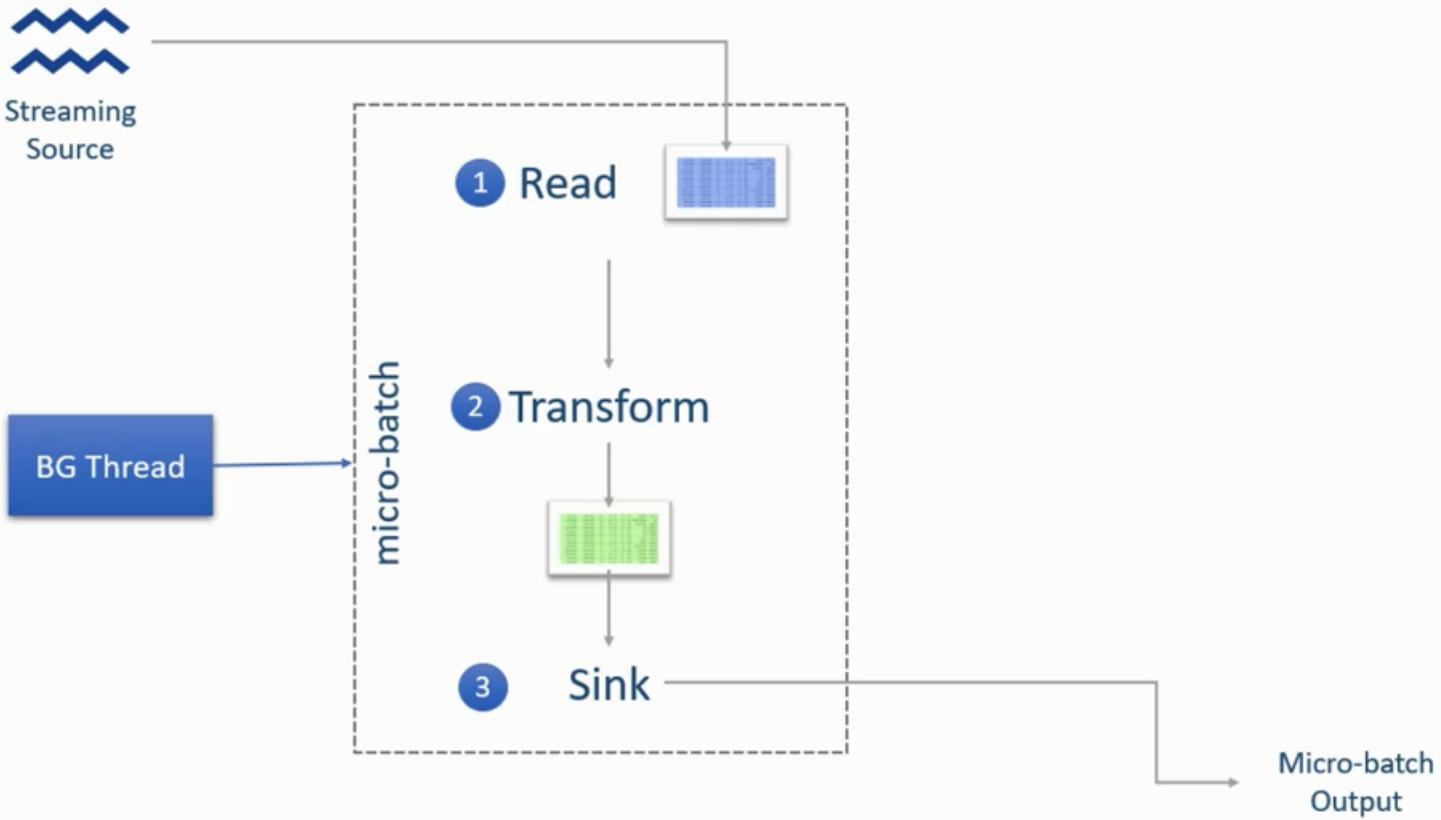
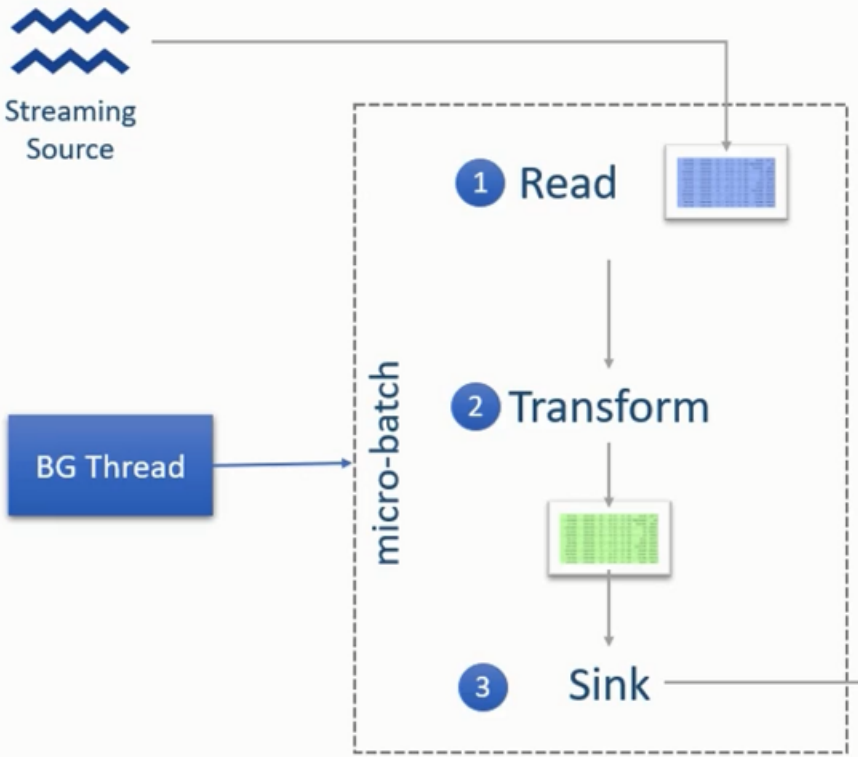
Introduction to windowing in the world of Streaming

By Apache Spark

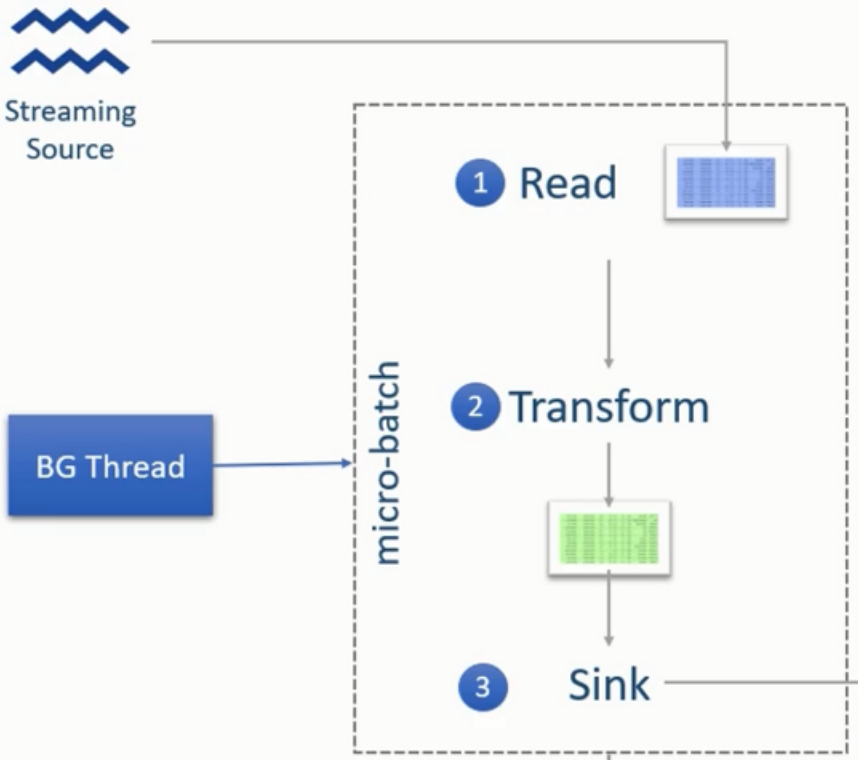
1. Stateless transformations
2. Stateful transformations

What is the state?


Streaming Source
1 Read
micro-batch
2 Transform
BG Thread
3 Sink
Micro-batch Output

