%flink.pyflink

**# Step 1: Create all the Streaming tables**

st\_env.execute\_sql("""DROP TEMPORARY TABLE IF EXISTS tmptickerstream""")

st\_env.execute\_sql("""DROP TEMPORARY TABLE IF EXISTS tmpclickstream""")

st\_env.execute\_sql("""DROP TEMPORARY TABLE IF EXISTS tmpimpressionstream""")

st\_env.execute\_sql("""DROP TABLE IF EXISTS clickstream""")

st\_env.execute\_sql("""DROP TABLE IF EXISTS impressionstream""")

st\_env.execute\_sql("""DROP TEMPORARY TABLE IF EXISTS tmpctrstream""")

st\_env.execute\_sql("""DROP TABLE IF EXISTS ctrstream""")

st\_env.execute\_sql("""DROP TEMPORARY TABLE IF EXISTS tmpdestinationstream""")

st\_env.execute\_sql("""DROP TABLE IF EXISTS destinationstream""")

st\_env.execute\_sql("""DROP TEMPORARY TABLE IF EXISTS tmpanomalydetectionstream""")

st\_env.execute\_sql("""DROP TABLE IF EXISTS anomalydetectionstream""")

st\_env.execute\_sql("""CREATE TEMPORARY TABLE tmptickerstream (

`browseraction` STRING,

`site` STRING,

`rowtime` TIMESTAMP(3) METADATA FROM 'timestamp',

WATERMARK FOR rowtime AS rowtime - INTERVAL '0' SECOND

)

WITH (

'connector' = 'kinesis',

'stream' = 'tickerstream',

'aws.region' = 'us-east-1',

'scan.stream.initpos' = 'TRIM\_HORIZON',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TEMPORARY TABLE tmpclickstream (

`rowtime` TIMESTAMP(3),

`clickcount` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'clickstream',

'aws.region' = 'us-east-1',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TEMPORARY TABLE tmpimpressionstream (

`rowtime` TIMESTAMP(3),

`impressioncount` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'impressionstream',

'aws.region' = 'us-east-1',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TABLE clickstream (

`rowtime` TIMESTAMP(3),

`clickcount` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'clickstream',

'aws.region' = 'us-east-1',

'scan.stream.initpos' = 'TRIM\_HORIZON',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TABLE impressionstream (

`rowtime` TIMESTAMP(3),

`impressioncount` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'impressionstream',

'aws.region' = 'us-east-1',

'scan.stream.initpos' = 'TRIM\_HORIZON',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TEMPORARY TABLE tmpctrstream (

`time` BIGINT,

`ctr` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'ctrstream',

'aws.region' = 'us-east-1',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TABLE ctrstream (

`time` BIGINT,

`ctr` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'ctrstream',

'aws.region' = 'us-east-1',

'scan.stream.initpos' = 'TRIM\_HORIZON',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TEMPORARY TABLE tmpdestinationstream (

`time` BIGINT,

`ctr` DOUBLE,

`score` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'destinationstream',

'aws.region' = 'us-east-1',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TABLE destinationstream (

`time` BIGINT,

`ctr` DOUBLE,

`score` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'destinationstream',

'aws.region' = 'us-east-1',

'scan.stream.initpos' = 'TRIM\_HORIZON',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TEMPORARY TABLE tmpanomalydetectionstream (

`rowtime` TIMESTAMP,

`ctrpercent` DOUBLE,

`anomaly\_score` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'anomalydetectionstream',

'aws.region' = 'us-east-1',

'format' = 'json'

)""")

st\_env.execute\_sql("""CREATE TABLE anomalydetectionstream (

`rowtime` TIMESTAMP,

`ctrpercent` DOUBLE,

`anomaly\_score` DOUBLE

)

WITH (

'connector' = 'kinesis',

'stream' = 'anomalydetectionstream',

'aws.region' = 'us-east-1',

'scan.stream.initpos' = 'TRIM\_HORIZON',

'format' = 'json'

)""")

%flink(parallelism=1)

**//Step 2: Create the UDF (User Defined Function)**

import software.amazon.flink.example.RandomCutForestUDF

stenv.registerFunction("RANDOM\_CUT\_FOREST", new RandomCutForestUDF())

%flink.ssql(type=update)

**-- Step 3: View the Ticker Stream Data in real-time**

select \* from tmptickerstream;

%flink.ssql(type=update, parallelsim=1)

**-- Step 4: Generate the Impression Stream Data in real-time based on the browser action value in the Ticker stream**

INSERT INTO tmpimpressionstream

SELECT

TUMBLE\_START(rowtime, INTERVAL '10' SECOND) AS rowtime,

COUNT(\*) AS impressioncount

FROM tmptickerstream

WHERE browseraction='Impression'

GROUP BY

TUMBLE(rowtime, INTERVAL '10' SECOND);

%flink.ssql(type=update, parallelsim=1)

**-- Step 5: Generate the Click Stream Data in real-time based on the browser action value in the Ticker stream**

INSERT INTO tmpclickstream

SELECT

TUMBLE\_START(rowtime, INTERVAL '10' SECOND) AS rowtime,

COUNT(\*) AS clickcount

FROM tmptickerstream

WHERE browseraction='Click'

GROUP BY

TUMBLE(rowtime, INTERVAL '10' SECOND);

%flink.ssql(type=update, parallelsim=1)

**-- Step 6: Generate the Click Through Rate based on the ratio of Click vs Impression in real-time**

INSERT INTO tmpctrstream

SELECT unix\_timestamp(cast(t1.rowtime as string)) as `time`, ROUND((t2.clickcount / t1.impressioncount \* 100), 1) as ctr

FROM impressionstream t1 INNER JOIN

clickstream t2

ON unix\_timestamp(cast(t1.rowtime as string)) = unix\_timestamp(cast(t2.rowtime as string));

%flink.ssql(type=update)

**-- Step 7: Analyze the Click Through Rate in real-time**

select \* from ctrstream;

%flink.ssql(type=update, parallelsim=1)

**-- Step 8: Generate the Anomaly Score in real-time with User Defined Function RANDOM\_CUT\_FOREST**

INSERT INTO tmpdestinationstream

SELECT `time`, ctr, RANDOM\_CUT\_FOREST(cast(ctr as float)) as score

FROM ctrstream;

%flink.ssql(type=update, parallelsim=1)

**-- Step 9: Create a Stream with the Anomaly data in real-time**

INSERT INTO tmpanomalydetectionstream

SELECT to\_timestamp(from\_unixtime(`time`)) as rowtime, ctr as ctrpercent, score as anomaly\_score

FROM destinationstream

WHERE score>2;

%flink.ssql(type=update)

**-- Step 10: View and Analyze the Anomalies in real-time**

select \* from anomalydetectionstream;