Using groupby

Let us understand how we can use `itertools.groupby` to take care of aggregations by key.

- `itertools.groupby` can be used to get the data grouped by a key.
- It can be used to take care of use cases similar to following by using aggregate functions after grouping by key.
  - Get count by order status.
  - Get revenue for each order.
  - Get order count by month.
- We need to ensure data is pre-sorted by the key, so that all the values associated with each key are grouped together.

Print to PDF

```
import itertools as iter
```

```
iter.groupby?
```

```
Init signature: iter.groupby(self, /, *args, **kwargs)
Docstring:
groupby(iterable, key=None) -> make an iterator that returns consecutive
keys and groups from the iterable.  If the key function is not specified or
is None, the element itself is used for grouping.
Type:           type
Subclasses:
```

```
l = [1, 1, 3, 2, 1, 3, 2]
```

```
l_grouped = iter.groupby(l)
```

```
list(l_grouped)
```

```
[(1, <itertools._grouper at 0x7fdd9fb73cf8>),
 (3, <itertools._grouper at 0x7fdd41350d68>),
 (2, <itertools._grouper at 0x7fdd41350a90>),
 (1, <itertools._grouper at 0x7fdd3be07b00>),
 (3, <itertools._grouper at 0x7fdd3be07ef0>),
 (2, <itertools._grouper at 0x7fdd3be07f60>)]
```

```
l_sorted = sorted(l)
```

```
ls_grouped = iter.groupby(l_sorted)
```

```
list(ls_grouped)
```

```
[(1, <itertools._grouper at 0x7fdd3be781d0>),
 (2, <itertools._grouper at 0x7fdd3be07630>),
 (3, <itertools._grouper at 0x7fdd3be07be0>)]
```

Note

Rebuilding l_sorted and ls_grouped as ls_grouped will be flushed out after being read by `list(ls_grouped)`.

```
l_sorted = sorted(l)
```

```
ls_grouped = iter.groupby(l_sorted)
```

```
list(iter.starmap(lambda key, values: (key, len(list(values))), ls_grouped))
```

```
[(1, 3), (2, 2), (3, 2)]
```

```
%run 02_preparing_data_sets.ipynb
```

Task 1 - Order Count by Status

Get count by order status using orders data set.

```
orders[:3]
```

```
['1,2013-07-25 00:00:00.0,11599,CLOSED',
 '2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT',
 '3,2013-07-25 00:00:00.0,12111,COMPLETE']
```

```
orders_sorted = sorted(orders, key=lambda k: k.split(',')[3])
```

```
orders_sorted[:3]
```

```
['50,2013-07-25 00:00:00.0,5225,CANCELED',
 '112,2013-07-26 00:00:00.0,5375,CANCELED',
 '527,2013-07-28 00:00:00.0,5426,CANCELED']
```

```
orders_grouped = iter.groupby(orders_sorted, lambda order: order.split(',')[3])
```

```
list(orders_grouped)[:3]
```

```
[('CANCELED', <itertools._grouper at 0x7fdd3be07b70>),
 ('CLOSED', <itertools._grouper at 0x7fdd3be62be0>),
 ('COMPLETE', <itertools._grouper at 0x7fdd3a12c0f0>)]
```

```
orders_sorted = sorted(orders, key=lambda k: k.split(',')[3])
orders_grouped = iter.groupby(orders_sorted, lambda order: order.split(',')[3])
order_count_by_status = iter.starmap(lambda key, values: (key, len(list(values))),
orders_grouped)
```

```
list(order_count_by_status)
```

```
[('CANCELED', 1428),
 ('CLOSED', 7556),
 ('COMPLETE', 22899),
 ('ON_HOLD', 3798),
 ('PAYMENT_REVIEW', 729),
 ('PENDING', 7610),
 ('PENDING_PAYMENT', 15030),
 ('PROCESSING', 8275),
 ('SUSPECTED_FRAUD', 1558)]
```

Task 2 - Revenue per Order

Get revenue per order using order_items data set.

```
order_items[:4]
```

```
['1,1,957,1,299.98,299.98',
 '2,2,1073,1,199.99,199.99',
 '3,2,502,5,250.0,50.0',
 '4,2,403,1,129.99,129.99']
```

```
order_subtotals = map(lambda oi: (int(oi.split(',')[1]), float(oi.split(',')[4])), order_items)
```

```
list(order_subtotals)[:3]
```

```
[(1, 299.98), (2, 199.99), (2, 250.0)]
```

```
order_subtotals = map(lambda oi: (int(oi.split(',')[1]), float(oi.split(',')[4])), order_items)
order_subtotals_sorted = sorted(order_subtotals)
```

```python
order_subtotals_grouped = iter.groupby(order_subtotals_sorted, lambda rec: rec[0])
```

```python
list(order_subtotals_grouped)[:3]
```

```
[(1, <itertools._grouper at 0x7fdd3be62da0>),
 (2, <itertools._grouper at 0x7fdd38d9c208>),
 (4, <itertools._grouper at 0x7fdd38d9c390>)]
```

```python
order_subtotals = map(lambda oi: (int(oi.split(',')[1]), float(oi.split(',')[4])), order_items)
order_subtotals_sorted = sorted(order_subtotals)

order_subtotals_grouped = iter.groupby(order_subtotals_sorted, lambda rec: rec[0])

item = list(order_subtotals_grouped)[0]
```

```python
print(item[1]) # Contains similar to this [(2, 199.99), (2, 250.0), (2, 129.99)]
```

```
<itertools._grouper object at 0x7fdd40adb048>
```

```python
i = [(2, 199.99), (2, 250.0), (2, 129.99)]
```

```python
list(map(lambda rec: rec[1], i))
```

```
[199.99, 250.0, 129.99]
```

```python
sum(list(map(lambda rec: rec[1], i))) # this will go as part of first argument to starmap
```

```
579.98
```

```python
order_subtotals = map(lambda oi: (int(oi.split(',')[1]), float(oi.split(',')[4])), order_items)
order_subtotals_sorted = sorted(order_subtotals)

order_subtotals_grouped = iter.groupby(order_subtotals_sorted, lambda rec: rec[0])

order_revenue = iter.starmap(
    lambda key, values: (key, round(sum(list(map(lambda rec: rec[1], values))), 2)),
    order_subtotals_grouped
)
```

```python
list(order_revenue)[:3]
```

```
[(1, 299.98), (2, 579.98), (4, 699.85)]
```

Note

Alternative solution by avoiding first map.

```python
order_items_sorted = sorted(order_items, key=lambda oi: int(oi.split(',')[1]))

order_items_grouped = iter.groupby(order_items_sorted, lambda oi: int(oi.split(',')[1]))
```

```python
order_items[1:4]
```

```
['2,2,1073,1,199.99,199.99', '3,2,502,5,250.0,50.0', '4,2,403,1,129.99,129.99']
```

```python
values = order_items[1:4]
```

```python
list(map(lambda rec: float(rec.split(',')[4]), values))
```

```
[199.99, 250.0, 129.99]
```

```python
sum(list(map(lambda rec: float(rec.split(',')[4]), values)))
```

```
579.98
```

```
order_revenue = iter.starmap(
    lambda key, values: (key, round(sum(list(map(lambda rec: float(rec.split(',')[4]),
values))), 2)),
    order_items_grouped
)
```

```
list(order_revenue)[:3]
```

```
[(1, 299.98), (2, 579.98), (4, 699.85)]
```