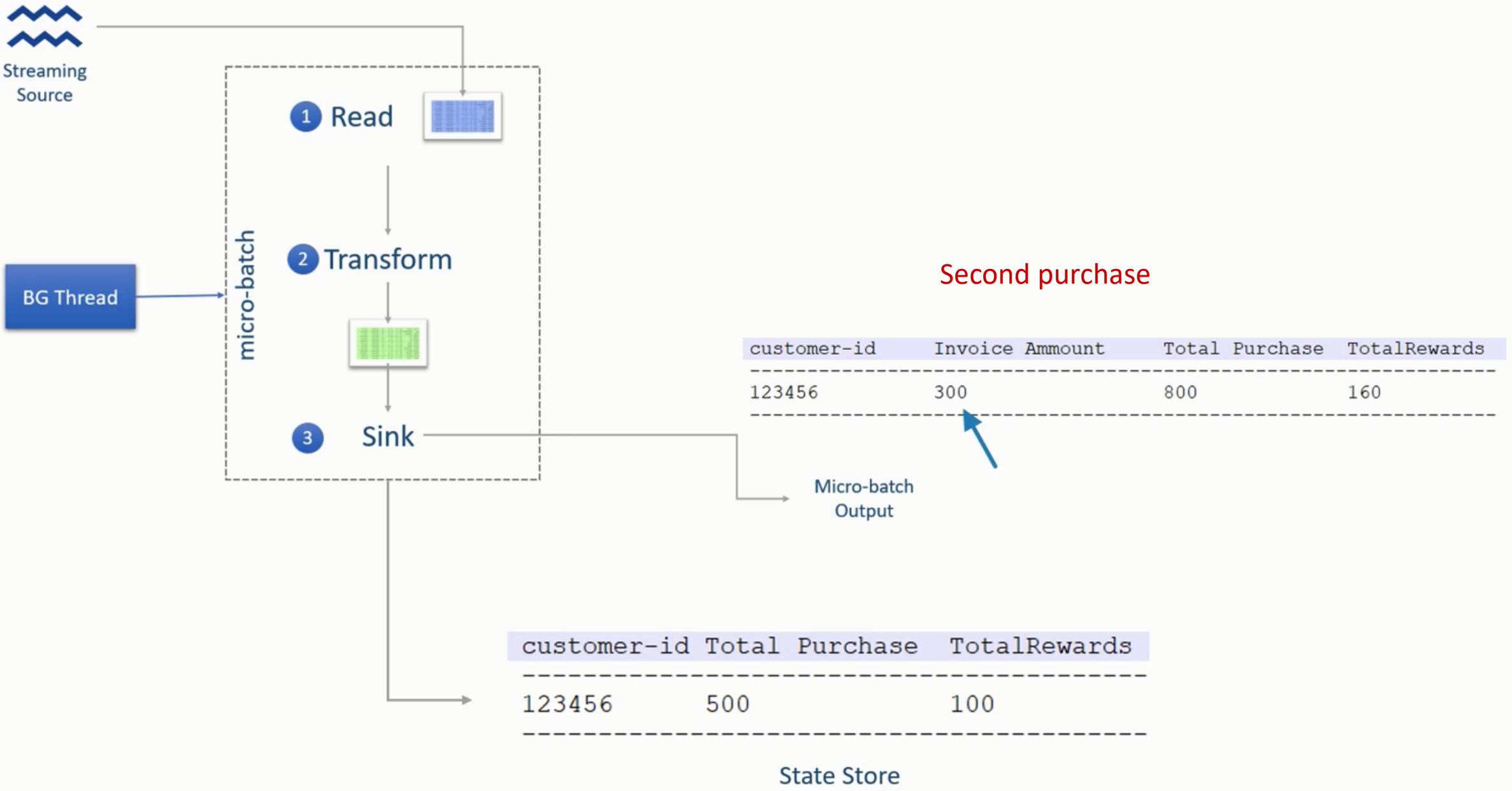


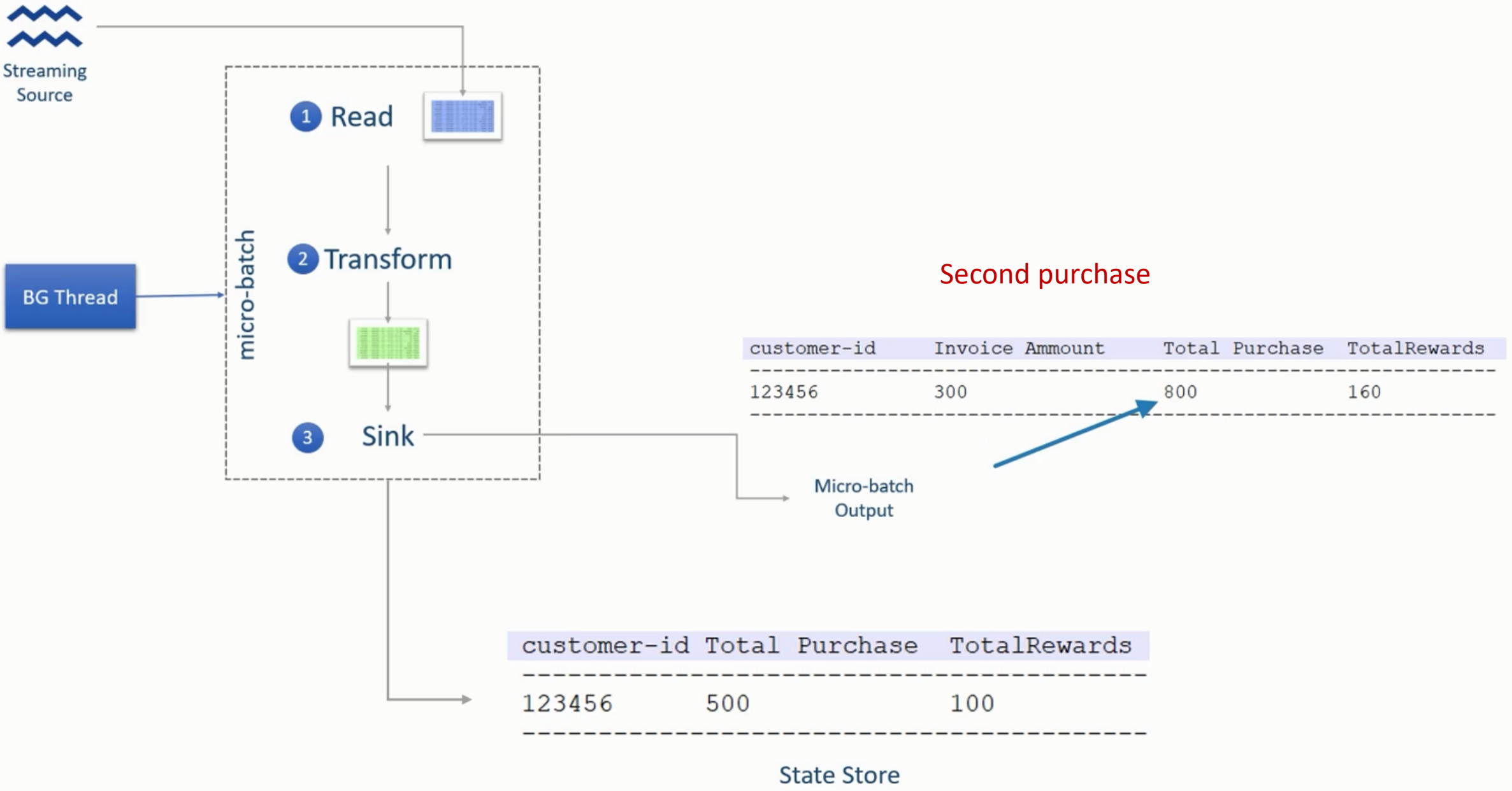
customer-id	Invoice Ammount	Total Purchase	TotalRewards
123456	500	500	100

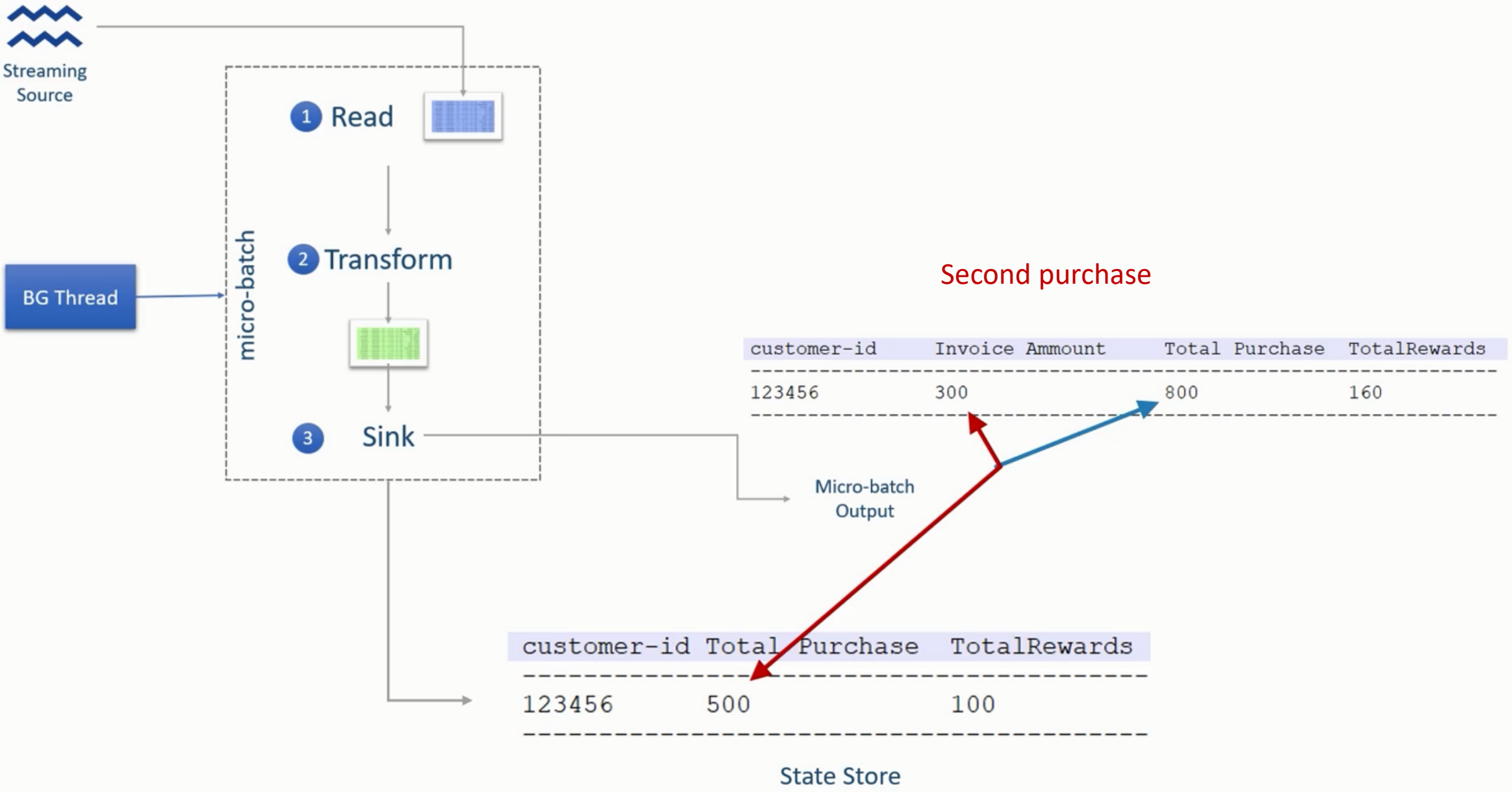


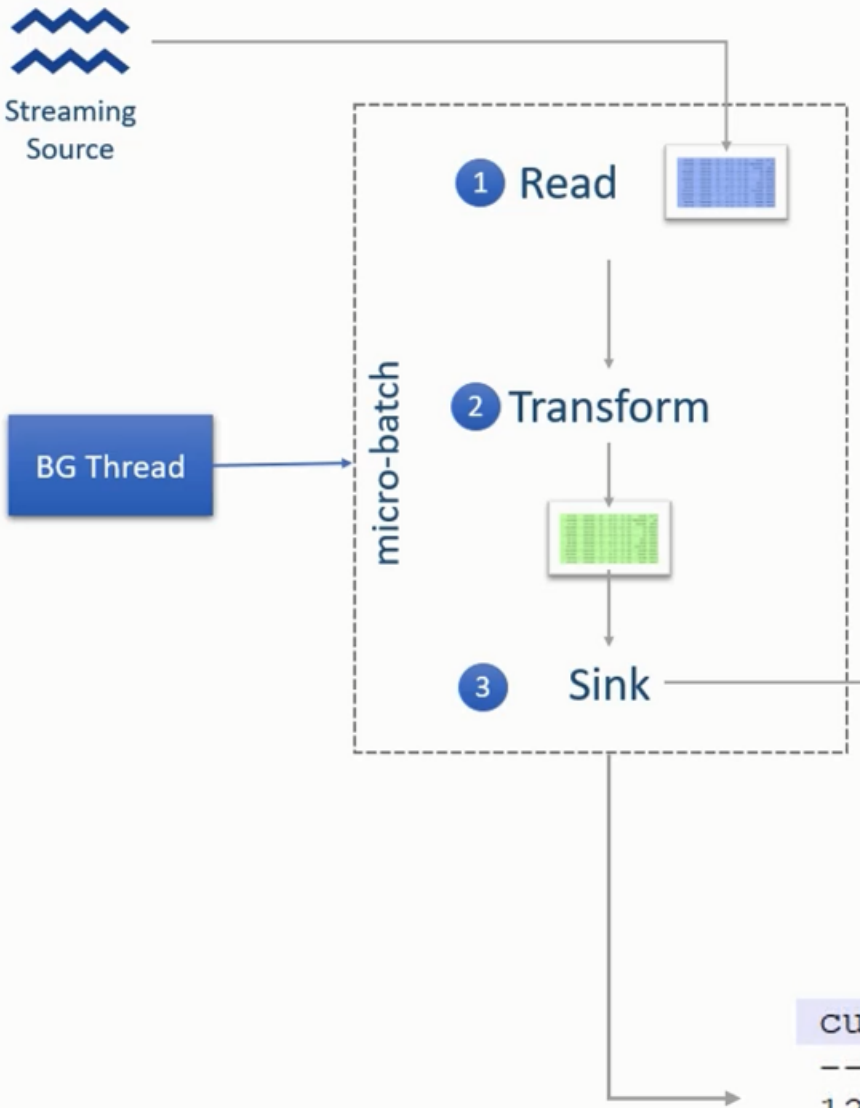
First purchase

customer-id	Invoice Ammount	Total Purchase	TotalRewards
123456	500	500	100


Streaming Source
1 Read
2 Transform
3 Sink
micro-batch
BG Thread
Second purchase
customer-id Invoice Ammount Total Purchase TotalRewards
123456 300 800 160
Micro-batch Output
customer-id Total Purchase TotalRewards
123456 500 100
State Store


Streaming Source
1 Read
2 Transform
3 Sink
micro-batch
BG Thread
Second purchase
customer-id Invoice Ammount Total Purchase TotalRewards
123456 300 800 160
Micro-batch Output
customer-id Total Purchase TotalRewards
123456 500 100
State Store


Streaming Source
1 Read
2 Transform
3 Sink
micro-batch
BG Thread
Second purchase
customer-id Invoice Ammount Total Purchase TotalRewards
123456 300 800 160
Micro-batch Output
customer-id Total Purchase TotalRewards
123456 500 100
State Store



