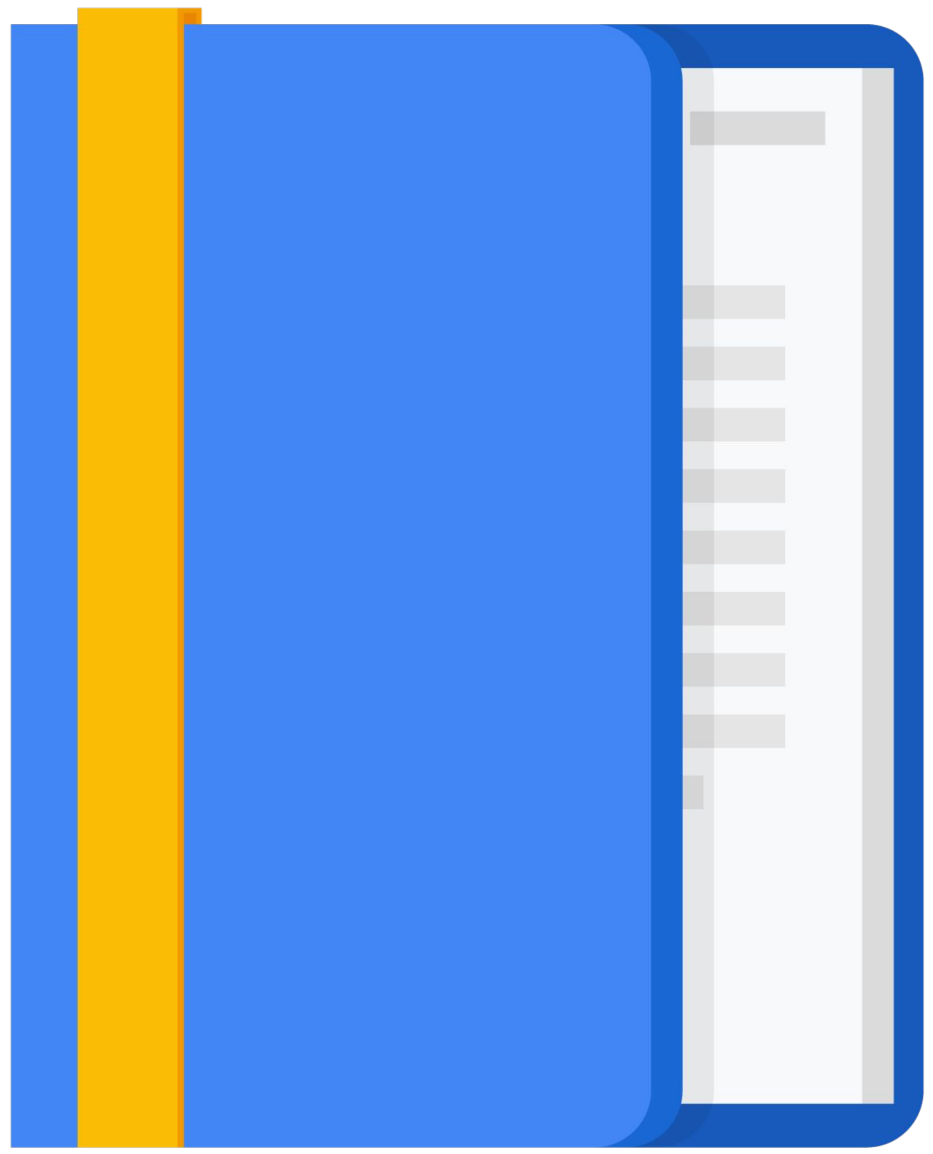
Beam Notebooks

Reza Rokni

Developer Advocate,
Google Cloud

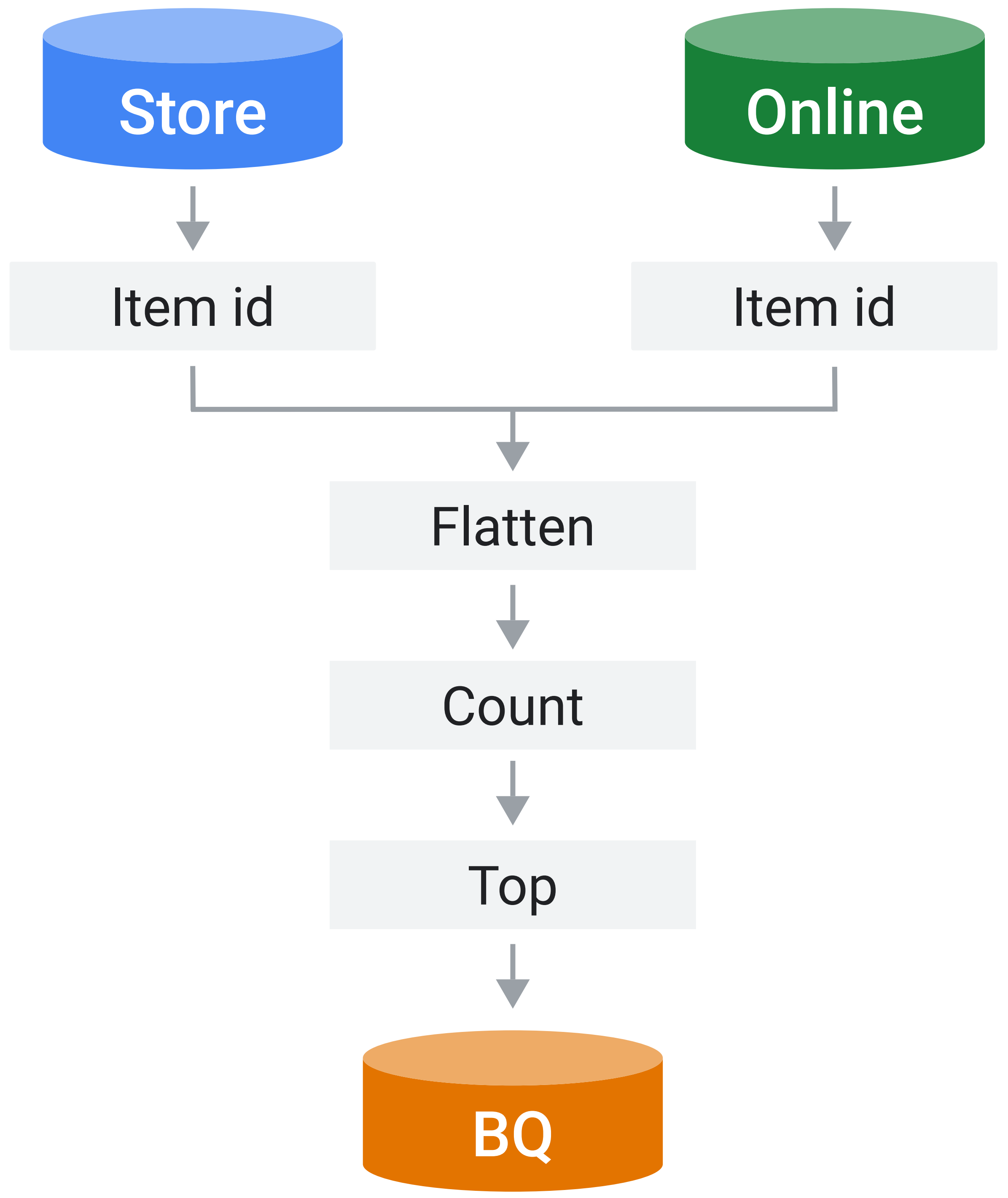
Agenda

- Course Intro
- Beam Concepts Review
- Windows, Watermarks, and Triggers
- Sources and Sinks
- Schemas
- State and Timers
- Best Practices
- SQL and DataFrames
- **Beam Notebooks**
- Summary



Apache Beam and interactive development

```
storeSales = p  |
beam.io.ReadFromText(“purchases-store”)

                | beam.Map(lambda s: ...)

onlineSales = p |
beam.io.ReadFromText(“purchases-online”)

                | beam.Map(lambda s: ...)

topSales = (storeSales, onlineSales)

                | beam.Flatten()

                | beam.Combiners.Count.perKey()

                | beam.Combiners.Top.of(10, key = lambda
x: x[1])

topSales        | beam.io.WriteToBigQuery(topSales)
```



