Lab 2 - Streaming analytics with Amazon Managed Service for Apache Flink

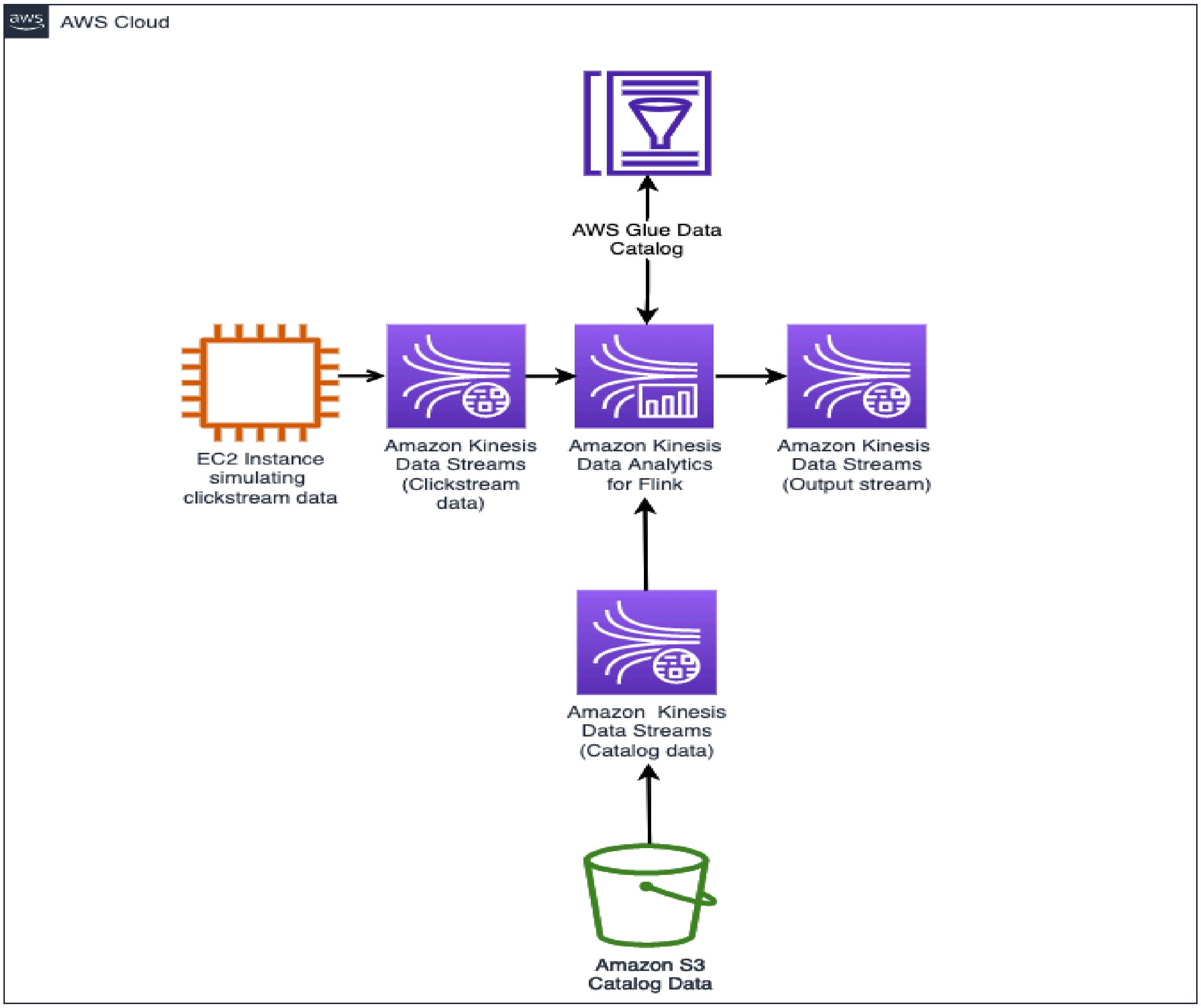
Objective:

1. Build a real-time streaming analytics pipeline in Managed Apache Flink Studio using Apache Flink to ingest, enrich, and analyze the clickstream data.
2. Perform interactive data analytics and visualize using Apache Zeppelin notebooks with Managed Apache Flink Studio.

Imagine you have a river of data flowing in, and you want to collect and sort different things from it without stopping the flow. **Flink can do that.**
It can handle data as it comes in, making it possible to get insights and take actions right away.

