
Northeastern University

College of Professional Studies

Final Project Report

by

Group 8

Anubhav Saha

Priyanka Nagesh Adiga

Introduction to Cloud Computing

Professor: Anjum Biswas

02/26/2021

## Introduction

Club feast is a beverage-delivery startup, based in the San Francisco bay area. They connect their customers with their favorite stores and provide affordable beverage options by pre-scheduling the orders. Customers need to book their orders at least 24 hours before the scheduled delivery window. In a short period of time, they have partnered with a number of stores and are witnessing an exponential growth in their customer base. They know the importance of data and the importance of making data-driven decisions. To achieve this, they are now focusing on developing dashboards to analyze the user behavior and to monitor key performance metrics in real-time.

For now, the data that is being collected from the website is stored in a local repository and anyone who wants to see the current trends will have to pull the data, run scripts to normalize the data and then create dashboards. This process takes 1-2 days of time and by the time the reports are ready to be published, the data has already become obsolete. To overcome this issue, they need help in setting up a cloud infrastructure which can help them to efficiently manage and store their data and the dashboards can be built as and when required.

## Project Proposal

In this project, we are planning to analyze the clickstream data generated from the website of the company Club feast and detect the user behavior by analyzing the sequence of clicks they are making, the amount of time they are spending on the landing page and whether that leads to an order. Our goal is to help the company maximize the orders coming from the website, and to ensure that the page visit leads to an order. By analyzing visitor behavior, we can also understand their preferences and make them turn into active subscribers.

By tracking the user behavior real time, we can customize our suggestions in order to increase the likelihood of that user ending up making an order from the website and thus improving the turnover. We can also perform A/B testing of the landing webpage and see which version of it improves the turnover. In order to implement this, we will be using AWS services such as Amazon Kinesis, Kinesis Data Analytics, Kinesis Data Firehose, Amazon Simple Storage Service (Amazon S3), Amazon Athena, IAM, Amazon Lambda, Amazon Glue and Amazon QuickSight.

## Background and Context

The Club feast website has enormous user traffic and as a way of improving the business it is important to analyze the user behavior when the users are surfing through the website. This can only be achieved by analyzing the click stream data of the users. This involves analyzing series of events performed by the users. These events can be anything ranging from in which part of the website the user spends most of his time, where the user begins the navigation, and how the navigation halts. Tracking of all these events for big number of users can be difficult and will require a robust mechanism that will effectively analyze the click stream data in real time in order to allow the business to action as and when required. These events are analyzed by first creating sessions for them or sessionization needs to be performed prior to analyzing the click stream data.

AWS offers services such as Amazon Kinesis which makes it easy to collect, process as well as analyze real time data and streaming the data so that it becomes easier to acquire timely insights. Furthermore, it offers key capabilities to cost effectively process the streaming data from the club feast website any scale and also provides us the flexibility of choosing the tools that best suit the requirements of our application. Another key advantage of performing sessionization using Amazon kinesis is that it provides a relatively lower data latency as compared to performing

sessionization using AWS Glue or Amazon EMR. Lastly, a lambda function can be used in order to trigger any events real time. For example, when a user visits the website and exists in less than 20 seconds can be recorded as an event and this event can be used as a triggering point to a lambda function that can further use this information in other machine learning models, based on our application requirements.

As session as we know is short lived and is an interactive exchange between a device and an end user in our case and these interactions result in a series of events that occur in a sequence and a session consists of a start and an end. A start and end of a session can be can be difficult to determine and are often defined by a time period without a relevant event associated with a user or a device. In this project we will be grouping the events of a specific user using the user ID. Other elements such as client IP or a machine ID can also be considered which allows us to separate the sessions that occur on different devices.

## Technology solution and Analysis

### High Level overview of the project Implementation

In this project in order to analyze the click stream data we are using Amazon Kinesis data streams in order to capture the click stream data and this is followed by using Kinesis Data analytics in order to build and analyze the sessions. The aggregated analytics is used in order to trigger the real time events on the lambda function and this is followed by sending the data to Kinesis Data Firehose. Further, the Kinesis data firehose sends the data thus received to an Amazon S3 bucket where the data is ingested into a table by an AWS Glue Crawler and is made available for running the queries with the help of Amazon Athena. The below diagram shows the end-end to sessionization solution that is being implemented in this project.