Apache Beam interactive runner

The interactive runner module allows

1. Interactive development

Apache Beam interactive runner

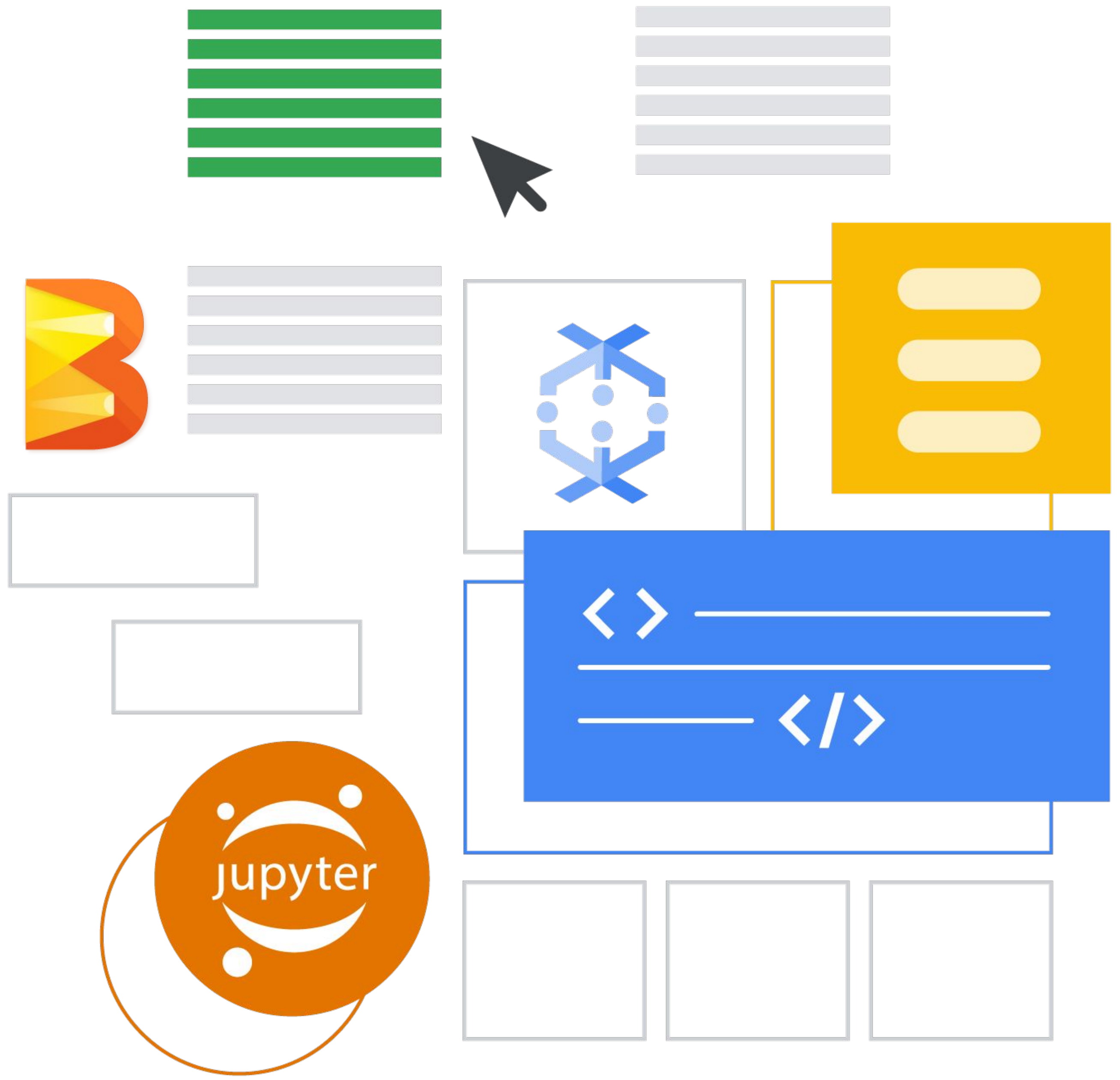
