In [1]:
```
import pandas as pd
import numpy as np

#load file
sales = pd.read_csv('sales.csv')
prices = pd.read_csv('prices.csv')
```

In [2]:
```
sales['ordered_at'] = pd.to_datetime(sales['ordered_at'])
prices['updated_at'] = pd.to_datetime(prices['updated_at'])
```

In [13]:
```
sales = sales.sort_values(by='ordered_at')
sales
```

Out[13]:

	product_id	ordered_at	quantity_ordered	price	revenue
86	3954203	2018-09-11 01:43:00	1	64000	64000
28	4085861	2018-09-11 06:26:00	1	60000	60000
26	4085861	2018-09-11 06:53:00	1	60000	60000
27	4085861	2018-09-11 08:24:00	1	60000	60000
123	4085861	2018-09-11 09:30:00	1	53500	53500
...	...	...	...	...	...
67	4085861	2018-09-18 20:23:00	1	53500	53500
77	4085861	2018-09-18 20:43:00	1	53500	53500
79	4085861	2018-09-18 20:54:00	1	53500	53500
87	3954203	2018-09-18 21:26:00	1	57500	57500
11	3998909	2018-09-18 22:11:00	1	16500	16500

175 rows × 5 columns

In [14]:
```
prices = prices.sort_values(by='updated_at')
prices
```

Out[14]:

	product_id	old_price	new_price	updated_at
4	3954203	68800	60000	2018-09-10 16:32:00
7	3998909	19000	17000	2018-09-10 16:35:00
0	64	270000	239000	2018-09-10 16:37:00
11	4085861	60000	53500	2018-09-11 08:51:00
1	3954203	60000	64000	2018-09-11 11:54:00
9	4085861	53500	67000	2018-09-12 03:51:00
6	3998909	17000	15500	2018-09-13 06:43:00
13	4085861	67000	62500	2018-09-13 06:43:00
3	3954203	64000	60500	2018-09-15 03:49:00
12	4085861	62500	58000	2018-09-15 03:51:00
5	3998909	15500	16500	2018-09-16 05:09:00
10	4085861	58000	53500	2018-09-17 03:35:00

2	3954203	60500	57500	2018-09-17 22:59:00
8	4085861	53500	52000	2018-09-17 22:59:00

1.1 With direction='nearest'

In [15]:
```
merge_nearest = pd.merge_asof(sales, prices, left_on='ordered_at', right_on='updated_at'
```

In [16]:
```
merge_nearest['listed_price'] = np.where(merge_nearest['ordered_at'] >= merge_nearest['u
                                         merge_nearest['new_price'], merge_nearest['old_
merge_nearest.head(10)
```

Out[16]:

	product_id	ordered_at	quantity_ordered	price	revenue	old_price	new_price	updated_at	listed_price
0	3954203	2018-09-11 01:43:00	1	64000	64000	68800	60000	2018-09-10 16:32:00	60000
1	4085861	2018-09-11 06:26:00	1	60000	60000	60000	53500	2018-09-11 08:51:00	60000
2	4085861	2018-09-11 06:53:00	1	60000	60000	60000	53500	2018-09-11 08:51:00	60000
3	4085861	2018-09-11 08:24:00	1	60000	60000	60000	53500	2018-09-11 08:51:00	60000
4	4085861	2018-09-11 09:30:00	1	53500	53500	60000	53500	2018-09-11 08:51:00	53500
5	4085861	2018-09-11 11:06:00	1	60000	60000	60000	53500	2018-09-11 08:51:00	53500
6	3954203	2018-09-11 11:11:00	1	60000	60000	60000	64000	2018-09-11 11:54:00	60000
7	3954203	2018-09-11 11:11:00	1	60000	60000	60000	64000	2018-09-11 11:54:00	60000
8	4085861	2018-09-11 11:34:00	1	60000	60000	60000	53500	2018-09-11 08:51:00	53500
9	4085861	2018-09-11 11:47:00	2	60000	120000	60000	53500	2018-09-11 08:51:00	53500

In [17]:
```
merge_nearest['revenue'] = merge_nearest['quantity_ordered'] * merge_nearest['listed_pri
total_revenue_by_product_and_price = merge_nearest.groupby(['product_id', 'listed_price'
total_revenue_by_product_and_price
```

Out[17]:

	product_id	listed_price	revenue
0	64	239000	956000
1	3954203	57500	57500
2	3954203	60000	180000
3	3954203	64000	640000
4	3998909	15500	15500
5	3998909	16500	231000
6	3998909	17000	34000
7	4085861	52000	1040000
8	4085861	53500	2140000