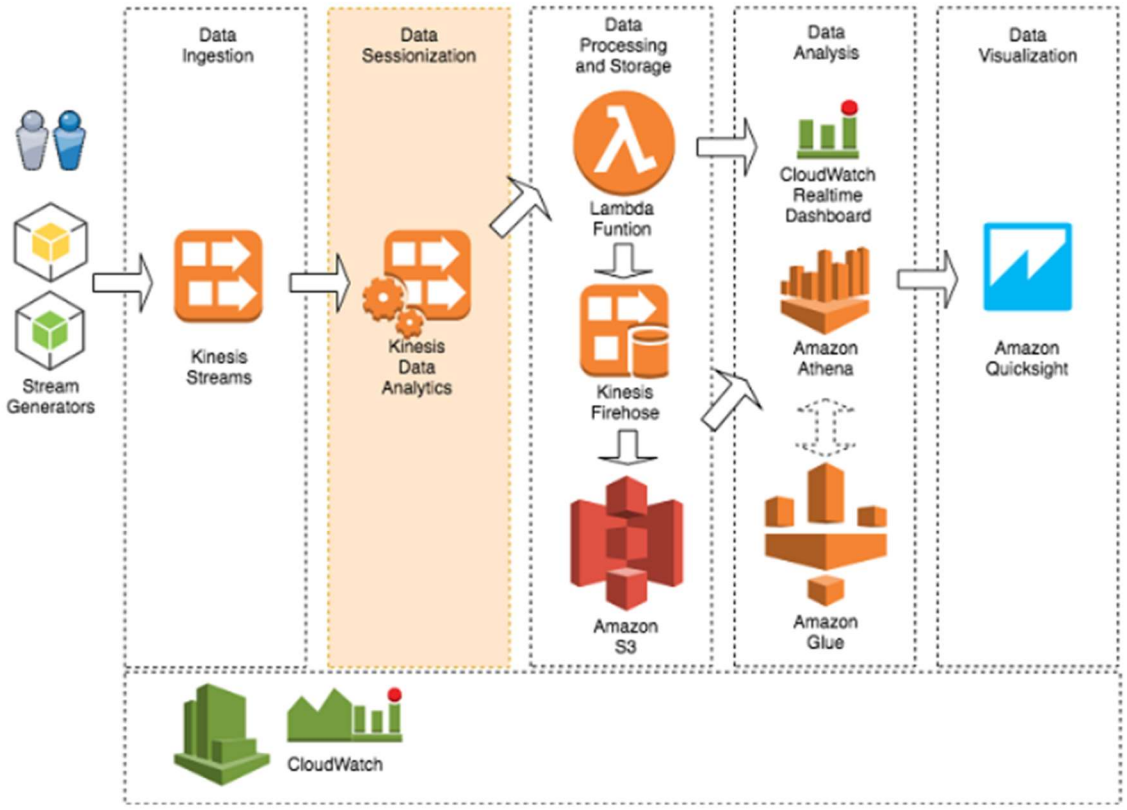



Image Retrieved from Hugo Rozestraten, (February 07, 2019), Create real-time clickstream sessions and run analytics with Amazon Kinesis Data Analytics, AWS Glue, and Amazon Athena. (1)

**Data Ingestion**

Here we are using Kinesis data streams in order to build custom applications that process or analyze streaming data. Kinesis data stream can continuously capture and store terabytes of data per hour from large number of data sources.

**Data Sessionization**

In order to process the streaming data in real time Kinesis data analytics offers one of the easiest ways and using standard SQL. With the help of Kinesis data analytics, we will be querying the streaming data in order to be able to gain actionable insights and allows us to respond to our business and customer needs promptly.

**Data Processing and Data Storage**

With the help of an AWS lambda function, the sessionization stream is read from kinesis data analytics. In this project, the lambda function is used to trigger two events first event to trigger the real-time dashboard in Amazon Cloud watch and the second event to persists data with Kinesis data firehose.

**Data Analysis**

AWS Glue which is a fully managed ETL service allows us to categorize, clean and enrich our data. And in this project, it is used to crawl Amazon S3 as well as build or update metadata definition for the tables in Amazon Athena.