The interactive runner module allows

1. Interactive development
2. Access to intermediate results

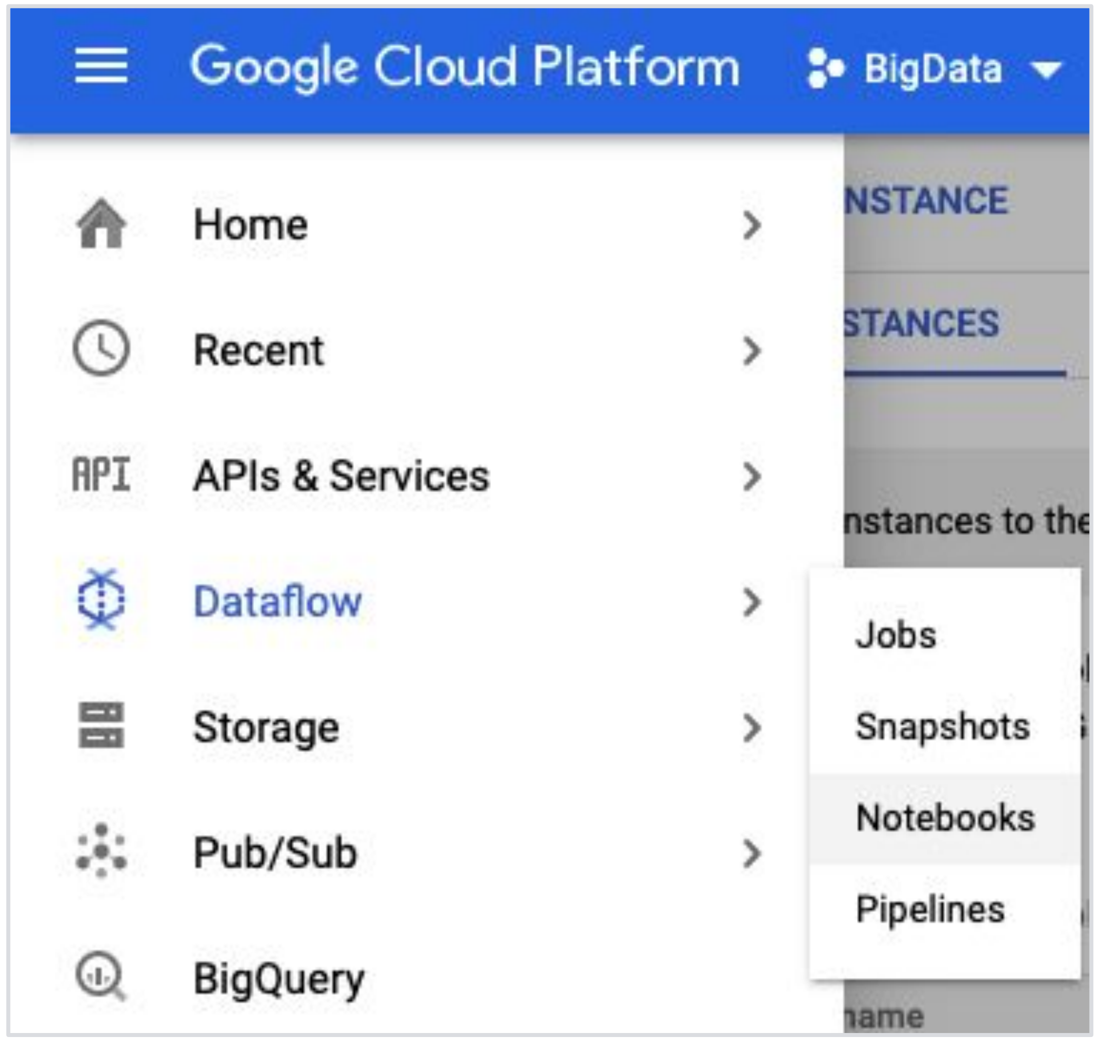
Apache Beam interactive runner

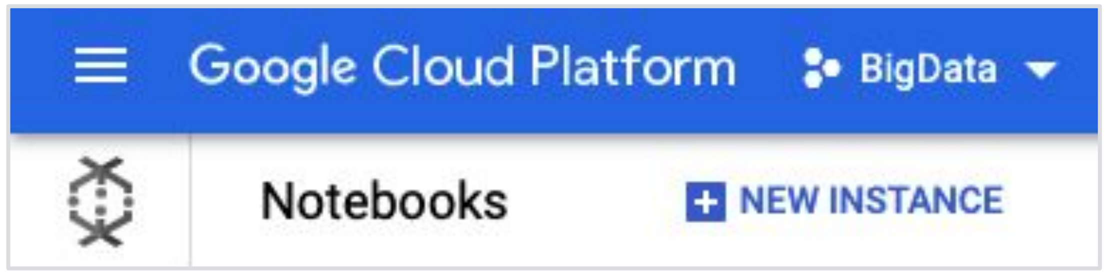
