Performing Total Aggregations

Let us understand how to perform total or global aggregations using Pandas.

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

- Getting number of records in the Data Frame.

```
orders.shape
```

```
(68883, 4)
```

```
orders.shape[0]
```

```
68883
```

```
order_items.shape[0]
```

```
172198
```

- Getting number of non np.NaN values in each attribute in a Data Frame

```
orders.count()
```

```
order_id             68883
order_date           68883
order_customer_id    68883
order_status         68883
dtype: int64
```

```
type(orders.count())
```

```
pandas.core.series.Series
```

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

```
68883
```

```
orders.order_id.count()
```

```
68883
```

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

```
68883
```

- Getting basic statistics of numeric fields of a Data Frame

```
orders.describe()
```

	order_id	order_customer_id
count	68883.000000	68883.000000
mean	34442.000000	6216.571099
std	19884.953633	3586.205241
min	1.000000	1.000000
25%	17221.500000	3122.000000
50%	34442.000000	6199.000000
75%	51662.500000	9326.000000
max	68883.000000	12435.000000

- Get revenue for order id 2 from order_items

```
order_items[order_items.order_item_order_id == 2]
```

	order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order_item_
1	2	2	1073	1	
2	3	2	502	5	
3	4	2	403	1	

```
order_items[order_items.order_item_order_id == 2].order_item_subtotal
```

```
1    199.99
2    250.00
3    129.99
Name: order_item_subtotal, dtype: float64
```

```
order_items[order_items.order_item_order_id == 2].order_item_subtotal.sum()
```

```
579.98
```

```
order_items[order_items.order_item_order_id == 2]['order_item_subtotal'].sum()
```

```
579.98
```

Task 1

Use orders and get total number of records for a given month (201401).

```
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
```

```
0        2013-07-25 00:00:00.0
1        2013-07-25 00:00:00.0
2        2013-07-25 00:00:00.0
3        2013-07-25 00:00:00.0
4        2013-07-25 00:00:00.0
                ...
68878    2014-07-09 00:00:00.0
68879    2014-07-13 00:00:00.0
68880    2014-07-19 00:00:00.0
68881    2014-07-22 00:00:00.0
68882    2014-07-23 00:00:00.0
Name: order_date, Length: 68883, dtype: object
```

```
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_date'].str.slice(0, 7).str.replace('-', '').astype('int64')
```

```
0        201307
1        201307
2        201307
3        201307
4        201307
          ...
68878    201407
68879    201407
68880    201407
68881    201407
68882    201407
Name: order_date, Length: 68883, dtype: int64
```

```
orders['order_date'].str.slice(0, 7).str.replace('-', '').astype('int64') == 201401
```

```
0        False
1        False
2        False
3        False
4        False
         ...
68878    False
68879    False
68880    False
68881    False
68882    False
Name: order_date, Length: 68883, dtype: bool
```

```
orders[orders['order_date'].str.slice(0, 7).str.replace('-', '').astype('int64') == 201401]
```

	order_id	order_date	order_customer_id	order_status
25875	25876	2014-01-01 00:00:00.0	3414	PENDING_PAYMENT
25876	25877	2014-01-01 00:00:00.0	5549	PENDING_PAYMENT
25877	25878	2014-01-01 00:00:00.0	9084	PENDING
25878	25879	2014-01-01 00:00:00.0	5118	PENDING
25879	25880	2014-01-01 00:00:00.0	10146	CANCELED
...	...	...	...	...
68789	68790	2014-01-26 00:00:00.0	10302	CLOSED
68790	68791	2014-01-27 00:00:00.0	6524	COMPLETE
68791	68792	2014-01-28 00:00:00.0	9809	CANCELED
68792	68793	2014-01-30 00:00:00.0	5654	COMPLETE
68793	68794	2014-01-31 00:00:00.0	6873	COMPLETE

5908 rows × 4 columns

```
orders[orders['order_date'].str.slice(0, 7).str.replace('-', '').astype('int64') == 201401]
['order_id'].count()
```

```
5908
```

Task 2

Use order_items data set and compute total revenue generated for a given product_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.query('order_item_product_id == 502')
```

	order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order
2	3	2	502	5	
6	7	4	502	3	
19	20	8	502	1	
37	38	12	502	5	
41	42	14	502	1	
...	...	...	...	...	
172151	172152	68861	502	4	
172173	172174	68871	502	4	
172189	172190	68880	502	5	
172195	172196	68882	502	1	
172197	172198	68883	502	3	

21035 rows × 6 columns