Process:

1. Collects Data: Gather clickstream data from the website using Amazon Kinesis.
2. Processes Data: Use Amazon Managed Service for Apache Flink to process this data quickly.
3. Enriches Data: Combine the clickstream data with product information stored in Amazon S3 to add catalog details.
4. Analyze Data


AWS Cloud
AWS Glue Data Catalog
EC2 Instance simulating clickstream data
Amazon Kinesis Data Streams (Clickstream data)
Amazon Kinesis Data Analytics for Flink
Amazon Kinesis Data Streams (Output stream)
Amazon Kinesis Data Streams (Catalog data)
Amazon S3 Catalog Data

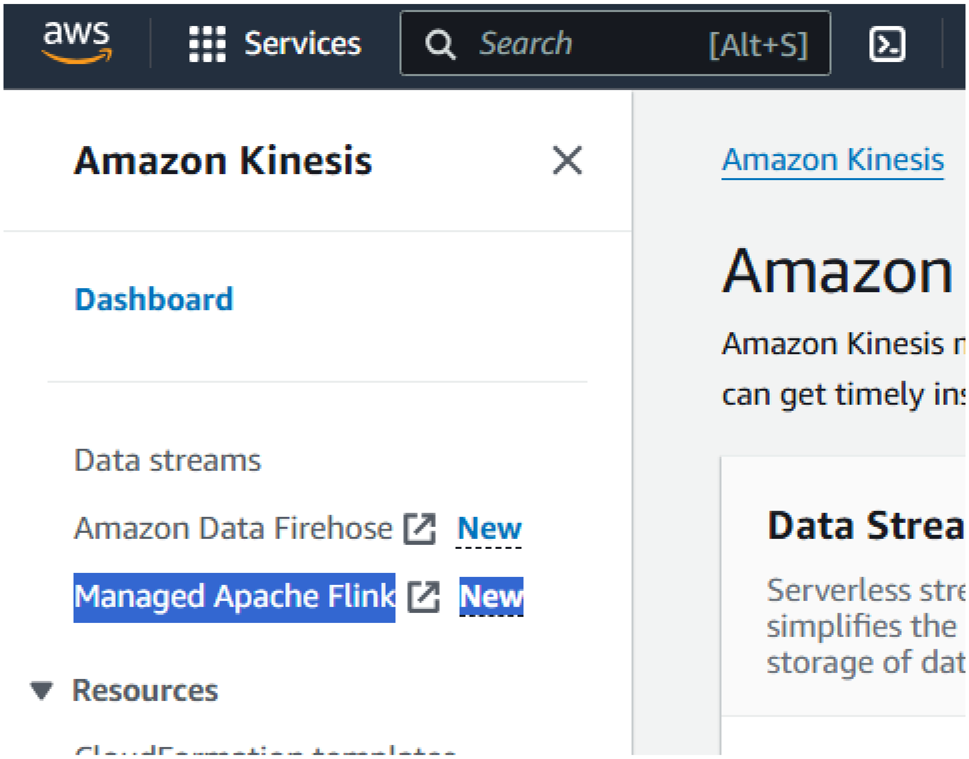
Note - We saw how to send data from EC2 to S3 in the last lab.
Now, here:
Kinesis Data Stream (1) collects clicking data from EC2 and sends to Flink for analysis.
Also, KDS (2) transports catalog data from S3 and sends it to Flink for data enrichment.
The Flink application running in Amazon Kinesis Data Analytics combines the clickstream data with the catalog data.

Output stream KDS(3) sends data for analysis and visualization.

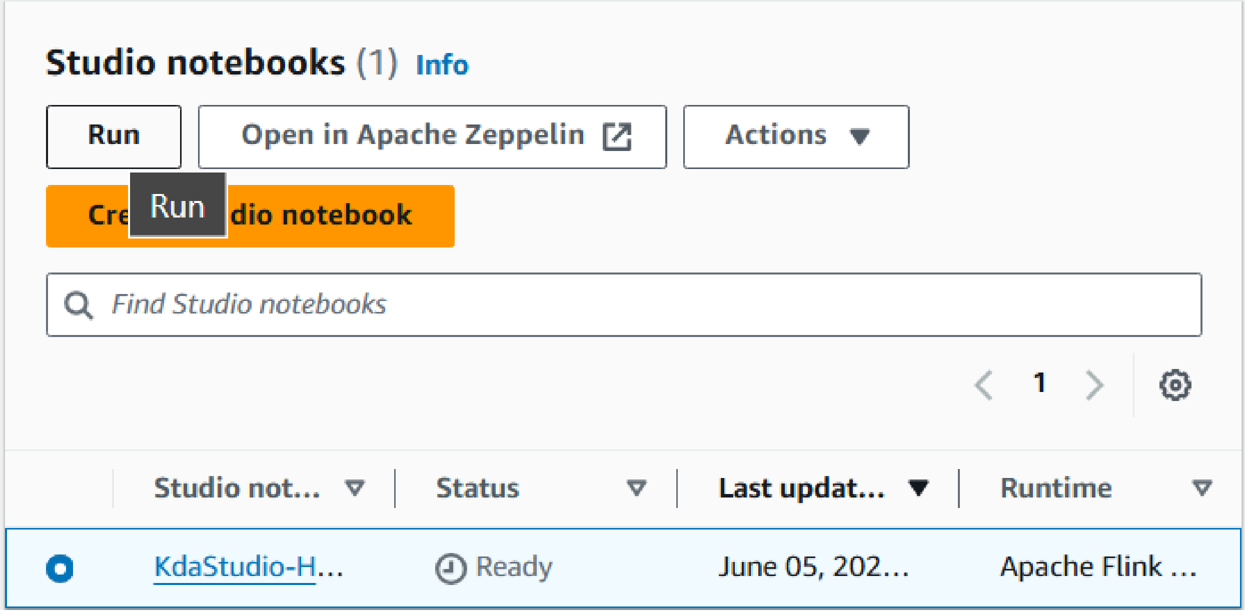
{AWS Glue Data Catalog is used to keep track of metadata.}