The interactive runner module allows

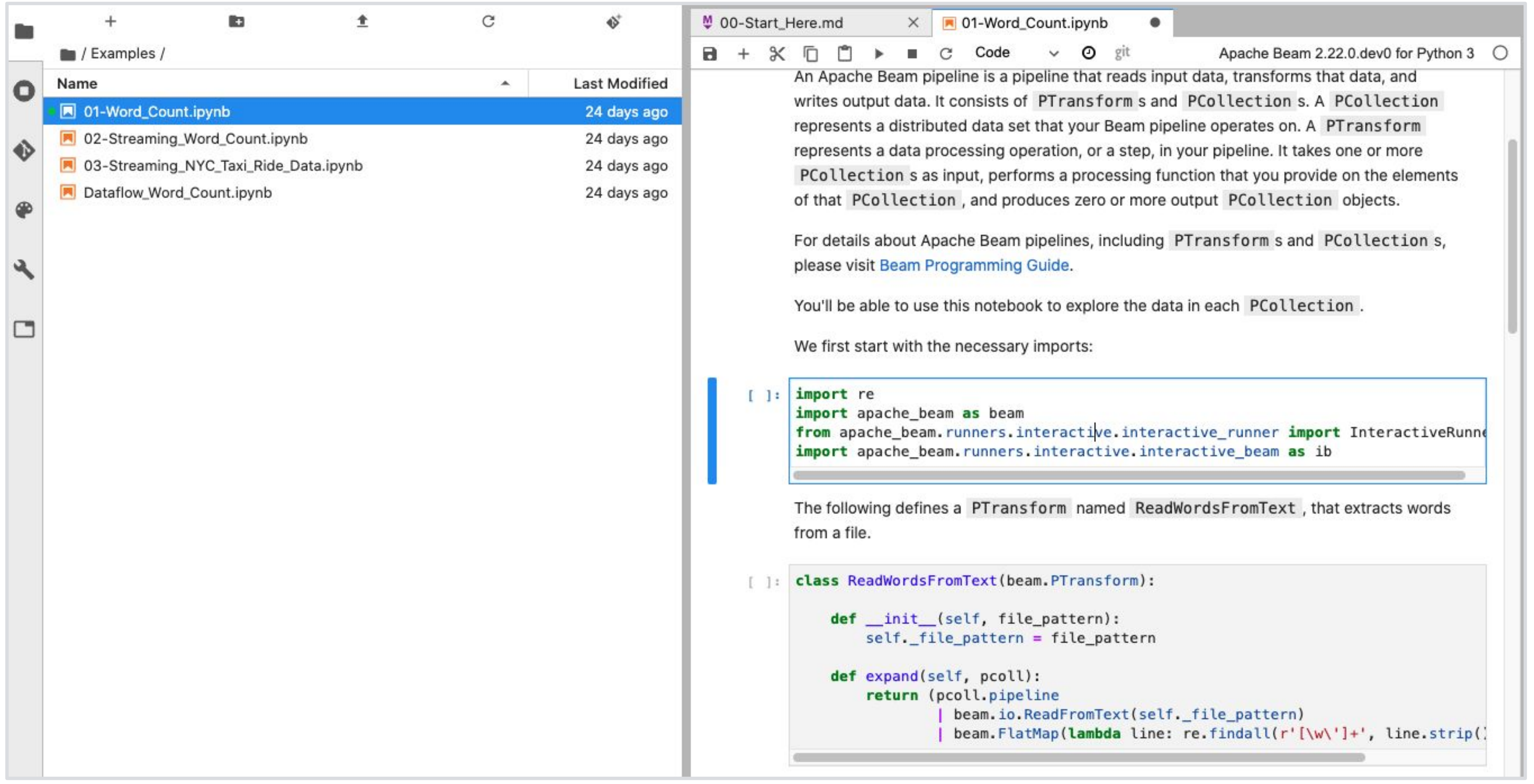
1. Interactive development
2. Access to intermediate results
3. Stream or batch sources