```
order_items.query('order_item_product_id == 502')['order_item_subtotal'].sum()
```

```
3147800.0
```

```
order_items.query('order_item_product_id == 502').order_item_subtotal.sum()
```

```
3147800.0
```

Task 3

Use order_items data set and get total number of items sold as well as total revenue generated for a given product_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_for_product_id = order_items.query('order_item_product_id == 502')
order_items_for_product_id
```

	order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order
2	3	2	502	5	
6	7	4	502	3	
19	20	8	502	1	
37	38	12	502	5	
41	42	14	502	1	
...	...	...	...	...	
172151	172152	68861	502	4	
172173	172174	68871	502	4	
172189	172190	68880	502	5	
172195	172196	68882	502	1	
172197	172198	68883	502	3	

21035 rows × 6 columns

```
order_items_for_product_id[['order_item_quantity', 'order_item_subtotal']]
```

	order_item_quantity	order_item_subtotal
2	5	250.0
6	3	150.0
19	1	50.0
37	5	250.0
41	1	50.0
...	...	...
172151	4	200.0
172173	4	200.0
172189	5	250.0
172195	1	50.0
172197	3	150.0

21035 rows × 2 columns

```
order_items_for_product_id[['order_item_quantity', 'order_item_subtotal']].sum()
```

```
order_item_quantity      62956.0
order_item_subtotal    3147800.0
dtype: float64
```

```
tuple(order_items_for_product_id[['order_item_quantity', 'order_item_subtotal']].sum())
```

```
(62956.0, 3147800.0)
```

```
dict(order_items_for_product_id[['order_item_quantity', 'order_item_subtotal']].sum())
```

```
{'order_item_quantity': 62956.0, 'order_item_subtotal': 3147800.0}
```

Task 4

Create a collection with sales and commission percentage. Using that collection compute total commission amount.
If the commission percent is None or not present, treat it as 0.

- Each element in the collection should be a tuple.
- First element is the sales amount and second element is commission percentage.
- Commission for each sale can be computed by multiplying commission percentage with sales (make sure to divide commission percentage by 100).

- Some of the records does not have commission percentage, in that case commission amount for that sale shall be 0

```
transactions = [(376.0, 8),
(548.23, 14),
(107.93, 8),
(838.22, 14),
(846.85, 21),
(234.84,),
(850.2, 21),
(992.2, 21),
(267.01,),
(958.91, 21),
(412.59,),
(283.14,),
(350.01, 14),
(226.95,),
(132.7, 14)]
```

```
sales = pd.DataFrame(transactions, columns=['sale_amount', 'commission_pct'])
```

```
sales
```

	sale_amount	commission_pct
0	376.00	8.0
1	548.23	14.0
2	107.93	8.0
3	838.22	14.0
4	846.85	21.0
5	234.84	NaN
6	850.20	21.0
7	992.20	21.0
8	267.01	NaN
9	958.91	21.0
10	412.59	NaN
11	283.14	NaN
12	350.01	14.0
13	226.95	NaN
14	132.70	14.0

```
sales_filled = sales.fillna(0.0)
sales_filled
```

	sale_amount	commission_pct
0	376.00	8.0
1	548.23	14.0
2	107.93	8.0
3	838.22	14.0
4	846.85	21.0
5	234.84	0.0
6	850.20	21.0
7	992.20	21.0
8	267.01	0.0
9	958.91	21.0
10	412.59	0.0
11	283.14	0.0
12	350.01	14.0
13	226.95	0.0
14	132.70	14.0

```
(sales_filled['sale_amount'] * (sales_filled['commission_pct'] / 100))
```

```
0      30.0800
1      76.7522
2       8.6344
3     117.3508
4     177.8385
5       0.0000
6     178.5420
7     208.3620
8       0.0000
9     201.3711
10      0.0000
11      0.0000
12     49.0014
13      0.0000
14     18.5780
dtype: float64
```

```
(sales_filled['sale_amount'] * (sales_filled['commission_pct'] / 100)).sum()
```

```
1066.5104000000001
```

```
(sales_filled['sale_amount'] * (sales_filled['commission_pct'] / 100)).sum().round(2)
```

```
1066.51
```