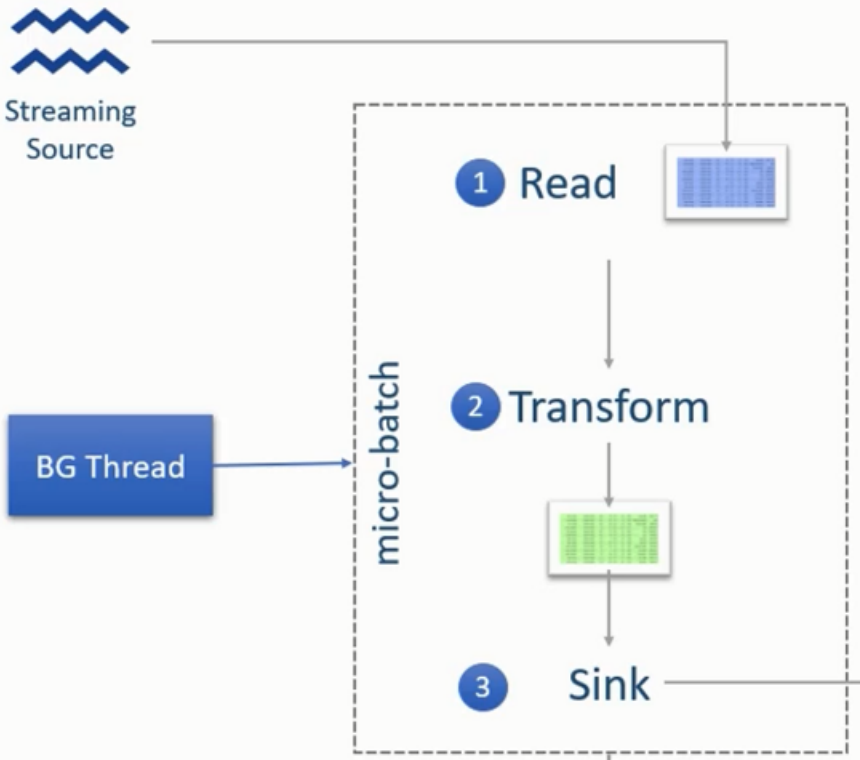
Second purchase

customer-id	Invoice Ammount	Total Purchase	TotalRewards
123456	300	800	160

Now the Store is updated

customer-id	Total Purchase	TotalRewards
123456	800	160

State Store



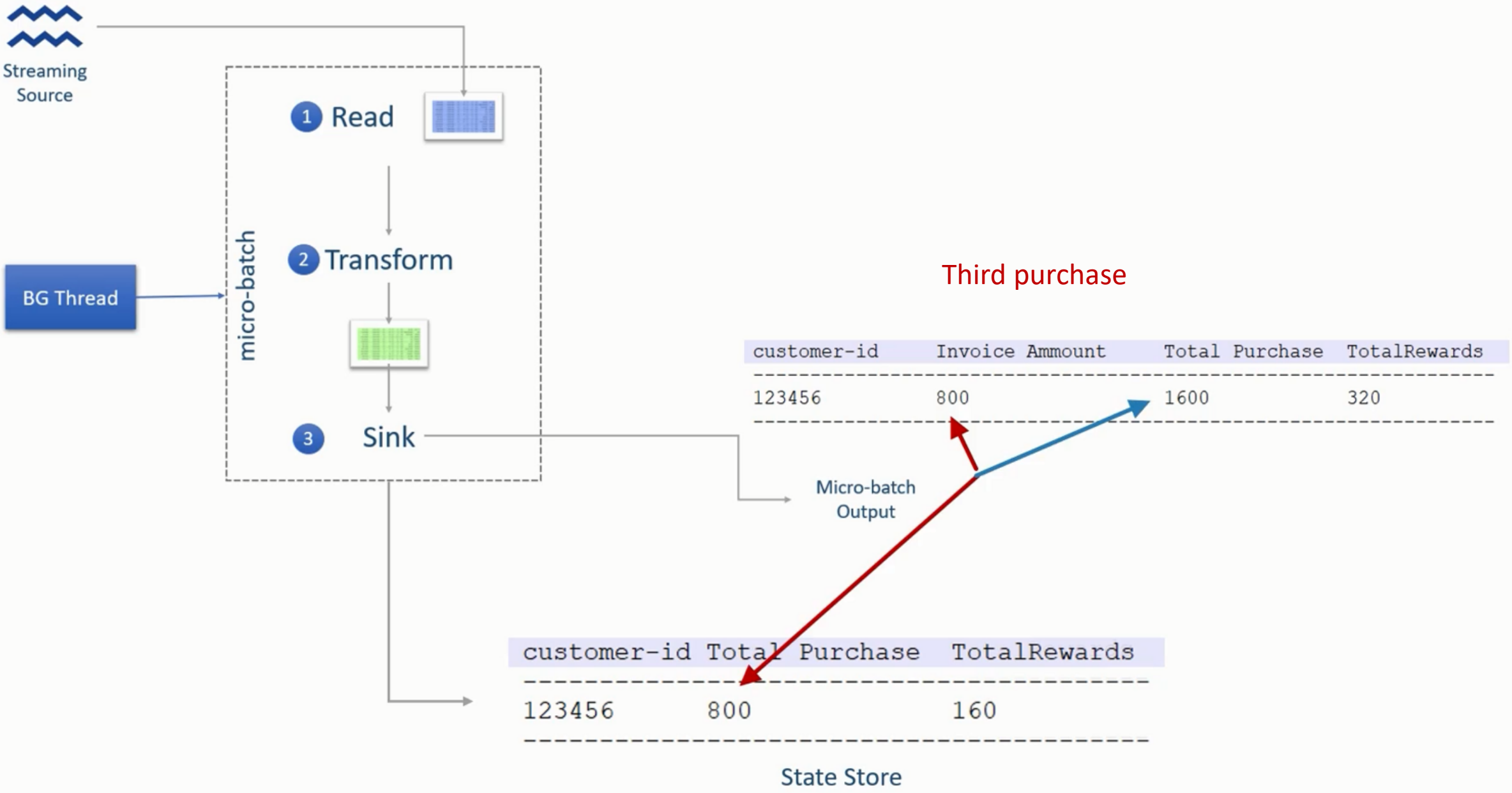
Third purchase

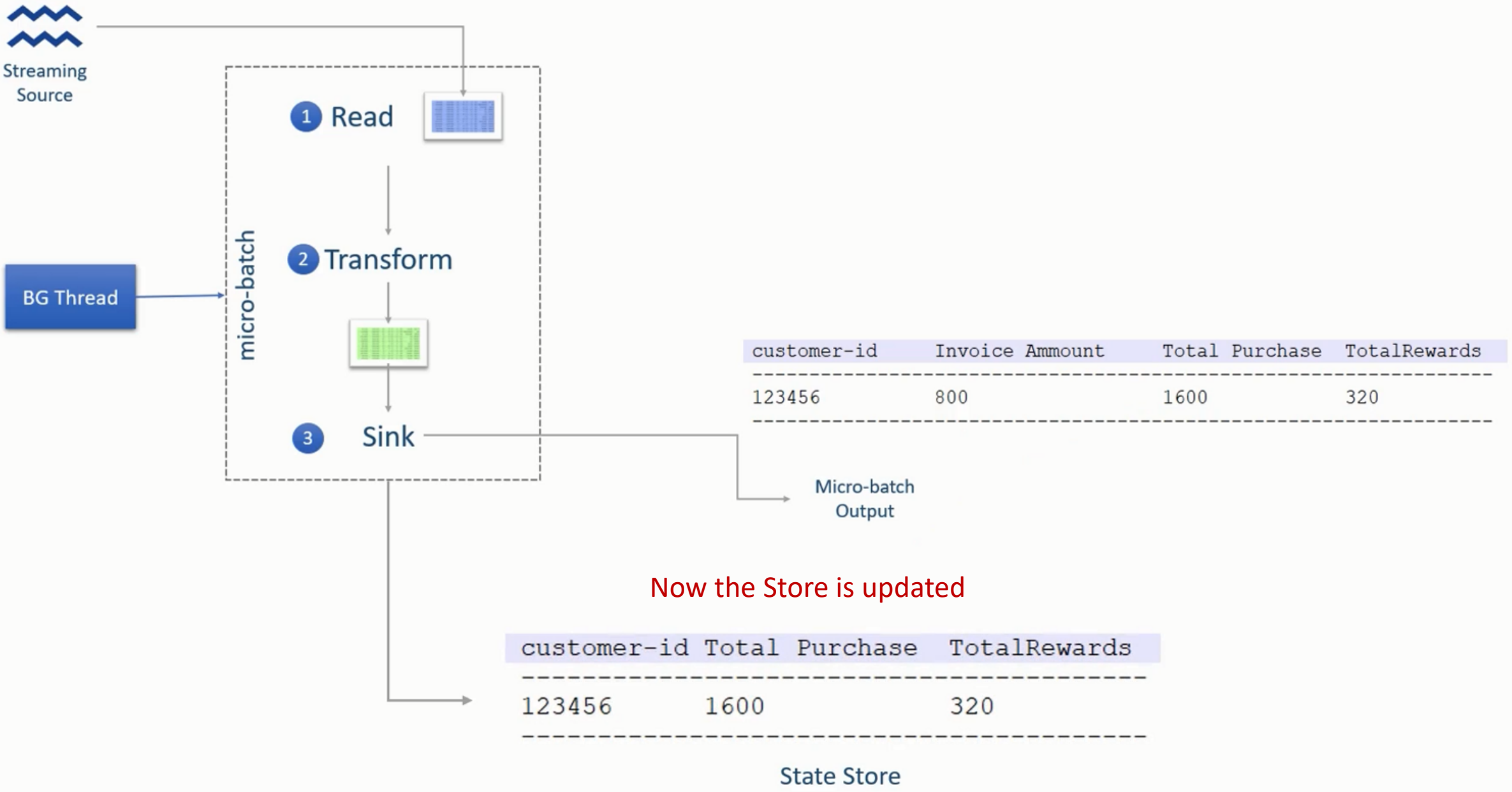
customer-id	Invoice Ammount	Total Purchase	TotalRewards
123456	800	1600	320

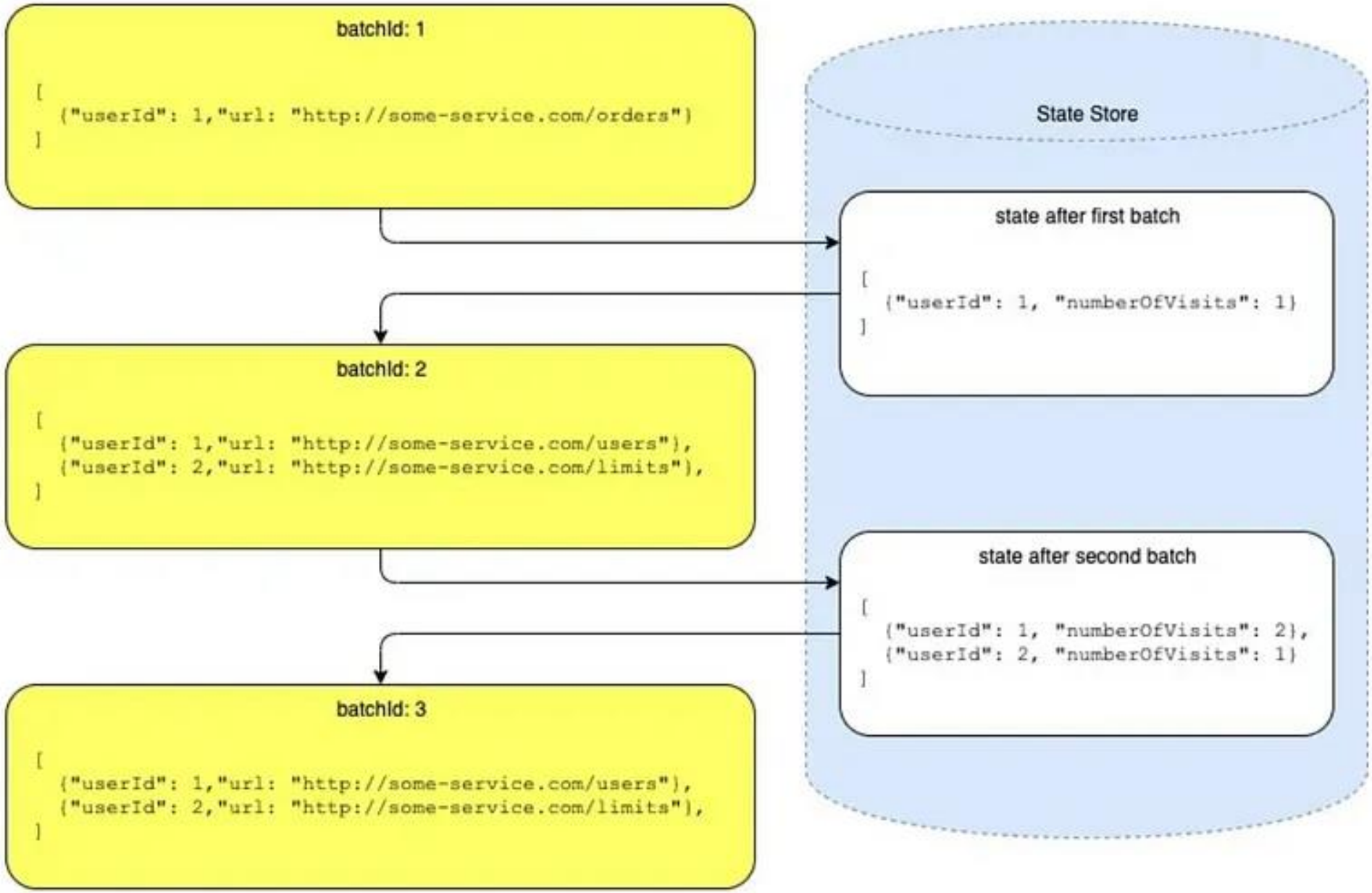
Micro-batch Output

customer-id	Total Purchase	TotalRewards
123456	800	160

State Store


Streaming Source
1 Read
2 Transform
3 Sink
micro-batch
BG Thread
Third purchase
customer-id Invoice Ammount Total Purchase TotalRewards
123456 800 1600 320
Micro-batch Output
customer-id Total Purchase TotalRewards
123456 800 160
State Store


Streaming Source
1 Read
micro-batch
2 Transform
BG Thread
3 Sink
customer-id Invoice Ammount Total Purchase TotalRewards
123456 800 1600 320
Micro-batch Output
Now the Store is updated
customer-id Total Purchase TotalRewards
123456 1600 320
State Store



State store and Spark Structured Streaming process schema

1. Stateless transformation

   select(), filter(), map(),
   flatMap(), explode()

2. Stateful transformations

   Grouping, Aggregations
   Windowing, and Joins

First we should reflect on these two categories.