Beam Notebooks







Name	Last Modified
01-Word_Count.ipynb	24 days ago
02-Streaming_Word_Count.ipynb	24 days ago
03-Streaming_NYC_Taxi_Ride_Data.ipynb	24 days ago
Dataflow_Word_Count.ipynb	24 days ago

An Apache Beam pipeline is a pipeline that reads input data, transforms that data, and writes output data. It consists of `PTransform` s and `PCollection` s. A `PCollection` represents a distributed data set that your Beam pipeline operates on. A `PTransform` represents a data processing operation, or a step, in your pipeline. It takes one or more `PCollection` s as input, performs a processing function that you provide on the elements of that `PCollection` , and produces zero or more output `PCollection` objects.

For details about Apache Beam pipelines, including `PTransform` s and `PCollection` s, please visit Beam Programming Guide.

You'll be able to use this notebook to explore the data in each `PCollection` .

We first start with the necessary imports:

```
[ ]: import re
     import apache_beam as beam
     from apache_beam.runners.interactive.interactive_runner import InteractiveRunne
     import apache_beam.runners.interactive.interactive_beam as ib
```

The following defines a `PTransform` named `ReadWordsFromText` , that extracts words from a file.

```
[ ]: class ReadWordsFromText(beam.PTransform):

         def __init__(self, file_pattern):
             self._file_pattern = file_pattern

         def expand(self, pcoll):
             return (pcoll.pipeline
                     | beam.io.ReadFromText(self._file_pattern)
                     | beam.FlatMap(lambda line: re.findall(r'[\w\']+', line.strip(
```

Add a transform

```
words = p | "read" >> beam.io.ReadFromPubSub(topic=topic)

windowed_words = (words
   | "window" >>
beam.WindowInto(beam.window.FixedWindows(10)))

windowed_words_counts = (windowed_words
   | "count" >> beam.combiners.Count.PerElement())
```

Set interactivity options before we run the cell

ib.options.recording_duration

Sets the amount of time the InteractiveRunner records data from an unbounded source

