

Discounts Processor

Git commit with doc: d9181bd

[HTML version](#)

1. Overview	1
2. Running the Application	2
2.1. Prerequisites	2
2.2. Using the Run Script	2
2.2.1. In development mode	2
2.2.2. In production mode (integrated with the main project)	3
3. Testing Procedures	3
3.1. Unit and Mock Tests	3
3.1.1. Running	3
3.2. Integration Tests	4
3.2.1. Running Integration Tests with TestContainers	4
3.3. Troubleshooting	4
3.3.1. Common Issues	4
3.3.2. Logs	4
4. Discount Conditions	4
4.1. Continuous View Discount	4
4.2. Most Viewed Product Discount	5
5. Event examples	5
5.1. Input (in <code>snowplow-enriched-good</code> topic)	5
5.2. Output (in <code>shopper-discounts</code> topic)	5

1. Overview

This project (**Discounts Processor**) is a Java/ [Apache Flink](#) application that processes event data from [Redpanda](#) and generates discount events based on user behavior patterns. It analyzes user interactions with products and creates targeted discount opportunities.

This module can be treated almost^[1] entirely independently of [the main project](#) that encompasses it. It is an alternative intended to produce the same result as the [connect.yaml](#) file configured using [Redpanda Connect](#). In other words, it is a discount processor that works under the same [rules](#).

This module also contains [real integration tests](#) that are executed with the help of [TestContainers](#).

2. Running the Application

2.1. Prerequisites

1. Bash
2. Java
3. [Docker](#)

These were the versions used during development:

```
$ echo $BASH_VERSION
5.2.21(1)-release

$ java -version
openjdk version "17.0.14" 2025-01-21
OpenJDK Runtime Environment Temurin-17.0.14+7 (build 17.0.14+7)
OpenJDK 64-Bit Server VM Temurin-17.0.14+7 (build 17.0.14+7, mixed mode, sharing)

$ docker --version
Docker version 27.3.1, build ce12230
```

2.2. Using the Run Script

2.2.1. In development mode

First, you need to start Redpanda using docker. To simplify the start/stop, use the [docker.sh](#) script:

```
$ ./docker.sh up
```

The application can be executed using the provided [run.sh](#) script, which handles building and running the application:

Alternative 1: Start the application with an existing JAR:

```
$ ./run.sh
```

Alternative 2: Force a rebuild and start the application:

```
$ ./run.sh --build
```

The script will:

- Build the application if the JAR doesn't exist or if **--build** is specified

- Configure appropriate JVM options
- Handle graceful shutdown on SIGTERM/SIGINT
- Display colored status messages during execution



The script automatically manages the application lifecycle and provides proper cleanup on shutdown.

To generate events in order to trigger discounts, use [the Simulator project](#).

To verify that discounts are being generated correctly, consume from the output topic:

```
$ docker exec -it redpanda rpk topic consume shopper-discounts
```

You should see discount events in the output that match the expected patterns based on the input events.

2.2.2. In production mode (integrated with the main project)

In this mode you can test this script integrated with [the main project](#).



This is not finished yet.

3. Testing Procedures

This document outlines the testing procedures for the Discounts Processor application.

3.1. Unit and Mock Tests

Unit tests verify the core business logic of the discount processors without external dependencies.

3.1.1. Running

To run all unit and mock tests:

```
$ ./gradlew test
```

To run a specific test class:

```
$ ./gradlew test --tests "com.example.DiscountsProcessorTest"
```

To run a mocked test for the discount processor:

```
$ ./gradlew test --tests "com.example.DiscountsProcessorMockTest"
```

3.2. Integration Tests

The integration tests cover tests for the following processors:

- [src/main/java/com/example/processor/ContinuousViewProcessor.java](#).
- [src/main/java/com/example/processor/MostViewedProcessor.java](#).

3.2.1. Running Integration Tests with TestContainers

To run the a integration test with [TestContainers](#):

```
$ ./gradlew test --tests "com.example.DiscountsProcessorIntegrationTest"
```

To run all integration tests:

```
$ ./gradlew integrationTest
```

3.3. Troubleshooting

3.3.1. Common Issues

- **No discounts generated:** Ensure that enough events are being sent to trigger the discount conditions (9+ events for continuous view, 5+ events for most viewed).
- **Connection issues:** Verify that Redpanda is running and accessible.
- **Serialization errors:** Check that the event format matches the expected schema.

3.3.2. Logs

Application logs can be configured in two files depending on the environment (real or test):

1. [logback.xml](#)
2. [logback-test.xml](#)

4. Discount Conditions

4.1. Continuous View Discount

A single discount of 10% (witin a 5-minute window) is generated when:

- A user views the same product continuously within a 90-second window.

4.2. Most Viewed Product Discount

A single discount of 10% (within a 5-minute window) is generated when:

- A product is viewed at least 3 times by the same user.



This rule is being reworked to: the product with the most views within a 5 minute window gets the discount. If two products have the same number of views, then the one with the longest viewing time will get the discount.

5. Event examples

5.1. Input (in **snowplow-enriched-good** topic)

snowplow_ecommerce_action event:

```
{
  "collector_tstamp": "2025-04-04T07:05:00.119Z",
  "event_name": "snowplow_ecommerce_action",
  "user_id": "1",
  "product_id": "5",
  "product_name": "SP Flex Runner 2",
  "product_price": 42.99,
  "webpage_id": "page_5"
}
```

page_view event:

```
{
  "collector_tstamp": "2025-04-04T07:05:12.130Z",
  "event_name": "page_ping",
  "user_id": "1",
  "webpage_id": "page_5"
}
```

5.2. Output (in **shopper-discounts** topic)

discount event examples:

Continuous View Discount:

```
{
  "user_id": "1",
  "product_id": "5",
  "discount": {
```

```
"rate": 0.1,  
  "by_view_time": {  
    "duration_in_seconds": 100  
  }  
}
```

Most Viewed Product Discount:

```
{  
  "user_id": "1",  
  "product_id": "5",  
  "discount": {  
    "rate": 0.1,  
    "by_number_of_views": {  
      "views": 5,  
      "duration_in_seconds": 30  
    }  
  }  
}
```

[1] It only relies on scripts that generate documentation.