More insights on the implementation of the project have been explained in the below sections.

**Lambda Click Stream generator**

In order to generate the workload, we are using python lambda function with random values simulating a beverage selling application. In order to achieve this, we will be creating a Lambda function from the AWS console and use the programming language Python to create a payload generator, which will generate our clickstream data using the column names that we provide and arguments that we pass, it will use random functions to pick choices from those arguments that we provide and then schedule it using CloudWatch Events.

**Using window SQL functions in kinesis data analytics**

Next, we would be grouping sessions that would let us combine all the events from a particular user and device that showed up in the data in a given time interval. Amazon Kinesis Data Analytics SQL queries in the application code and executes it continuously over in-application streams. We need to specify bounded queries using a window defined in terms of time or rows, and these queries are known as window SQL functions. Using the Kinesis Data Analytics SQL, we will be generating our stream.

**Automated deployment with AWS CloudFormation**

For the real-time sessionization, we will be making use of AWS CloudFormation which would include creating a Stack and then pass the New Kinesis stream that we generated. Along with the Amazon Kinesis Configuration, we would also be passing our Amazon S3 destination configuration, such as the name of our S3 bucket where our sessionization will be put, how many seconds do we want our data to buffer, before it gets added to our S3 bucket in a batch, also how much should this buffer size be, and finally what unit should we base our sessions in.

After this deployment, we will be able to start processing our stream data generated by Kinesis streams. We can check the CloudWatch real-time dashboard that would visualize our stream in terms of session duration and session events. We can then open the AWS Glue console and run a crawler to fetch the data so that we can analyze it using Amazon Athena. In Athena we can write our SQL queries to create views and run any kind of analysis we want. The results that we get will be finally visualized using Amazon QuickSight which is a visualization tool using which we will be creating our final dashboard with all the results.

## Major Goals

In this project, our major goal is to analyze the clickstream data generated from the website of the company Club feast and to understand the user behavior by clickstream data analytics, which essentials tracks the events generated using the user inputs (clicks) and the amount of time they spend on each page they visit and the turnover from that visit. Our goal is to help the company maximize the orders coming from the website. This can be achieving by maximizing the turnover from each customer visit, which will be possible by analyzing the visitor behavior. This will also assist us in creating custom recommendations and offers that would aid in the turnover.

To achieve this, we are basically simulating the clickstream data using the tools that Amazon provides us in their web services such as they Lambda function to create the data and generate the payload. Next, we will be using Amazon Kinesis to generate our streams, which will then be dumped in batches in regular intervals of seconds in an S3 bucket which would be the dummy clickstream data. Once we have this data, we will be using Amazon Glue to create a crawler that would crawl through the S3 bucket and create a schema in the database. We can then use Amazon Athena to query the database and complete our goal of the analysis of the clickstream data by observing the user behavior, which will be aided by Amazon QuickSight to visualize and display it in the final dashboard that would aid the business decision makers of our company to make data driven decisions to improve the turnover and generate more revenue to complete our project goal.

## Review and reflection

From this project, we would be learning how to simulate the clickstream data using various tools that cloud services such as and especially Amazon Web Services provide us. Since we are generating a dummy data and not using any public APIs to fetch any data from a live website such

as Facebook for developers, this would be a good learning experience for us to understand how the data is generated and stored in systems and what are all the different operations that take place such as sessionization, event generation, etc.

After we get our dummy clickstream data generated, the next major learning would be how to create a database and perform a sort of extraction of data from the S3 bucket using the crawlers that AWS provides us in Amazon Glue which helps us by automatically fetching the schema of our database. This is a very useful tool that would help us get our relational database that we are used to and finally use Amazon Athena to write our SQL queries and perform the data analysis that we have learned in our courses.

Finally, after the ETL operations and the data analysis, one of the most important steps is the final visualization of our results, for which Amazon provides us their Business Intelligence tool Amazon QuickSight, which is similar to Tableau that we all are familiar with. It helps us use the results from our other AWS services and create a custom dashboard that we can present to the business decision makers that would contain the visualizations that would aid in the data driven decision making.

