2013-07
1        2013-07
2        2013-07
3        2013-07
4        2013-07
          ...
68878    2014-07
68879    2014-07
68880    2014-07
68881    2014-07
68882    2014-07
Name: order_date, Length: 68883, dtype: object
```

```
orders['order_month'] = orders.order_date.str.slice(0, 7)
```

```
orders
```

	order_id	order_date	order_customer_id	order_status	order_month
0	1	2013-07-25 00:00:00.0	11599	CLOSED	2013-07
1	2	2013-07-25 00:00:00.0	256	PENDING_PAYMENT	2013-07
2	3	2013-07-25 00:00:00.0	12111	COMPLETE	2013-07
3	4	2013-07-25 00:00:00.0	8827	CLOSED	2013-07
4	5	2013-07-25 00:00:00.0	11318	COMPLETE	2013-07
...	...	...	...	...	...
68878	68879	2014-07-09 00:00:00.0	778	COMPLETE	2014-07
68879	68880	2014-07-13 00:00:00.0	1117	COMPLETE	2014-07
68880	68881	2014-07-19 00:00:00.0	2518	PENDING_PAYMENT	2014-07
68881	68882	2014-07-22 00:00:00.0	10000	ON_HOLD	2014-07
68882	68883	2014-07-23 00:00:00.0	5533	COMPLETE	2014-07

68883 rows × 5 columns

```
orders.query('order_status == "COMPLETE"'). \
    groupby('order_month')['order_id']. \
    count(). \
    sort_index()
```

```
order_month
2013-07     515
2014-07    1419
2013-10    1783
2014-06    1797
2014-05    1854
2014-02    1869
2013-08    1880
2013-12    1898
2014-01    1911
2014-04    1932
2013-09    1933
2014-03    1967
2013-11    2141
Name: order_id, dtype: int64
```

Task 3

Get order_revenue and order_quantity for each order_id. Add quantity of all items for each order_id to get order_quantity.

```
order_metrics = order_items. \
    groupby('order_item_order_id')[['order_item_subtotal', 'order_item_quantity']]. \
    agg(['sum'])
```

```
order_metrics.columns = ['order_revenue', 'order_quantity']
```

```
order_metrics
```

	order_revenue	**order_quantity**
order_item_order_id		
1	299.98	1
2	579.98	7
4	699.85	14
5	1129.86	10
7	579.92	7
...	...	...
68879	1259.97	3
68880	999.77	17
68881	129.99	1
68882	109.99	2
68883	2149.99	4

57431 rows × 2 columns