Task 1: Setting up Zeppelin notebook environment

1.1 Open Kinesis from console and click on Apache Flink

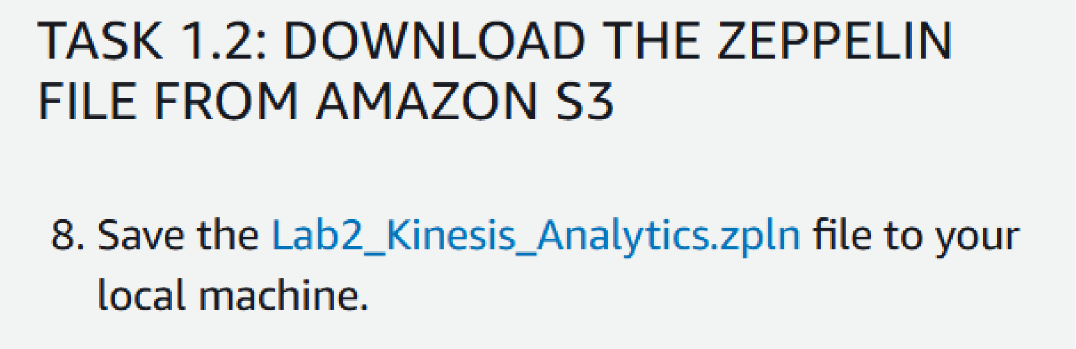


1.2 Open studio notebook option from left pane, and run this



	Studio not...	Status	Last updat...	Runtime
●	KdaStudio-H...	Ready	June 05, 202...	Apache Flink ...

1.3 Download the zeppelin file from the link



TASK 1.2: DOWNLOAD THE ZEPPELIN FILE FROM AMAZON S3

8. Save the Lab2_Kinesis_Analytics.zpln file to your local machine.

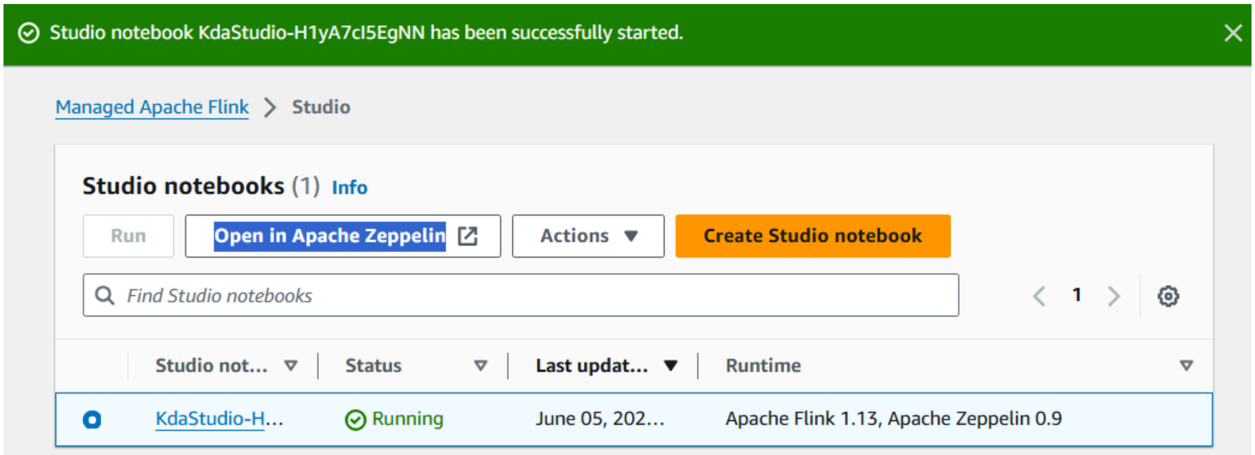
<span style="color:red">**Till then :**</span>