1. Stateless transformation
   Complete output mode not supported
2. Stateful transformations
   Excessive state causes out of memory

Only supports "**append**" and "**update**" mode!

1. Stateless transformation
   Complete output mode not supported
2. Stateful transformations
   Excessive state causes out of memory

1. Stateless transformation

   Complete output mode not supported

2. Stateful transformations

   Excessive state causes out of memory

10 Records → SELECT → 10 Records

When to use of **Managed** and
when use **Unmanaged?**

Stateful transformations

- Managed Stateful operations
- Unmanaged Stateful operations

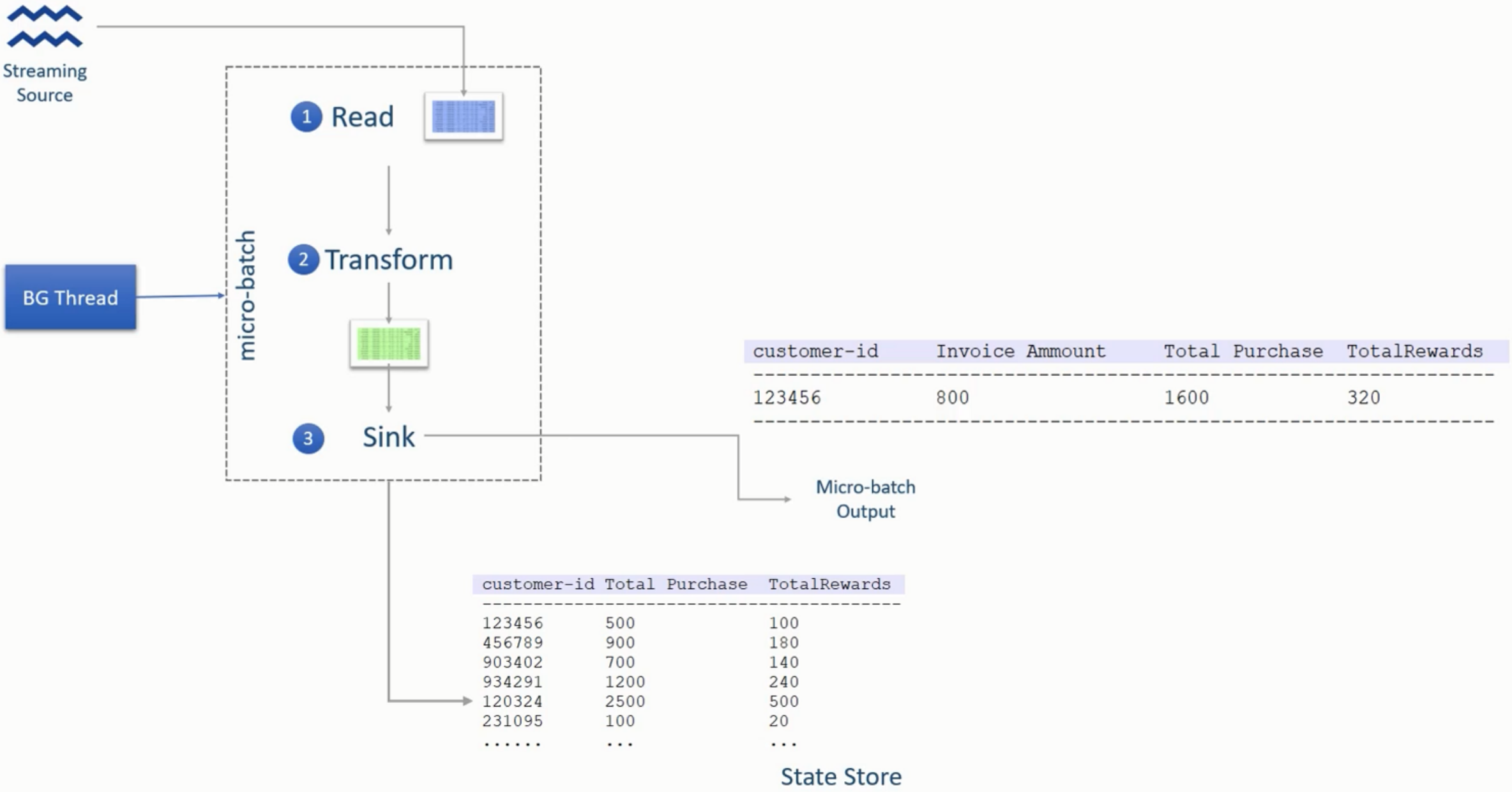
Managed operation will be handle by Spark.

Unmanaged will be managed by user.

In the sequel, we just focus on Managed stateful operations.

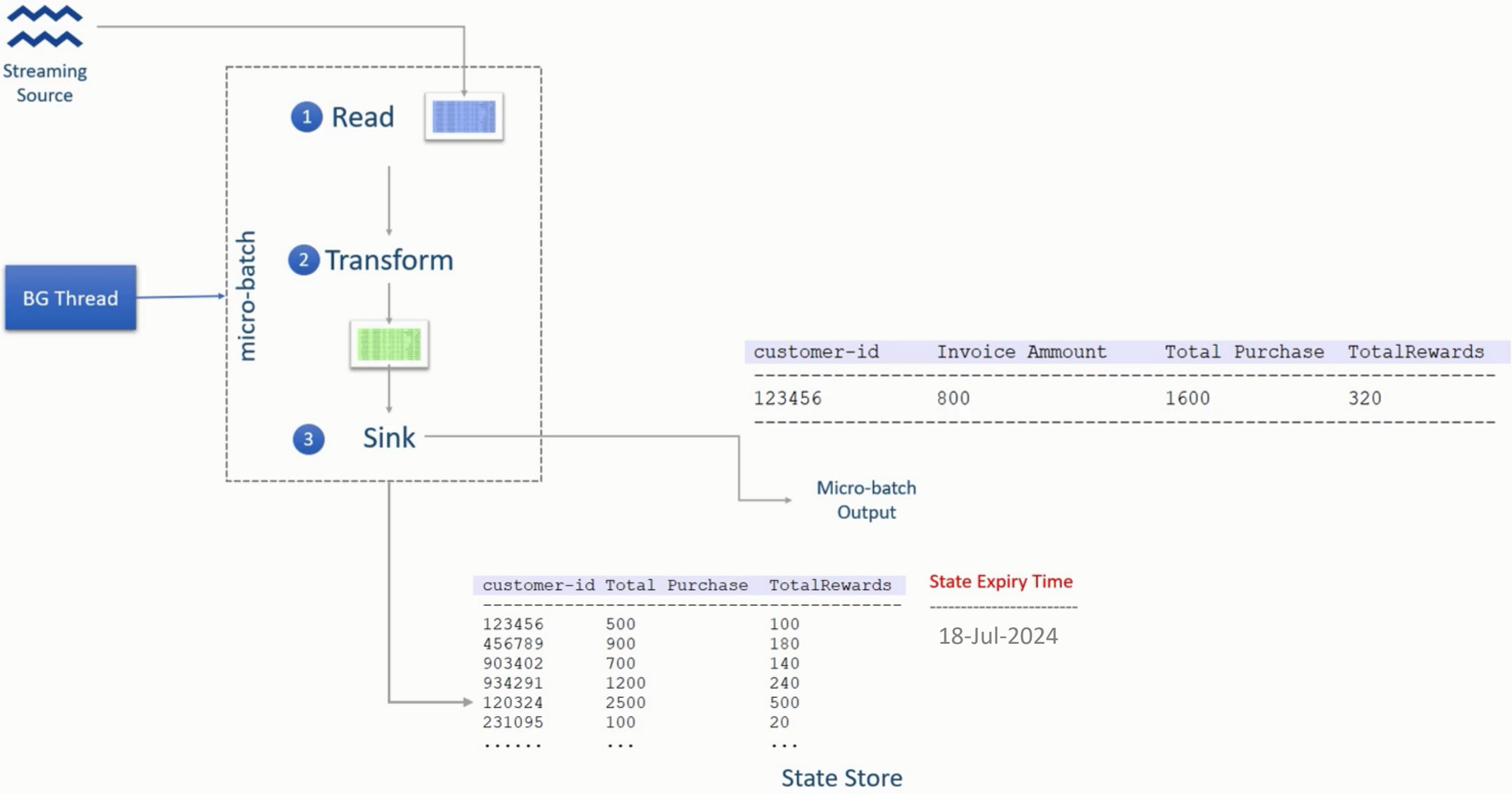
Aggregations

- Continuous aggregation
- Time-bound aggregations



customer-id	Invoice Ammount	Total Purchase	TotalRewards
123456	800	1600	320

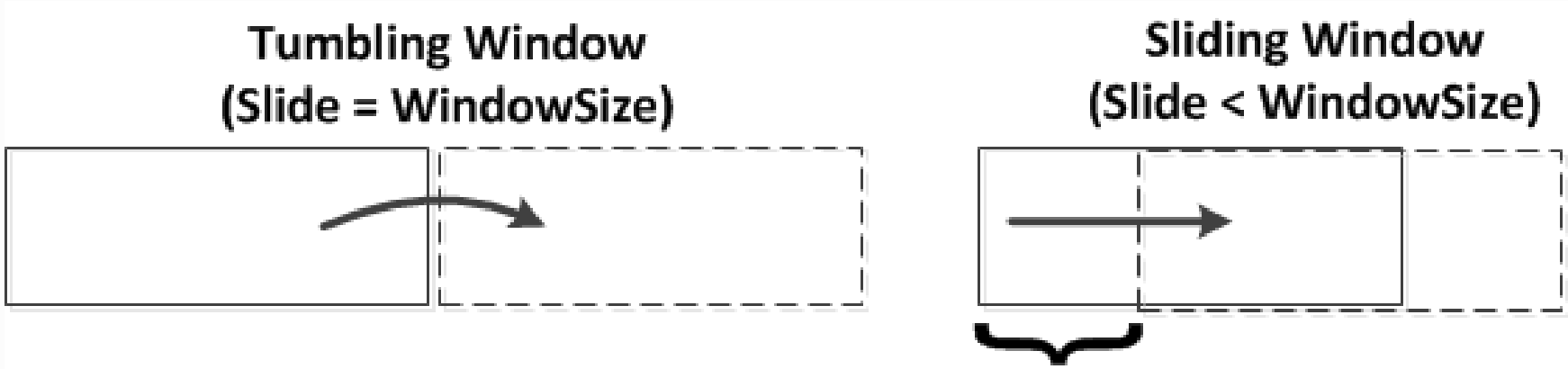
customer-id	Total Purchase	TotalRewards
123456	500	100
456789	900	180
903402	700	140
934291	1200	240
120324	2500	500
231095	100	20
.....	...	...