9	4085861	58000	2204000
10	4085861	60000	180000
11	4085861	62500	1812500
12	4085861	67000	871000

In [18]:
```
total_revenue = merge_nearest.groupby('product_id')['revenue'].sum()
total_revenue
```

Out[18]:
```
product_id
64           956000
3954203      877500
3998909      280500
4085861     8247500
Name: revenue, dtype: int64
```

1.2. With direction='backward' (combining with direction='forward')

In [28]:
```
merge_backward = pd.merge_asof(sales, prices, by='product_id', left_on='ordered_at', rig
merge_backward = merge_backward[['product_id', 'ordered_at', 'quantity_ordered', 'new_pr
merge_backward.head(10)
```

Out[28]:

	product_id	ordered_at	quantity_ordered	price
0	3954203	2018-09-11 01:43:00	1	60000.0
1	4085861	2018-09-11 06:26:00	1	NaN
2	4085861	2018-09-11 06:53:00	1	NaN
3	4085861	2018-09-11 08:24:00	1	NaN
4	4085861	2018-09-11 09:30:00	1	53500.0
5	4085861	2018-09-11 11:06:00	1	53500.0
6	3954203	2018-09-11 11:11:00	1	60000.0
7	3954203	2018-09-11 11:11:00	1	60000.0
8	4085861	2018-09-11 11:34:00	1	53500.0
9	4085861	2018-09-11 11:47:00	2	53500.0

In [29]:
```
merge_forward = pd.merge_asof(merge_backward, prices, by='product_id', left_on='ordered_
merge_forward.head()
```

Out[29]:

	product_id	ordered_at	quantity_ordered	price	old_price	new_price	updated_at
0	3954203	2018-09-11 01:43:00	1	60000.0	60000.0	64000.0	2018-09-11 11:54:00
1	4085861	2018-09-11 06:26:00	1	NaN	60000.0	53500.0	2018-09-11 08:51:00
2	4085861	2018-09-11 06:53:00	1	NaN	60000.0	53500.0	2018-09-11 08:51:00
3	4085861	2018-09-11 08:24:00	1	NaN	60000.0	53500.0	2018-09-11 08:51:00
4	4085861	2018-09-11 09:30:00	1	53500.0	53500.0	67000.0	2018-09-12 03:51:00

In [30]:
```
merge_forward['price'] = merge_forward['price'].fillna(merge_forward['old_price'])
final_data = merge_forward[['product_id', 'ordered_at', 'quantity_ordered', 'price']]
final_data
```

Out[30]:

	product_id	ordered_at	quantity_ordered	price
0	3954203	2018-09-11 01:43:00	1	60000.0
1	4085861	2018-09-11 06:26:00	1	60000.0
2	4085861	2018-09-11 06:53:00	1	60000.0
3	4085861	2018-09-11 08:24:00	1	60000.0
4	4085861	2018-09-11 09:30:00	1	53500.0
...	...	...	...	...
170	4085861	2018-09-18 20:23:00	1	52000.0
171	4085861	2018-09-18 20:43:00	1	52000.0
172	4085861	2018-09-18 20:54:00	1	52000.0
173	3954203	2018-09-18 21:26:00	1	57500.0
174	3998909	2018-09-18 22:11:00	1	16500.0

175 rows × 4 columns

In [31]:
```
final_data['revenue'] = final_data['quantity_ordered'] * final_data['price']
revenue_by_product_and_price = final_data.groupby(['product_id', 'price'], as_index=Fals
revenue_by_product_and_price
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_9020\2432931725.py:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  final_data['revenue'] = final_data['quantity_ordered'] * final_data['price']
```

Out[31]:

	product_id	price	revenue
0	64	239000.0	956000.0
1	3954203	57500.0	57500.0
2	3954203	60000.0	180000.0
3	3954203	64000.0	640000.0
4	3998909	15500.0	15500.0
5	3998909	16500.0	231000.0
6	3998909	17000.0	34000.0
7	4085861	52000.0	1040000.0
8	4085861	53500.0	2140000.0
9	4085861	58000.0	2204000.0
10	4085861	60000.0	180000.0
11	4085861	62500.0	1812500.0
12	4085861	67000.0	871000.0

In [32]:
```
total_revenue = final_data.groupby('product_id', as_index=False)['revenue'].sum()
total_revenue
```

Out[32]:

	product_id	revenue
0	64	956000.0
1	3954203	877500.0
2	3998909	280500.0
3	4085861	8247500.0