Task 2: Connect to the Amazon EC2 producer and start the clickstream generator

Skipping, same steps as lab 1

Task 3: Import the Zeppelin notebook

3.1 Click on open in zeppelin



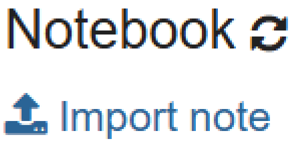
Studio notebook KdaStudio-H1yA7cI5EgNN has been successfully started.

Managed Apache Flink > Studio

Studio notebooks (1) Info

	Studio not...	Status	Last updat...	Runtime
●	KdaStudio-H...	Running	June 05, 202...	Apache Flink 1.13, Apache Zeppelin 0.9

3.2 Inside zeppelin UI, import the downloaded notebook

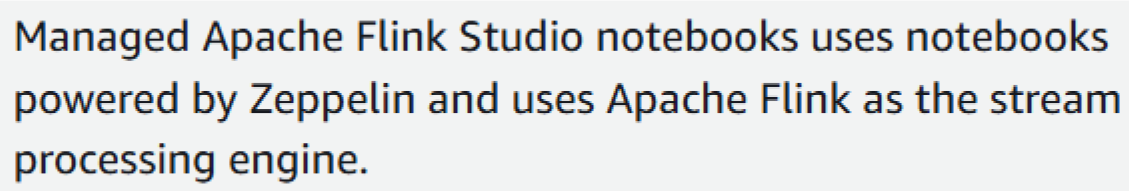


Task 4: Analytics development in Managed Apache Flink Studio with the Zeppelin notebook

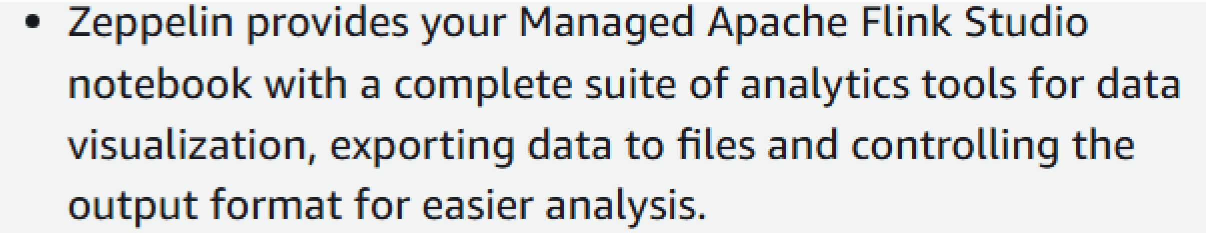
The Managed Apache Flink application, which will be set up in the Flink Studio notebook, will consume the data from the Kinesis Data Stream, whose name was given in .py code.

Flink Studio Notebook:
This is where we "talk" to Managed Apache Flink. It's like a digital workspace where we can ask questions about the clickstream data and tell Flink what to do with it.

Interactively Query Data Streams:
Instead of just looking at the data all at once, we can ask Flink questions about the data in real-time



Managed Apache Flink Studio notebooks uses notebooks powered by Zeppelin and uses Apache Flink as the stream processing engine.



- Zeppelin provides your Managed Apache Flink Studio notebook with a complete suite of analytics tools for data visualization, exporting data to files and controlling the output format for easier analysis.