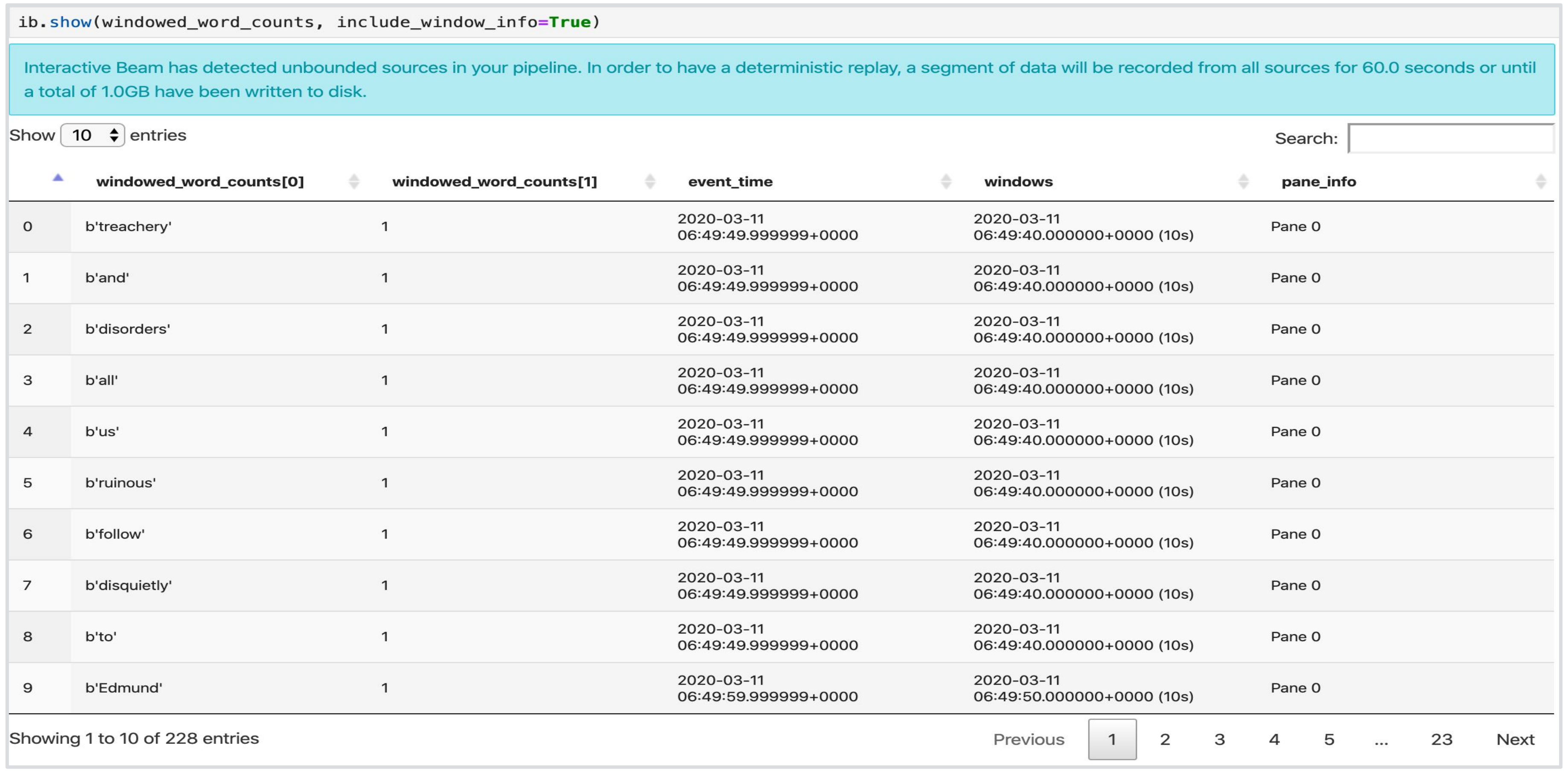
```
# Set the recording duration to 10 min
ib.options.recording_duration = ‘10m‘
```

ib.options.recording_size_limit

Sets the amount of data the InteractiveRunner records (in bytes) from an unbounded source

```
# Set the recording size limit to 1 GB
ib.options.recording_size_limit = 1e9
```

Access Transform output



```
ib.show(windowed_word_counts, include_window_info=True)
```

Interactive Beam has detected unbounded sources in your pipeline. In order to have a deterministic replay, a segment of data will be recorded from all sources for 60.0 seconds or until a total of 1.0GB have been written to disk.