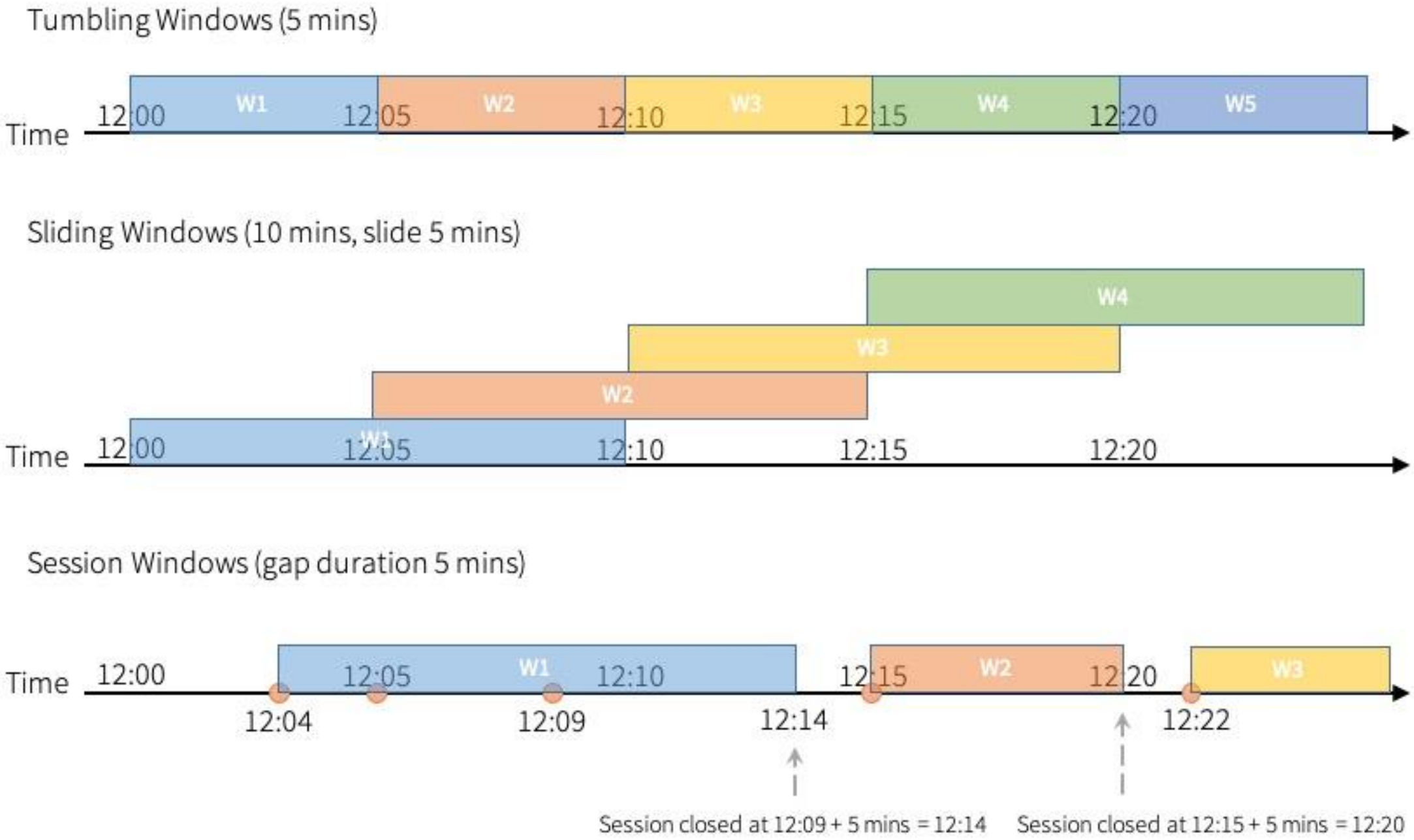

Streaming Source
1 Read
micro-batch
2 Transform
BG Thread
3 Sink
customer-id Invoice Ammount Total Purchase TotalRewards
123456 800 1600 320
Micro-batch Output
customer-id Total Purchase TotalRewards
123456 500 100
456789 900 180
903402 700 140
934291 1200 240
120324 2500 500
231095 100 20
State Expiry Time
18-Jul-2024
State Store

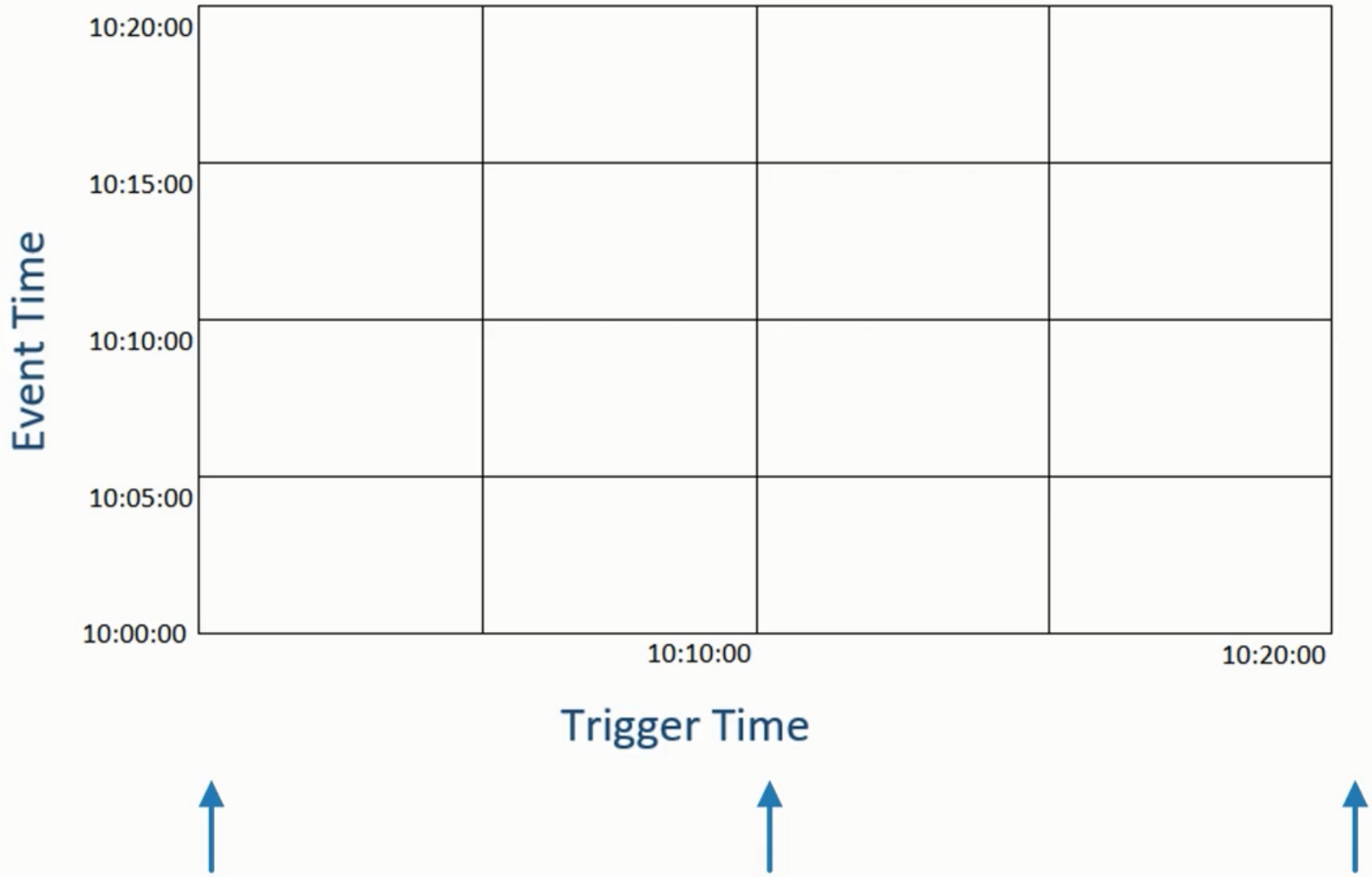
Windowing

Windowing Aggregates

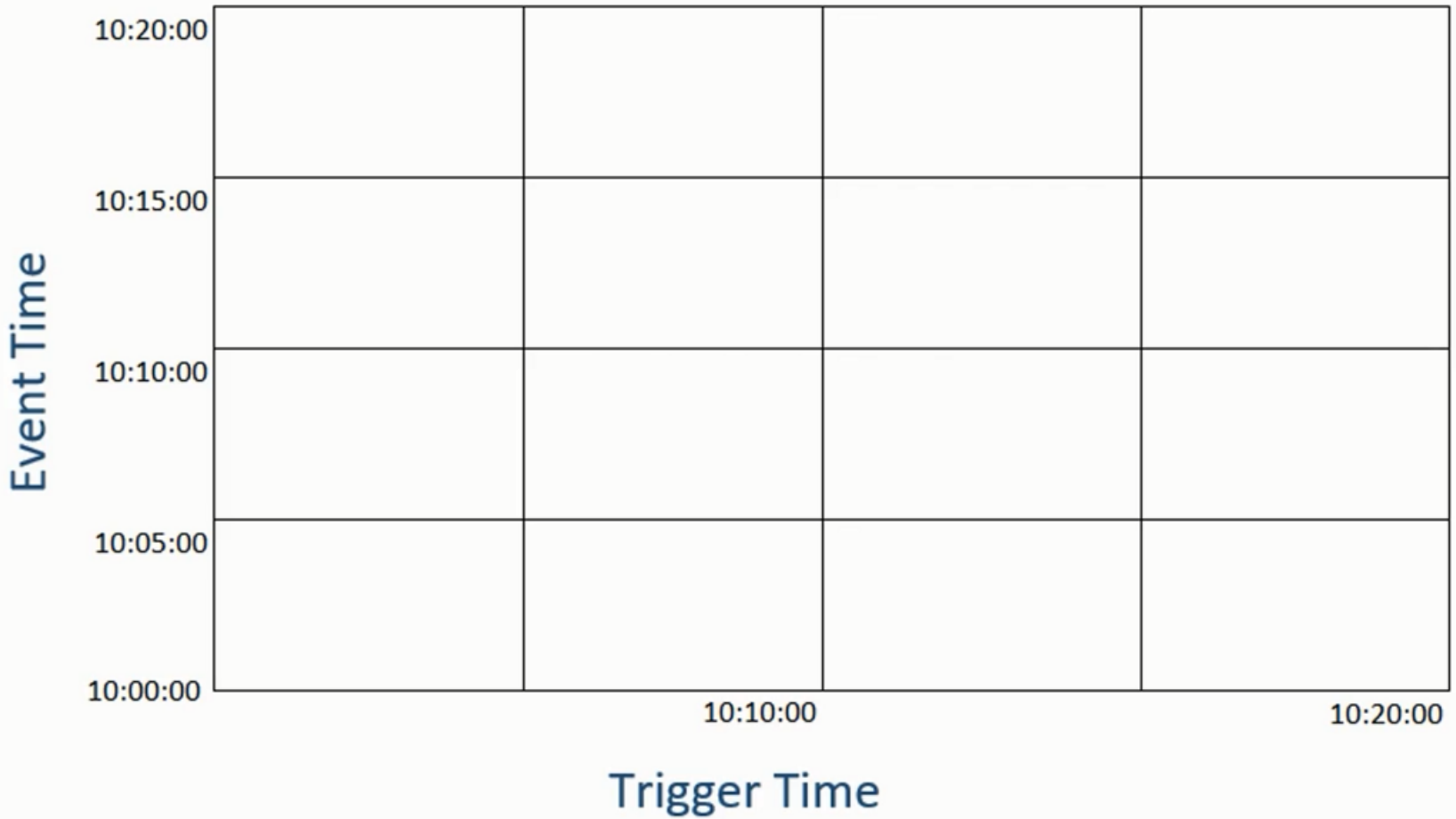
1. Tumbling Time Window
2. Sliding Time Window
3. Session Window



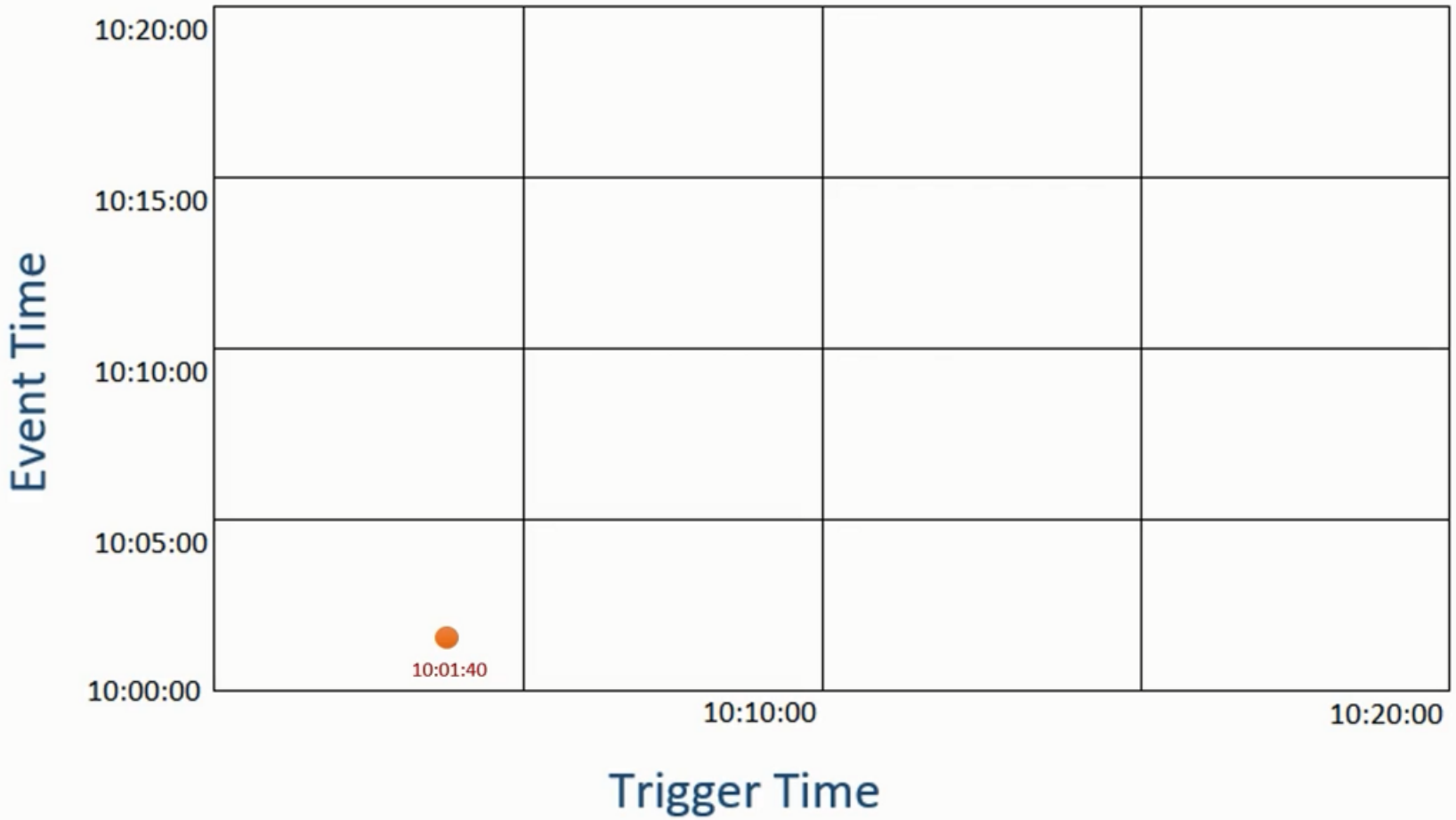

Tumbling Windows (5 mins)
Time
12:00
W1
12:05
W2
12:10
W3
12:15
W4
12:20
W5
Sliding Windows (10 mins, slide 5 mins)
W4
W3
W2
W1
Time
12:00
12:05
12:10
12:15
12:20
Session Windows (gap duration 5 mins)
Time
12:00
12:05
W1
12:10
12:15
W2
12:20
W3
12:04
12:09
12:14
12:22
Session closed at 12:09 + 5 mins = 12:14
Session closed at 12:15 + 5 mins = 12:20