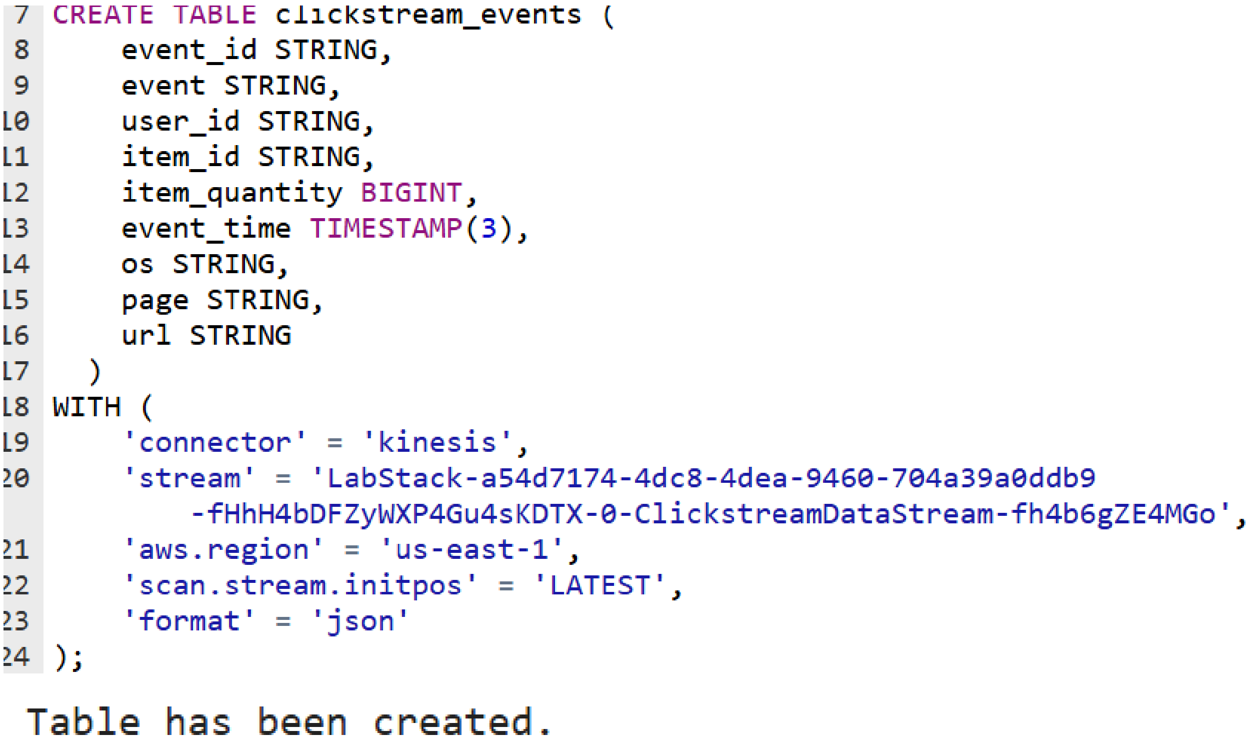
4.0 Open the notebook

Task 4.1: Ingestion - from two sources:

1. From the Kinesis data stream with clickstream data (produced by the clickstream generator)

2. From the Kinesis data stream with catalog data in an Amazon Simple Storage Service (AmazonS3) bucket

We create the in-memory table, clickstream_events, using Kinesis connector

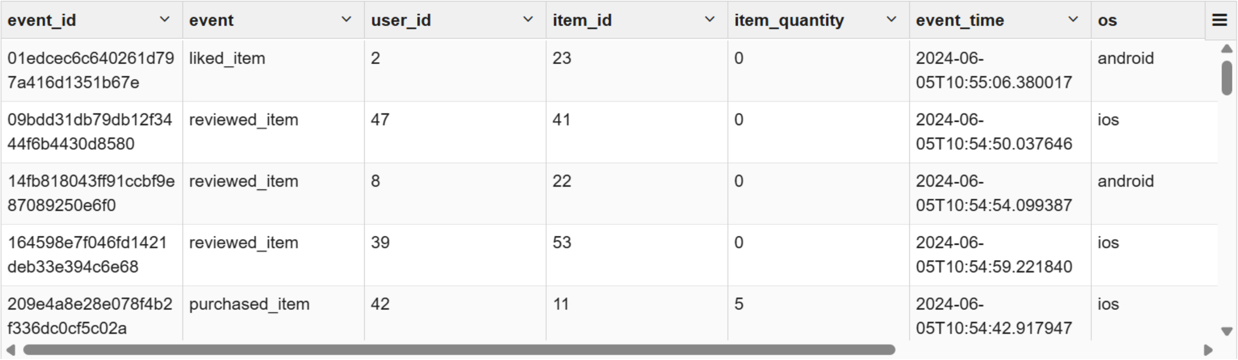


```
CREATE TABLE clickstream_events (
    event_id STRING,
    event STRING,
    user_id STRING,
    item_id STRING,
    item_quantity BIGINT,
    event_time TIMESTAMP(3),
    os STRING,
    page STRING,
    url STRING
  )
WITH (
    'connector' = 'kinesis',
    'stream' = 'LabStack-a54d7174-4dc8-4dea-9460-704a39a0ddb9-fHhH4bDFZyWXP4Gu4sKDTX-0-ClickstreamDataStream-fh4b6gZE4MGo',
    'aws.region' = 'us-east-1',
    'scan.stream.initpos' = 'LATEST',
    'format' = 'json'
);
```

```
Table has been created.
```

Here, we set stream value as the KDS (1) , which collected EC2 data.