## Limitations

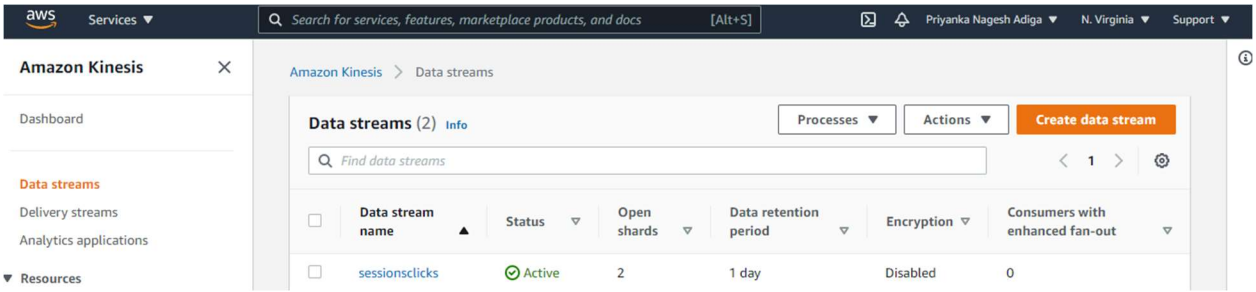
The major limitation in our project would be I think in the clickstream data generation process. Lack of real world experience in clickstream analytics might limit our imagination of different fields that maybe are a common occurrence in the clickstream data, since our data generator would be based on the fields and column names that we provide, and we would have to imagine these sorts of fields that we think show up in a clickstream data.

To overcome this limitation, we can certainly explore the internet and read articles about different clickstream analytics projects and look up the data structure of such data so that we do not miss the crucial fields that are required in these sorts of data and thus we would be able to simulate our clickstream data to as closer to a real clickstream data as possible, which would make our analysis of such data more valuable and effective, when it comes to deploying this project on a real world website's clickstream data.

## Implementation

**Step 1:** Clickstream data generation

To generate our dummy clickstream data, we have first created a Data Stream in Amazon Kinesis called sessionclicks as shown below:

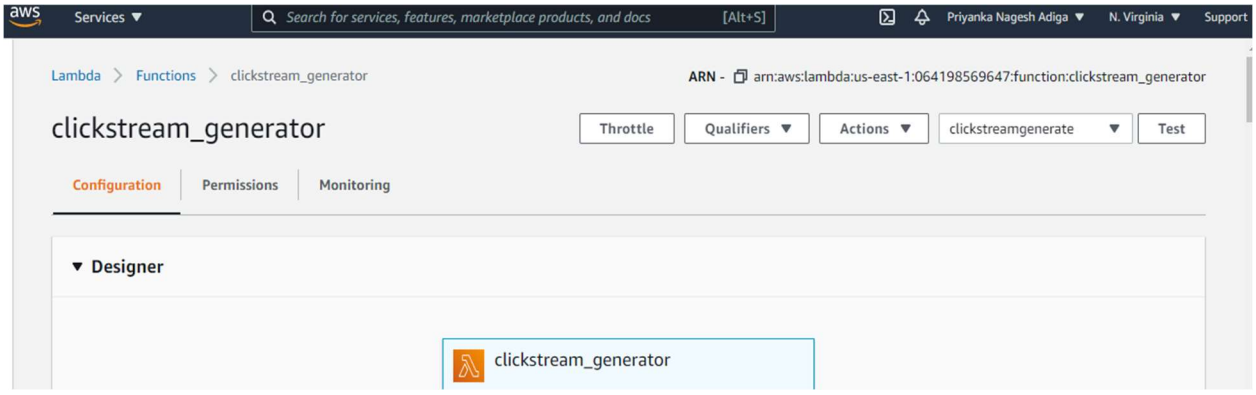


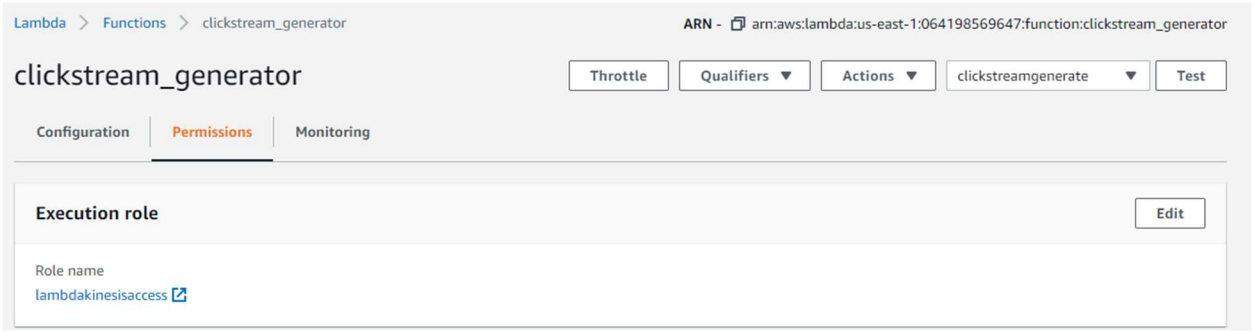

Now to generate the payload, we are using a Lambda function and invoking the put_record function to dump randomly generated data to the Kinesis data stream which we created.

We named our Lambda function as clickstream_generator, where first we imported the necessary libraries needed for our Python code. Then instantiated our Kinesis client using the boto3 library which will be used in the lambda_handler function to invoke the put_record function.

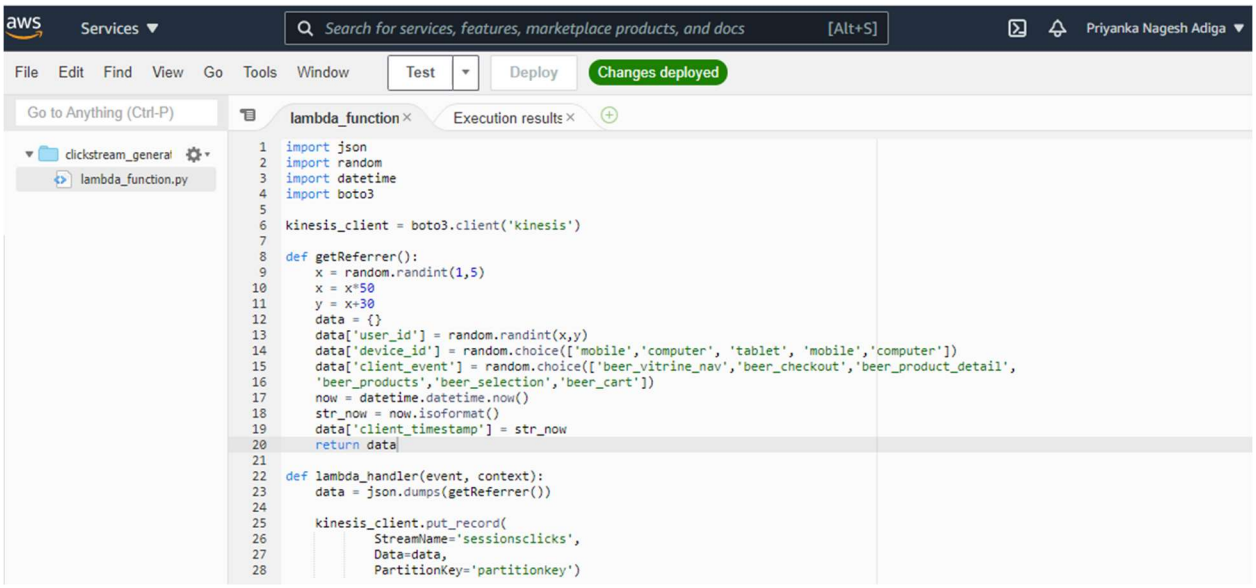
For clickstream data, the important elements are the user, the device the user is using, the events that user is causing through that device and at what time those events are occurring. Therefore, we defined the id fields for each of the above elements and passed parameters to the random function to randomly generate user ids, devices, client events and timestamps, which form our dummy data which we can dump on our data stream.