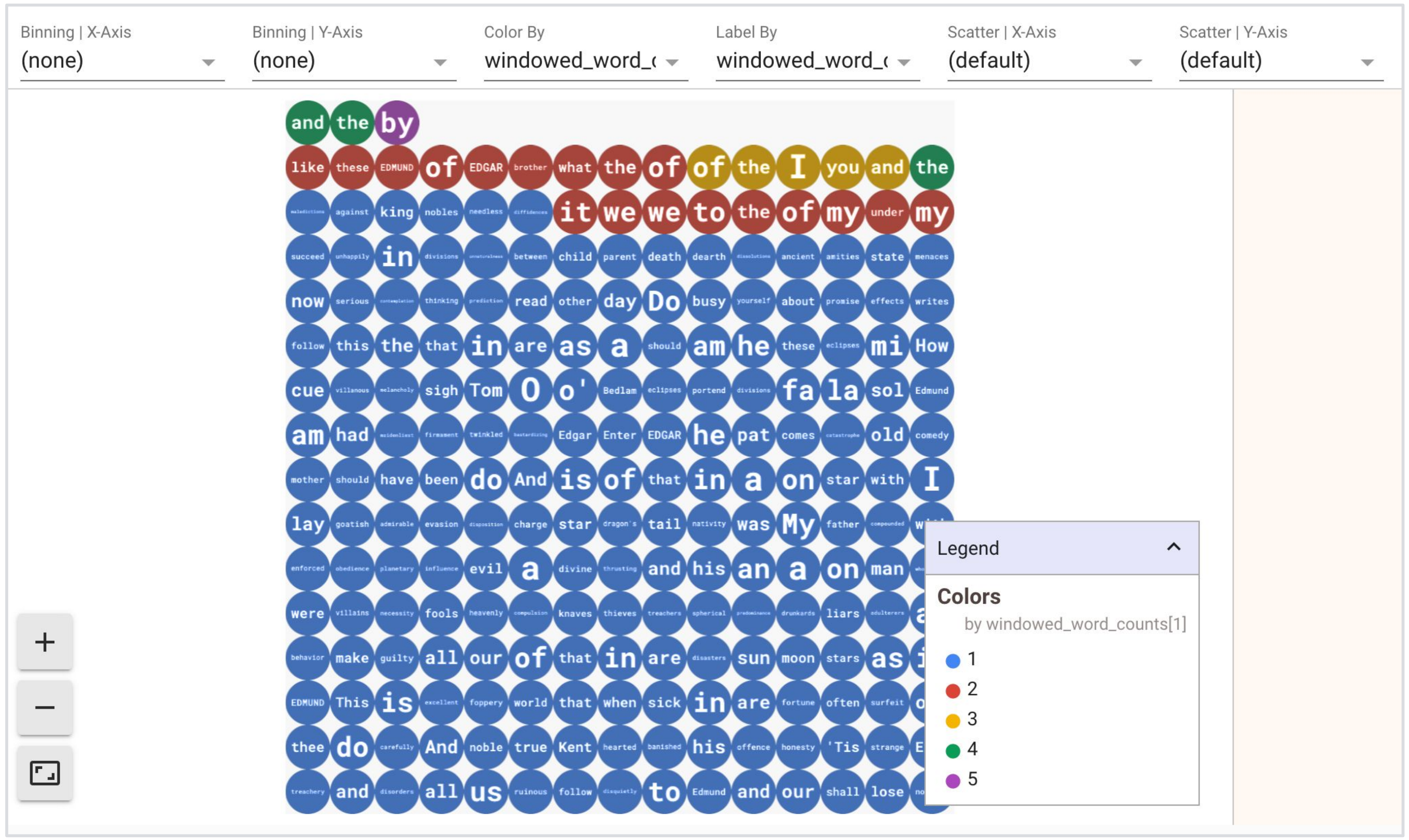
Show 10 entries Search:

	windowed_word_counts[0]	windowed_word_counts[1]	event_time	windows	pane_info
0	b'treachery'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
1	b'and'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
2	b'disorders'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
3	b'all'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
4	b'us'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
5	b'ruinous'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
6	b'follow'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
7	b'disquietly'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
8	b'to'	1	2020-03-11 06:49:49.999999+0000	2020-03-11 06:49:40.000000+0000 (10s)	Pane 0
9	b'Edmund'	1	2020-03-11 06:49:59.999999+0000	2020-03-11 06:49:50.000000+0000 (10s)	Pane 0

Showing 1 to 10 of 228 entries Previous 1 2 3 4 5 ... 23 Next

```
# Materializes the resulting PCollection in a table
ib.show(windowed_word_counts, include_window_info=True)

# Load the output in a Pandas DataFrame
ib.collect(windowed_word_counts, include_window_info=True)
```

Visualize Transform output



```
# Visualize the data in the notebook
ib.show(windowed_word_counts, include_window_info=True, visualize_data=True)
```

Going from development to production

```
# Import the production Dataflow runner
from apache_beam.runners import DataflowRunner

# Set up Apache Beam pipeline options
options = pipeline_options.PipelineOptions()

# Run the pipeline
runner = DataflowRunner()
runner.run_pipeline(p, options=options)
```