Now: view the simulated clickstream data (skipped the code)

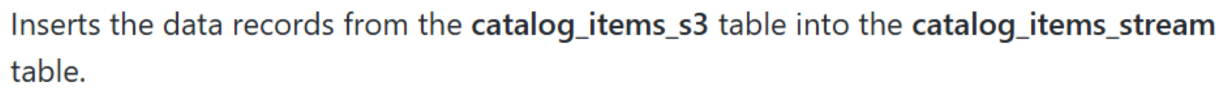


event_id	event	user_id	item_id	item_quantity	event_time	os
01edcec6c640261d797a416d1351b67e	liked_item	2	23	0	2024-06-05T10:55:06.380017	android
09bdd31db79db12f3444f6b4430d8580	reviewed_item	47	41	0	2024-06-05T10:54:50.037646	ios
14fb818043ff91ccbf9e87089250e6f0	reviewed_item	8	22	0	2024-06-05T10:54:54.099387	android
164598e7f046fd1421deb33e394c6e68	reviewed_item	39	53	0	2024-06-05T10:54:59.221840	ios
209e4a8e28e078f4b2f336dc0cf5c02a	purchased_item	42	11	5	2024-06-05T10:54:42.917947	ios

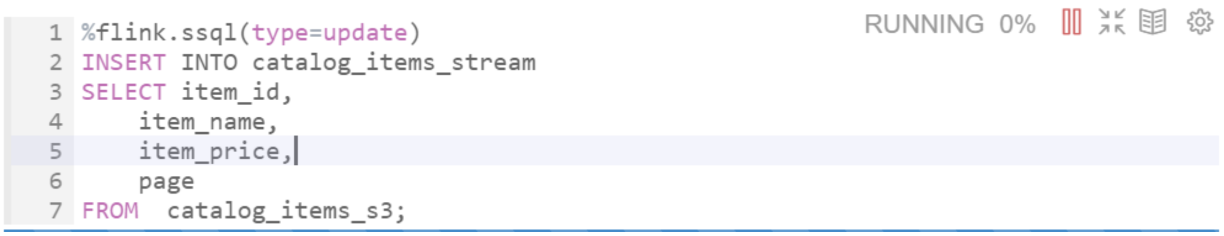
Now, ingest catalog data from S3:

Steps: create table to read S3 data -> create table for KDS(2) -> insert S3 data into KDS(2)

In the above 3 steps:
We directly go to last step of:

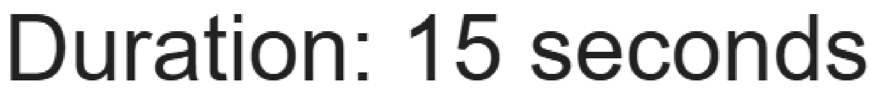


Inserts the data records from the **catalog_items_s3** table into the **catalog_items_stream** table.

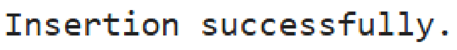


```
%flink.ssql(type=update)
INSERT INTO catalog_items_stream
SELECT item_id,
    item_name,
    item_price,
    page
FROM  catalog_items_s3;
```

Output-

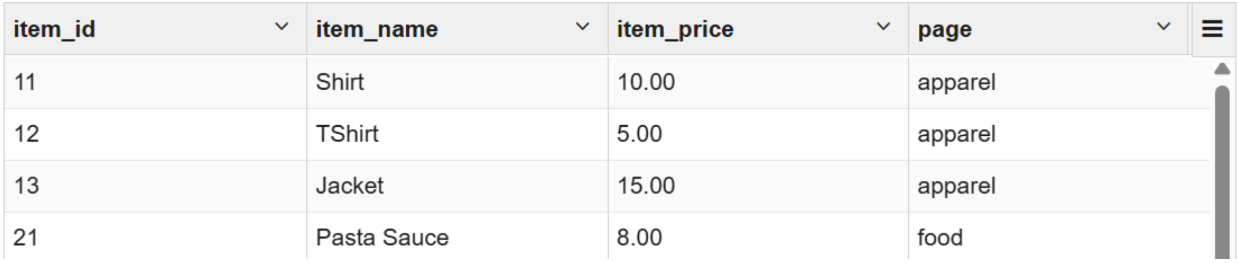


Duration: 15 seconds



```
Insertion successfully.
```

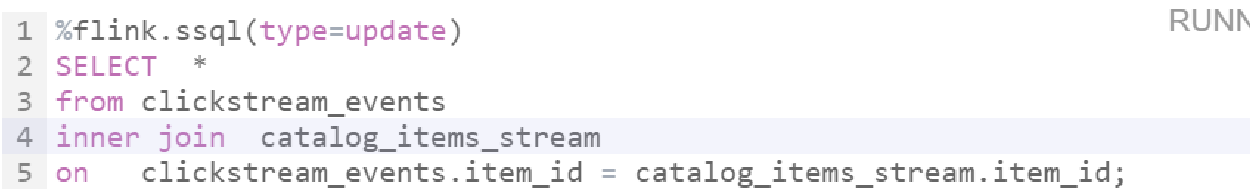
View the data from “catalog_items_stream”