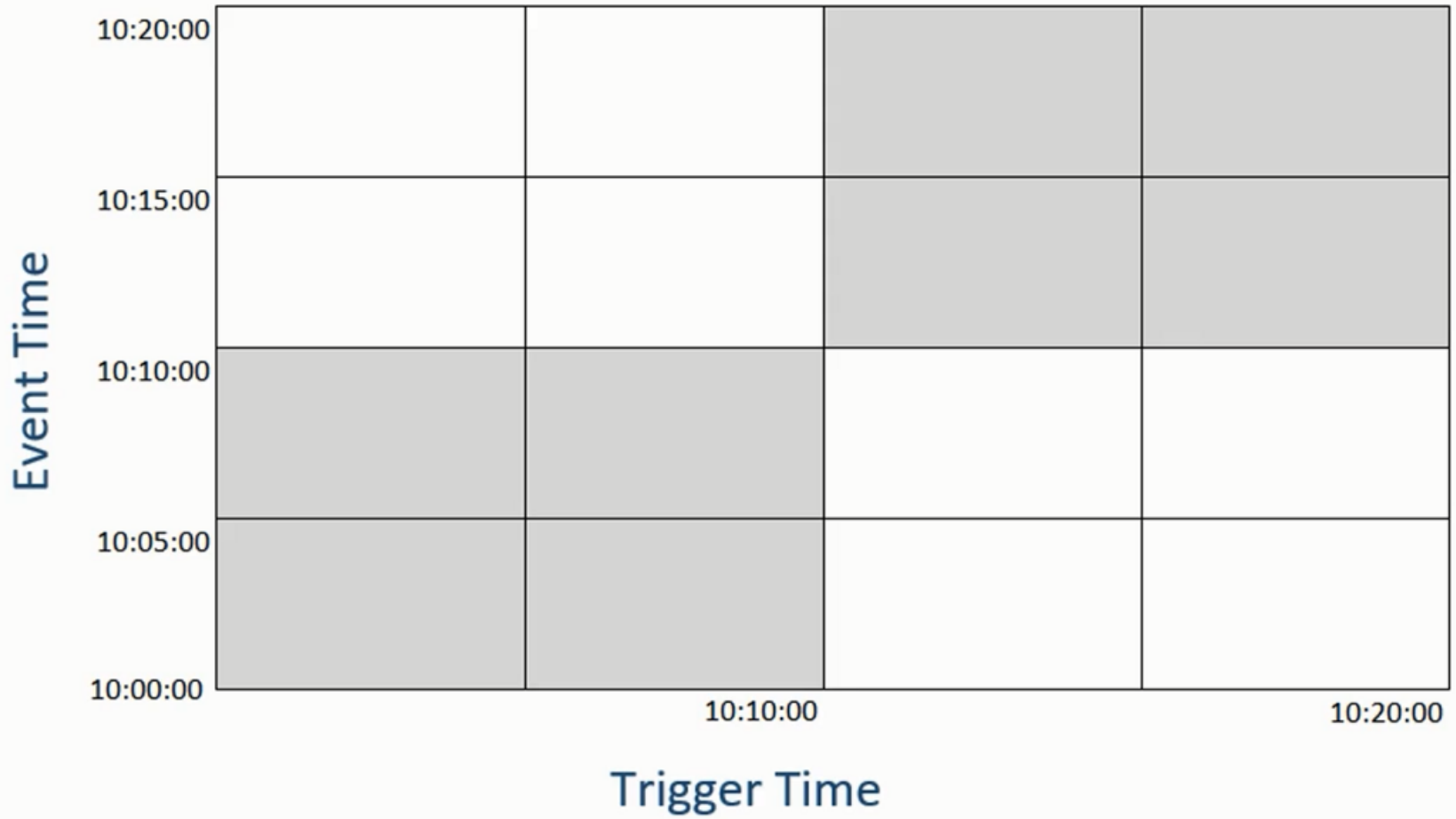

10:20:00
10:15:00
10:10:00
10:05:00
10:00:00
Event Time
10:10:00
10:20:00
Trigger Time