item_id	item_name	item_price	page
11	Shirt	10.00	apparel
12	TShirt	5.00	apparel
13	Jacket	15.00	apparel
21	Pasta Sauce	8.00	food

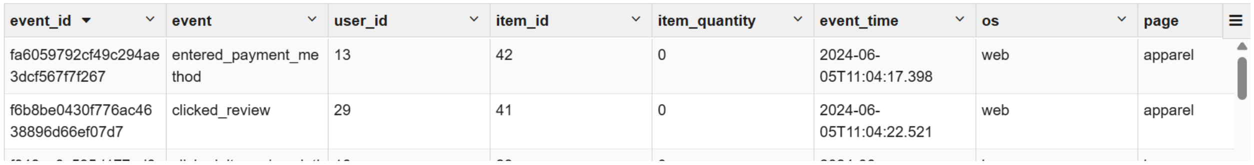
Task 4.2: Data enrichment

enrich the streaming clickstream data with the catalog data available in an S3 bucket.
Done using JOIN



```
%flink.ssql(type=update)
SELECT  *
from clickstream_events
inner join  catalog_items_stream
on   clickstream_events.item_id = catalog_items_stream.item_id;
```

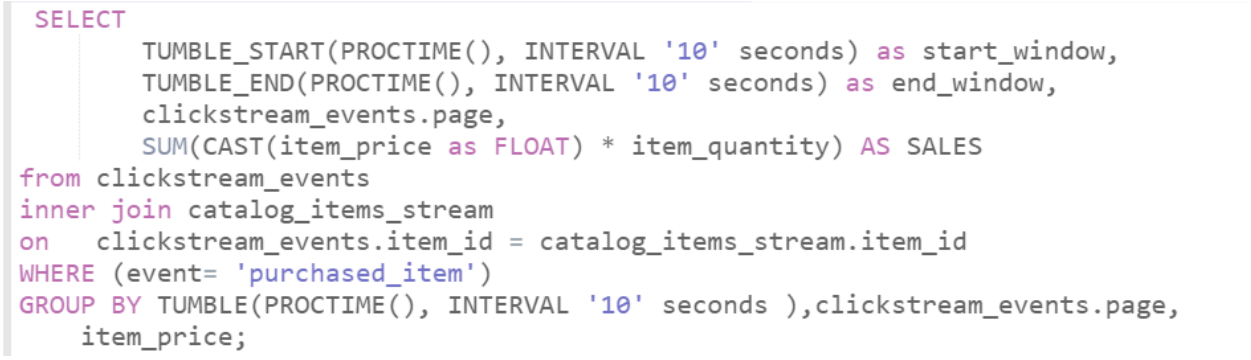
Output-



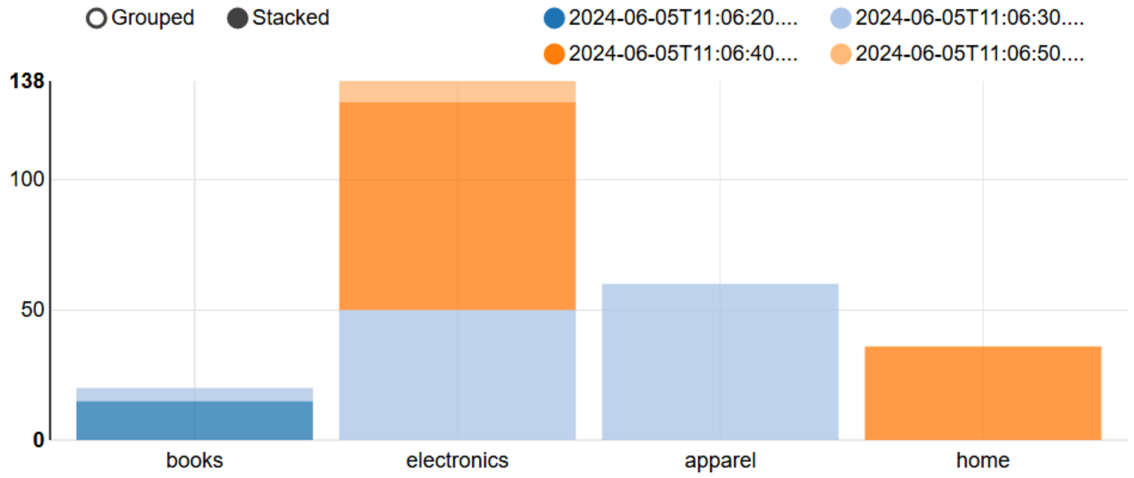
event_id	event	user_id	item_id	item_quantity	event_time	os	page
fa6059792cf49c294ae3dcf567f7f267	entered_payment_method	13	42	0	2024-06-05T11:04:17.398	web	apparel
f6b8be0430f776ac4638896d66ef07d7	clicked_review	29	41	0	2024-06-05T11:04:22.521	web	apparel