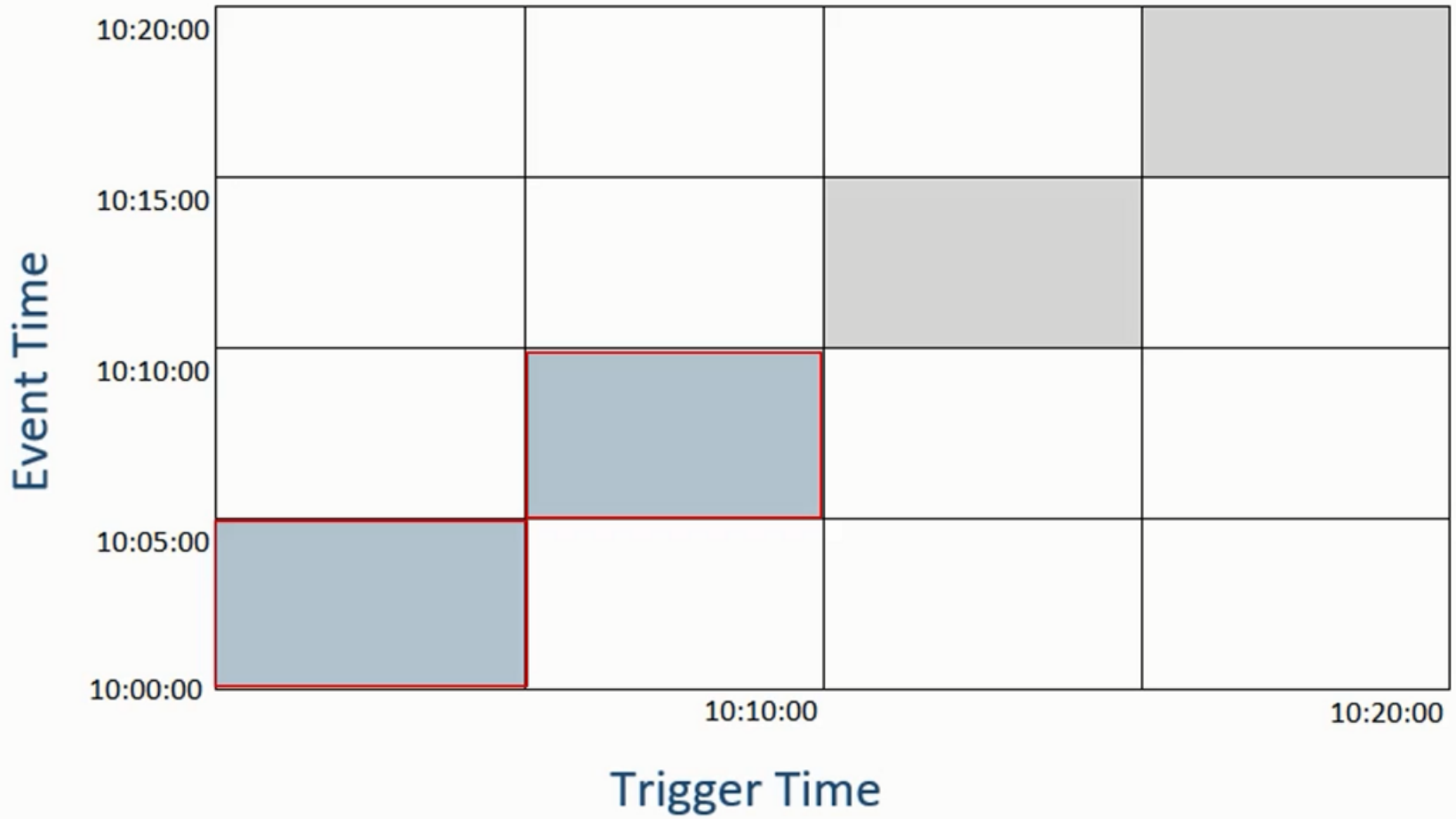
When our record is generated

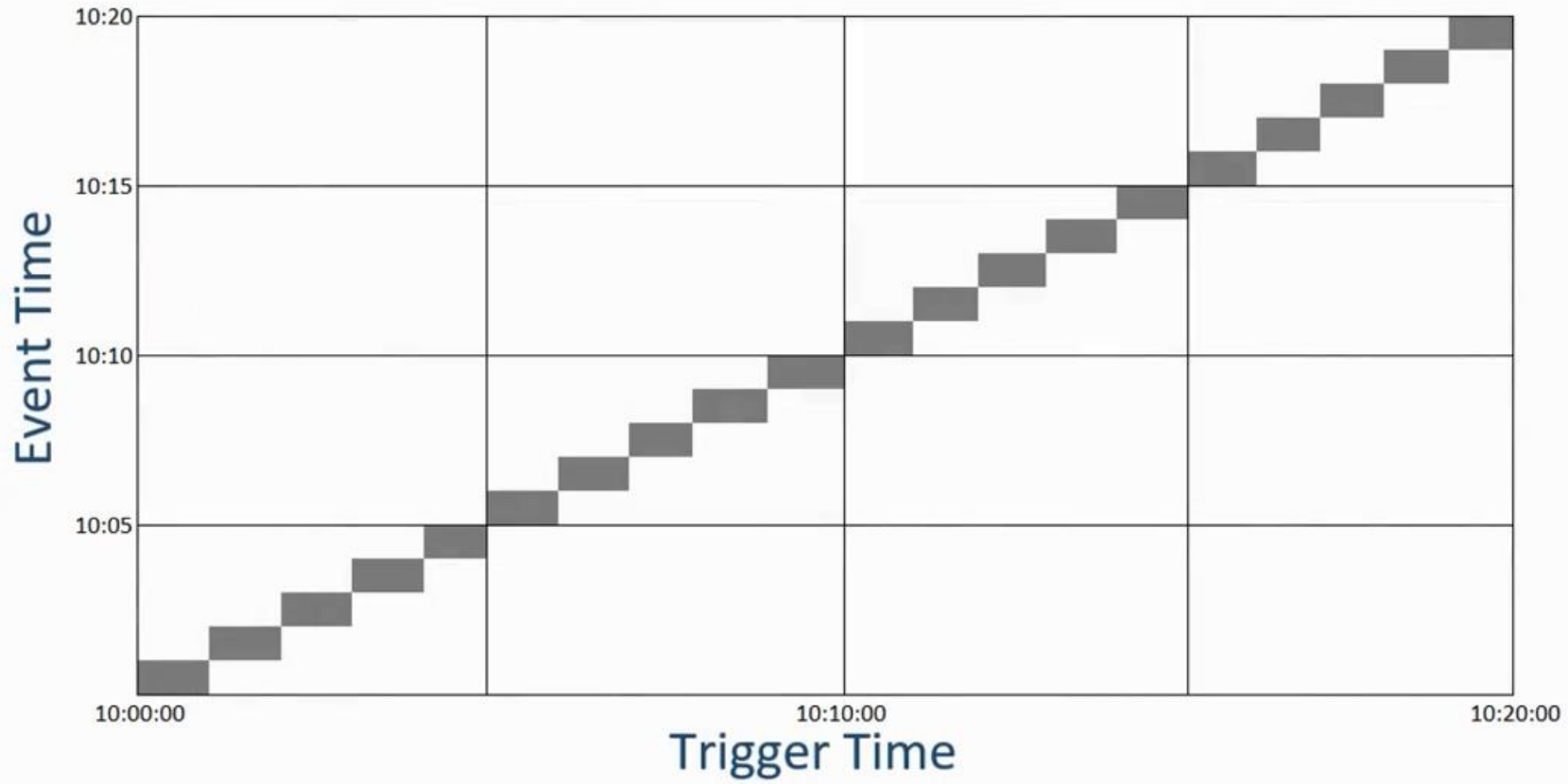


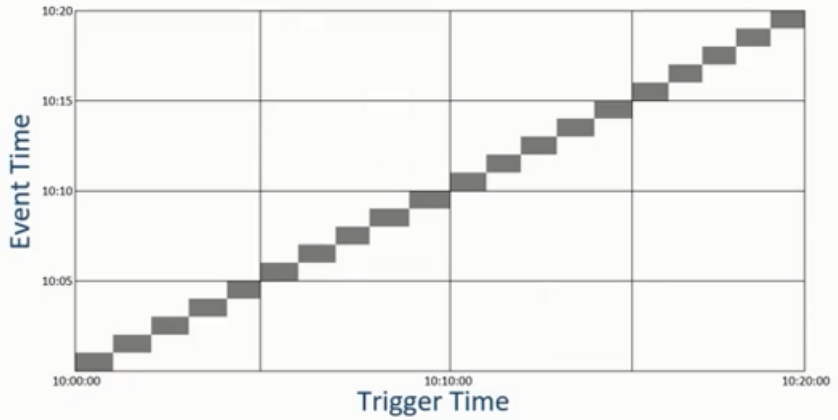
Frequency of our micro-Batches

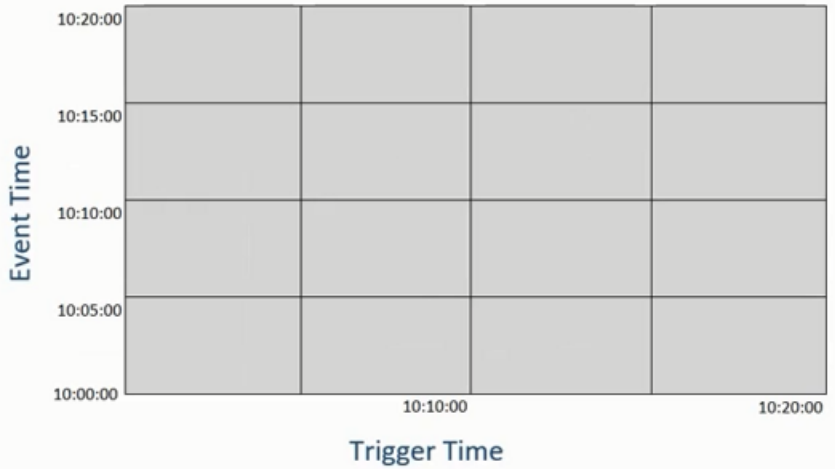

10:20:00
10:15:00
10:10:00
10:05:00
10:00:00
Event Time
10:01:40
10:10:00
10:20:00
Trigger Time

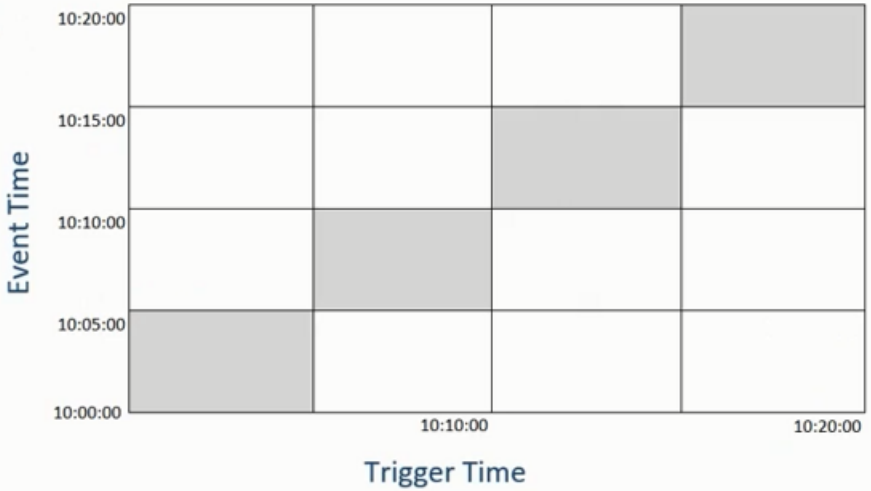

10:20:00
10:15:00
10:10:00
10:05:00
10:00:00
Event Time
10:10:00
10:20:00
Trigger Time

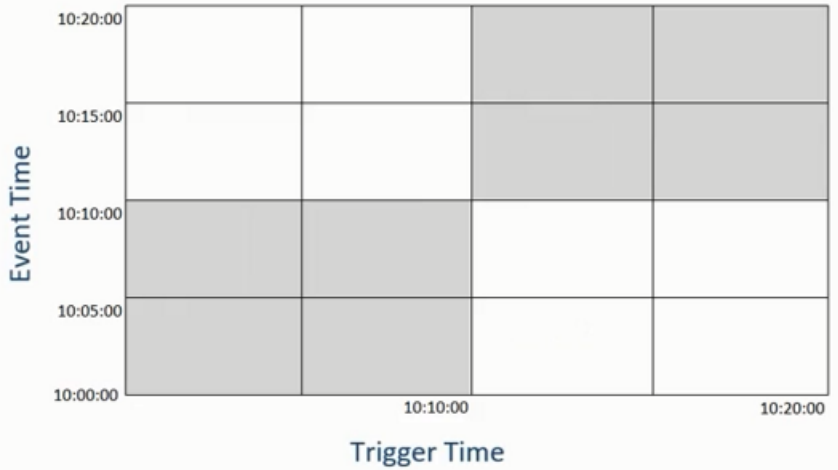

10:20:00
10:15:00
10:10:00
10:05:00
10:00:00
Event Time
10:10:00
10:20:00
Trigger Time


10:20
10:15
10:10
10:05
Event Time
10:00:00
10:10:00
10:20:00
Trigger Time



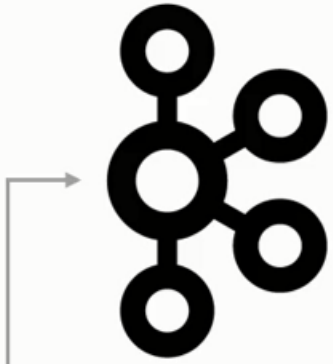






The Aggregation window has nothing to do with trigger time!

The Trigger is the time when we start proccessing our micro-batch



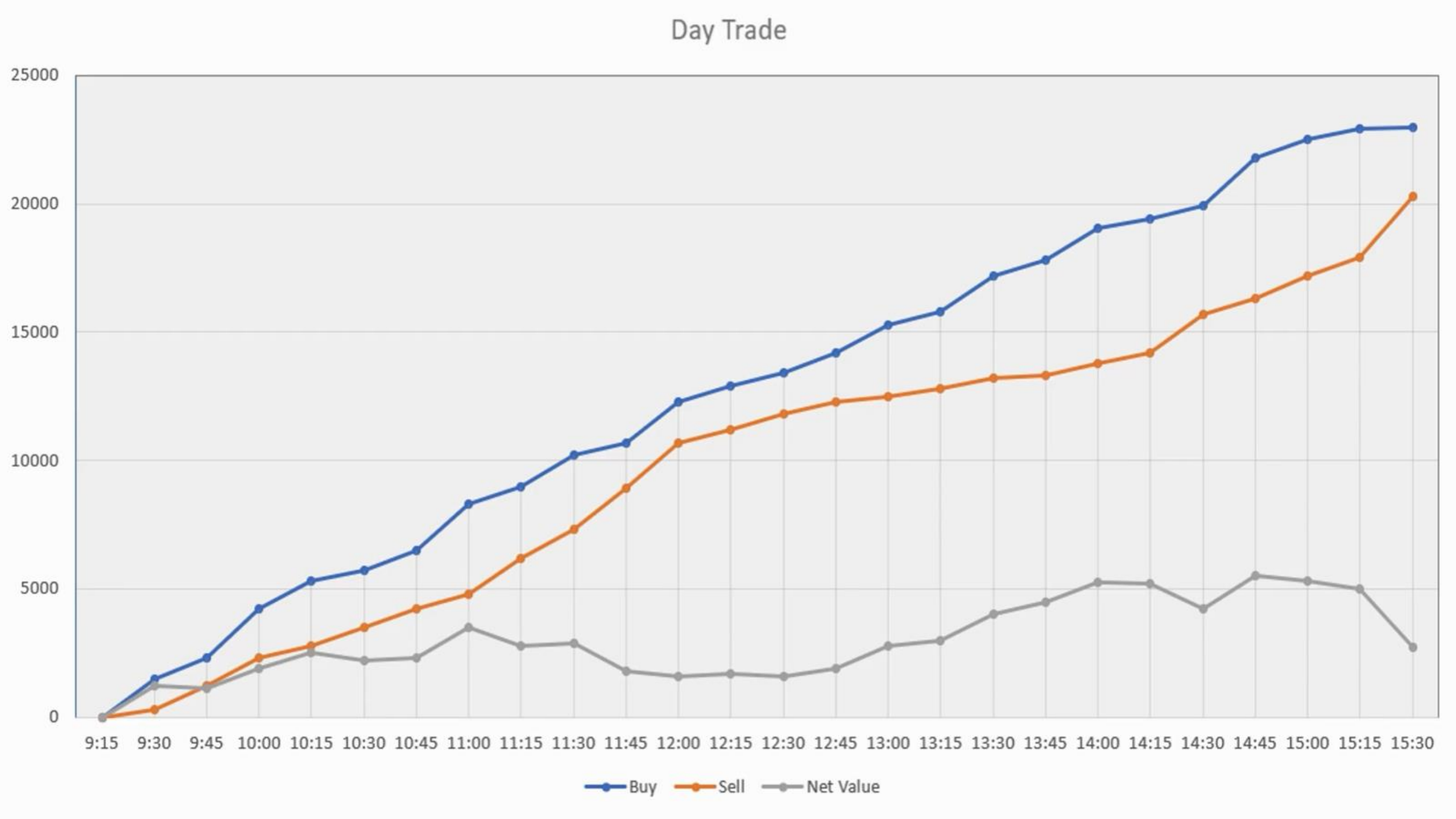
Trade Events


```
{"TransactionTime": "18 : 07 : 2024  10:01:12",
 "TransactionType": "BUY",
 "TransactionAmount": 500,
 "BrokerCode": "ABX"}

{"TransactionTime": "18 : 07 : 2024  10:05:30",
 "TransactionType": "SELL",
 "TransactionAmount": 300, ,
 "BrokerCode": "ABX"}
```

Trade Events


```
{"TransactionTime": "18 : 07 : 2024  10:01:12",
 "TransactionType": "BUY",
 "TransactionAmount": 500,
 "BrokerCode": "ABX"}

{"TransactionTime": "18 : 07 : 2024  10:05:30",
 "TransactionType": "SELL",
 "TransactionAmount": 300, ,
 "BrokerCode": "ABX"}
```


Start Time	End Time	Buy	Sell	Net Value
9:00	9:15	0	0	0
9:15	9:30	1500	300	1200
9:30	9:45	2300	1200	1100
9:45	10:00	4200	2300	1900
10:00	10:15	5300	2800	2500
10:15	10:30	5700	3500	2200
10:30	10:45	6500	4200	2300
10:45	11:00	8300	4800	3500


Day Trade
25000
20000
15000
10000
5000
0
9:15
9:30
9:45
10:00
10:15
10:30
10:45
11:00
11:15
11:30
11:45
12:00
12:15
12:30
12:45
13:00
13:15
13:30
13:45
14:00
14:15
14:30
14:45
15:00
15:15
15:30
Buy
Sell
Net Value

Trigger = 15 Min

Apache Spark Structured Streaming

Trade Events


```
{"TransactionTime": "18 : 07 : 2024  10:01:12",
 "TransactionType": "BUY",
 "TransactionAmount": 500,
 "BrokerCode": "ABX"}

{"TransactionTime": "18 : 07 : 2024  10:05:30",
 "TransactionType": "SELL",
 "TransactionAmount": 300, ,
 "BrokerCode": "ABX"}
```


Start Time	End Time	Buy	Sell	Net Value
9:00	9:15	0	0	0
9:15	9:30	1500	300	1200
9:30	9:45	2300	1200	1100
9:45	10:00	4200	2300	1900
10:00	10:15	5300	2800	2500
10:15	10:30	5700	3500	2200
10:30	10:45	6500	4200	2300
10:45	11:00	8300	4800	3500