Task 4.3: Analysis and visualization

Task = analyze the data to determine the Sales per category in a given time interval.



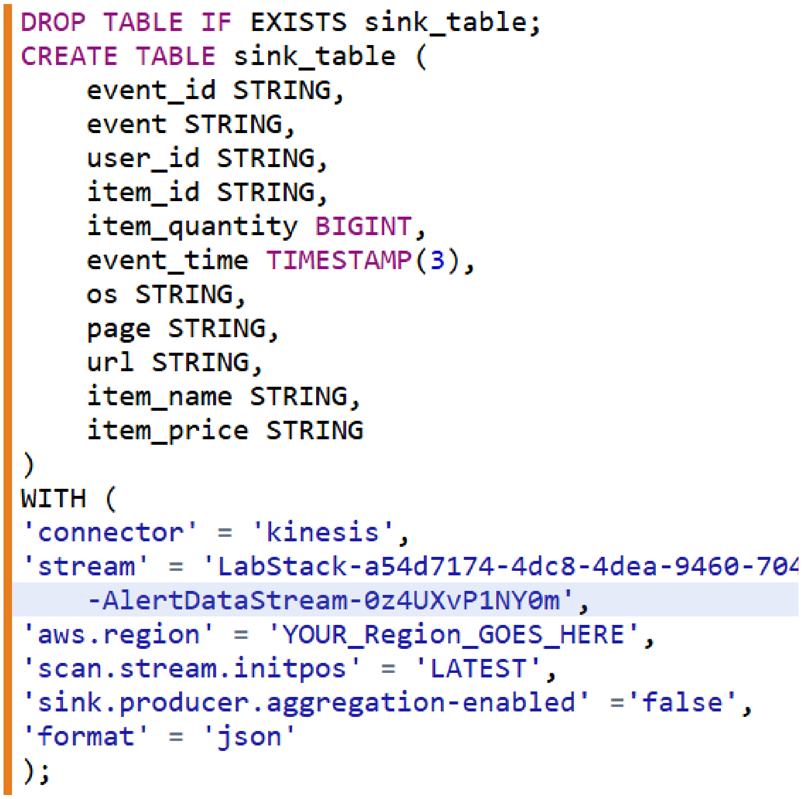
```
 SELECT
        TUMBLE_START(PROCTIME(), INTERVAL '10' seconds) as start_window,
        TUMBLE_END(PROCTIME(), INTERVAL '10' seconds) as end_window,
        clickstream_events.page,
        SUM(CAST(item_price as FLOAT) * item_quantity) AS SALES
from clickstream_events
inner join catalog_items_stream
on   clickstream_events.item_id = catalog_items_stream.item_id
WHERE (event= 'purchased_item')
GROUP BY TUMBLE(PROCTIME(), INTERVAL '10' seconds ),clickstream_events.page,
    item_price;
```



Task 4.4: Output to Kinesis data stream

(From lab:)
you write the output of the analysis to a Kinesis data stream which will be used for further downstream processing.

Create table with output stream value



```
DROP TABLE IF EXISTS sink_table;
CREATE TABLE sink_table (
    event_id STRING,
    event STRING,
    user_id STRING,
    item_id STRING,
    item_quantity BIGINT,
    event_time TIMESTAMP(3),
    os STRING,
    page STRING,
    url STRING,
    item_name STRING,
    item_price STRING
)
WITH (
'connector' = 'kinesis',
'stream' = 'LabStack-a54d7174-4dc8-4dea-9460-704
    -AlertDataStream-0z4UXvP1NY0m',
'aws.region' = 'YOUR_Region_GOES_HERE',
'scan.stream.initpos' = 'LATEST',
'sink.producer.aggregation-enabled' ='false',
'format' = 'json'
);
```