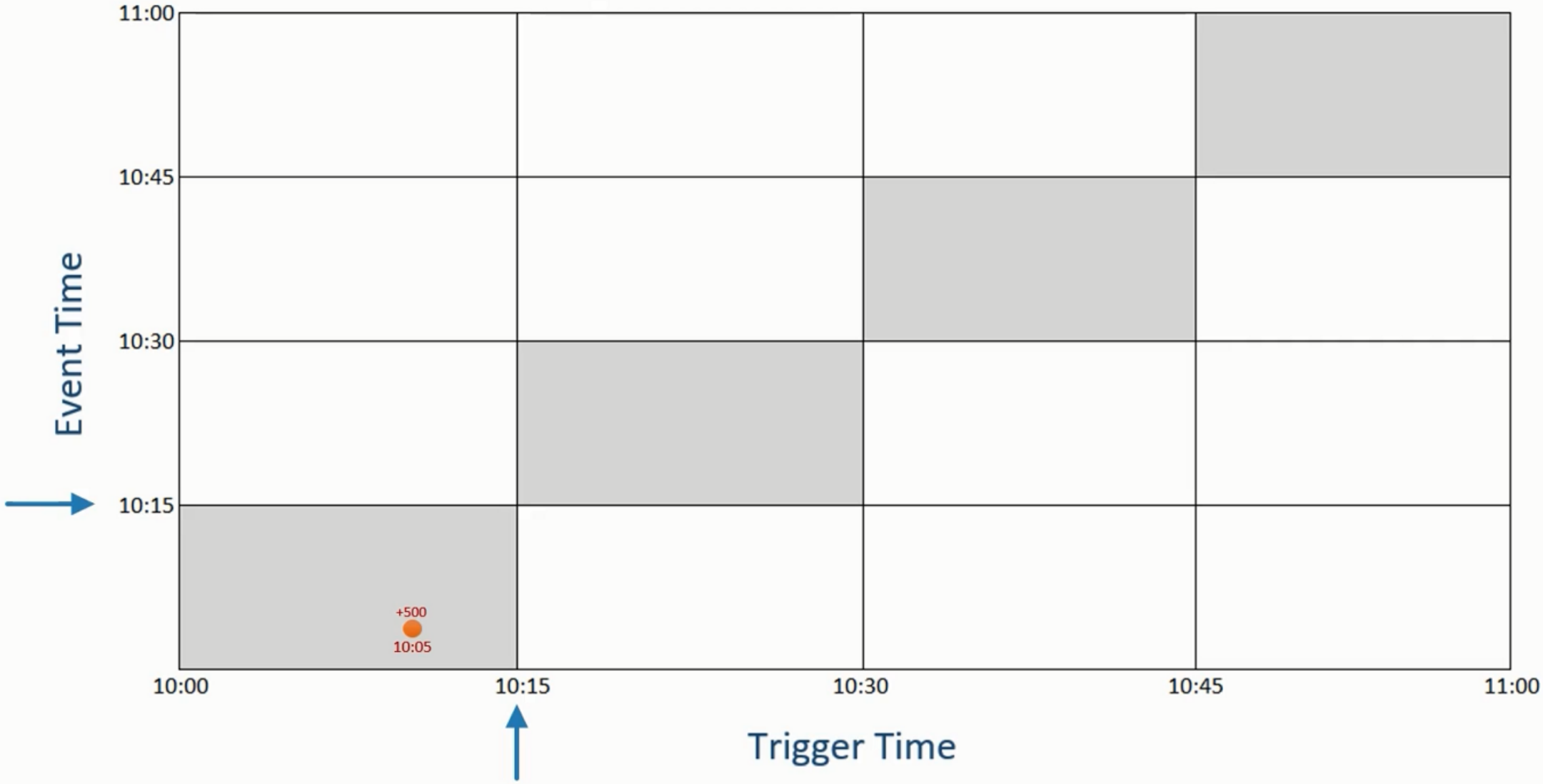
Trigger = 15 Min

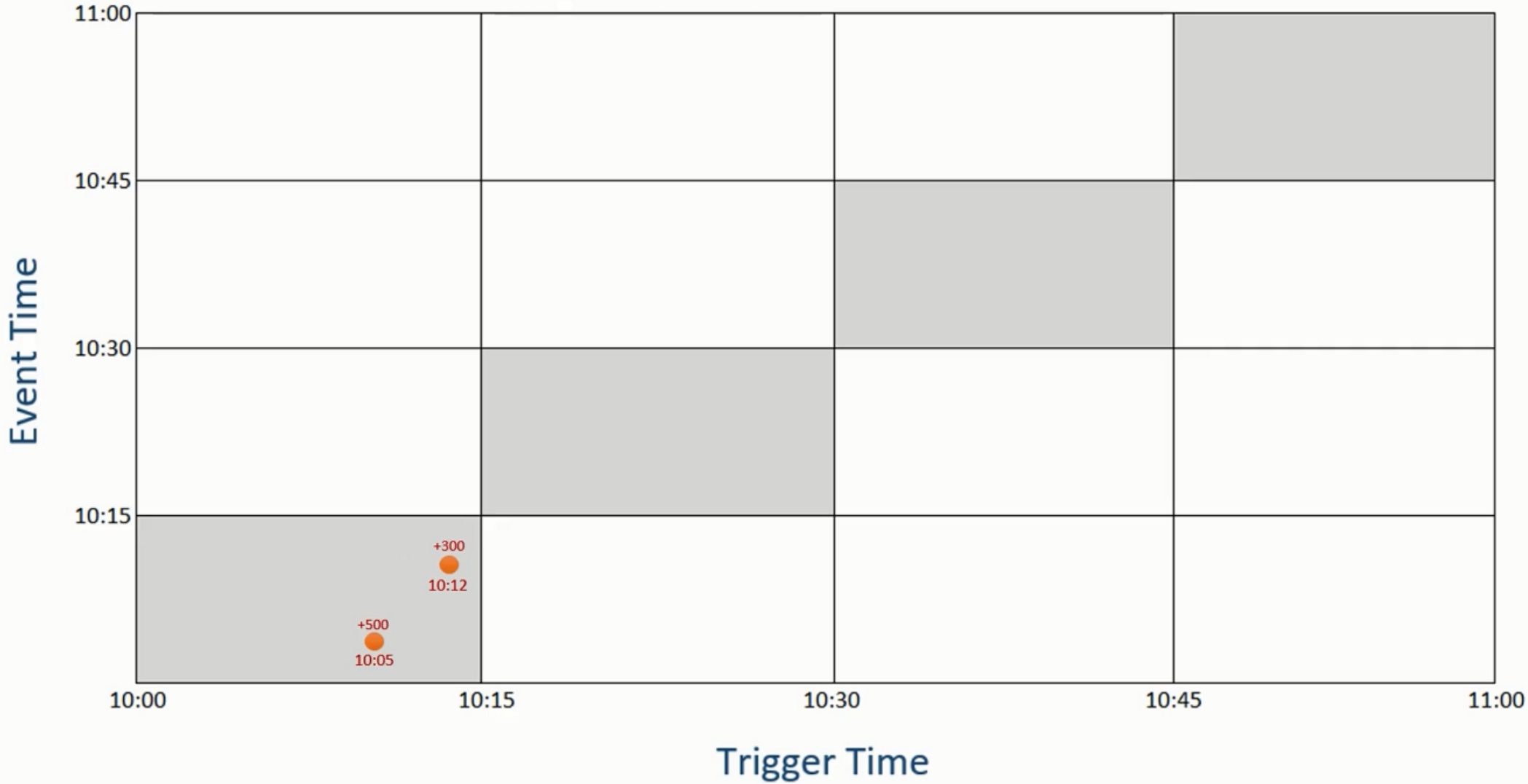
Trade Events

groupBy(window('15 minute'))

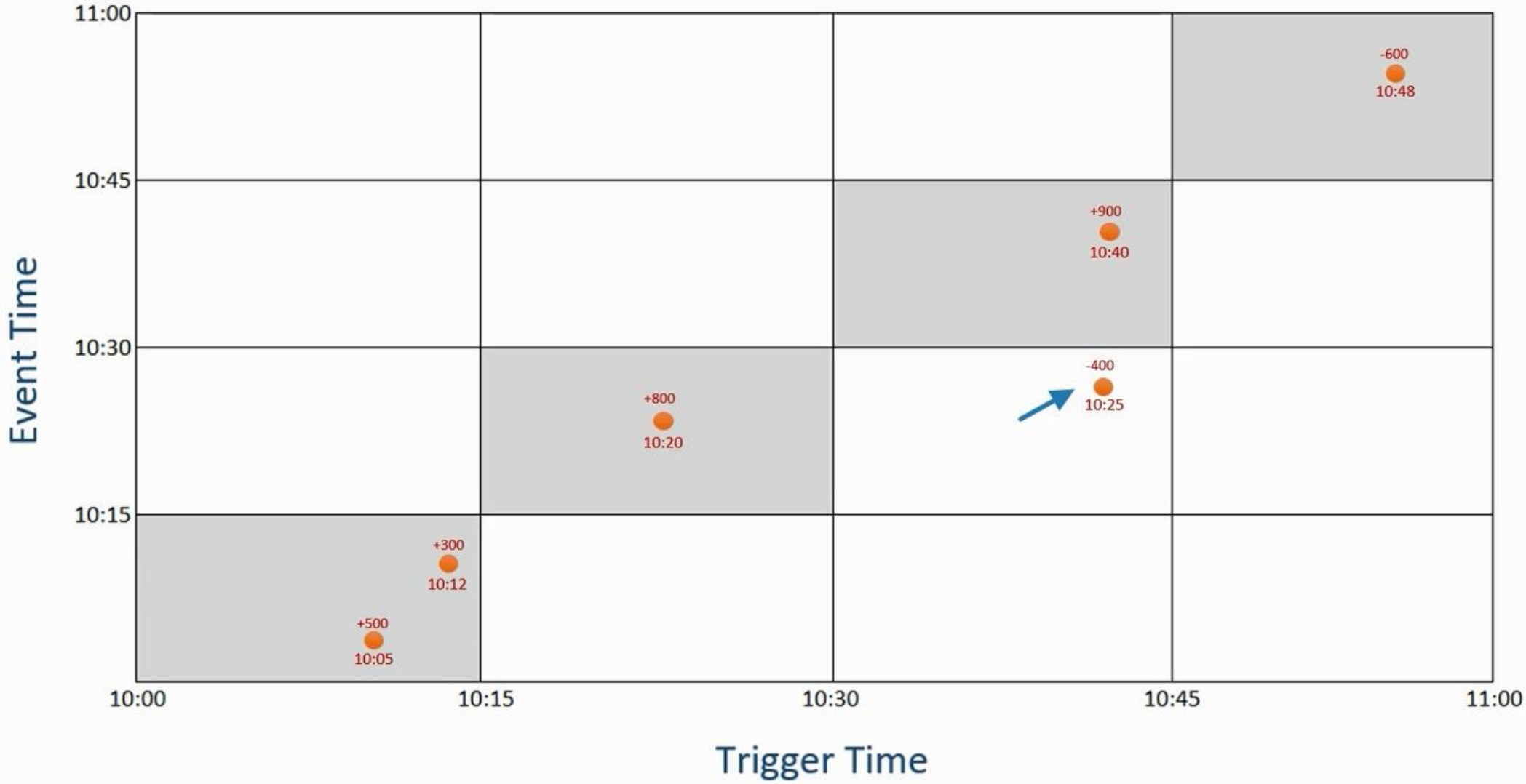

```
{"TransactionTime": "18 : 07 : 2024  10:01:12",
 "TransactionType": "BUY",
 "TransactionAmount": 500,
 "BrokerCode": "ABX"}

{"TransactionTime": "18 : 07 : 2024  10:05:30",
 "TransactionType": "SELL",
 "TransactionAmount": 300,
 "BrokerCode": "ABX"}
```


Start Time	End Time	Buy	Sell	Net Value
9:00	9:15	0	0	0
9:15	9:30	1500	300	1200
9:30	9:45	2300	1200	1100
9:45	10:00	4200	2300	1900
10:00	10:15	5300	2800	2500
10:15	10:30	5700	3500	2200
10:30	10:45	6500	4200	2300
10:45	11:00	8300	4800	3500


11:00
10:45
10:30
10:15
Event Time
+500
10:05
10:00
10:15
10:30
10:45
11:00
Trigger Time


11:00
10:45
10:30
10:15
Event Time
+300
10:12
+500
10:05
10:00
10:15
10:30
10:45
11:00
Trigger Time


11:00
10:45
10:30
10:15
Event Time
-600
10:48
+900
10:40
+800
10:20
+300
10:12
+500
10:05
10:00
10:15
10:30
10:45
11:00
Trigger Time


11:00
10:45
10:30
10:15
Event Time
-600
10:48
+900
10:40
-400
10:25
+800
10:20
+300
10:12
+500
10:05
10:00
10:15
10:30
10:45
11:00
Trigger Time

Now let’s go for coding!