Next, our lambda_handler function makes use of CloudWatch events to dump the key value pairs of the dummy data generated in the previous function on the sessionclicks data stream in Amazon Kinesis using the kinesis_client which was instantiated earlier as shown below:

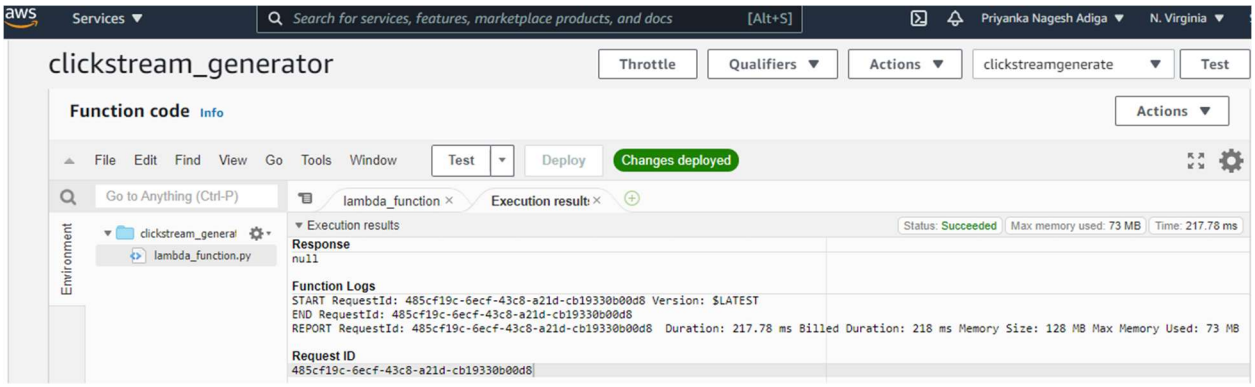



It is important that we add the appropriate permissions to this Lambda function so that Lambda can make use of Kinesis agent without errors. This can be achieved by IAM console and creating a role and attaching Kinesis Write access policy.



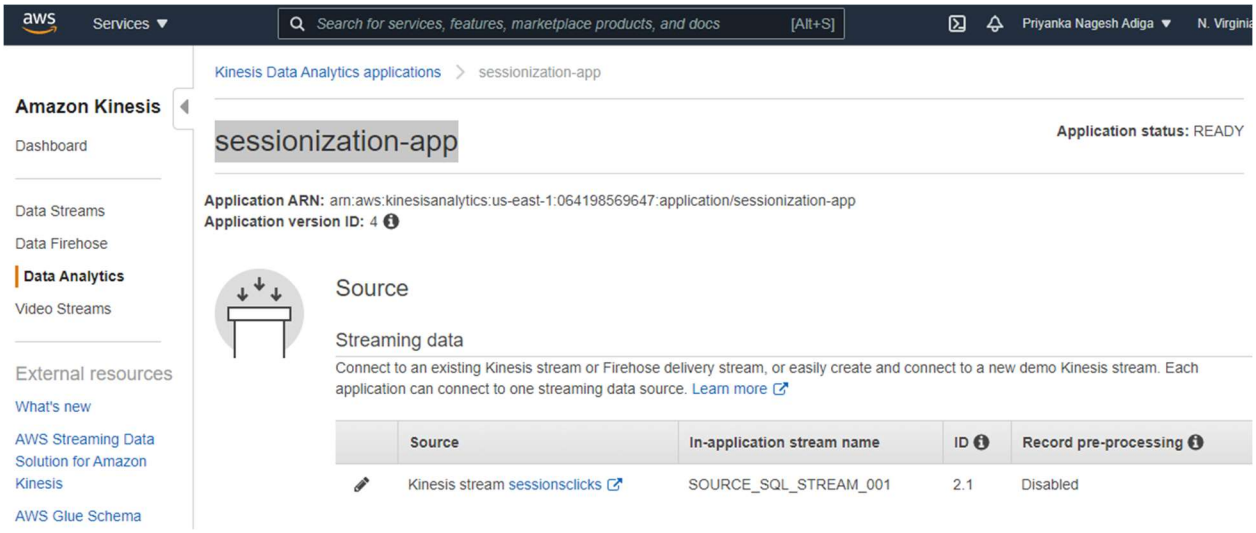

Upon execution by clicking Test, the execution results display the request ID and the logs, including the duration of the run and the duration for which it was billed in Amazon.




**Step 2**. Clickstream data processing by Sessionization:

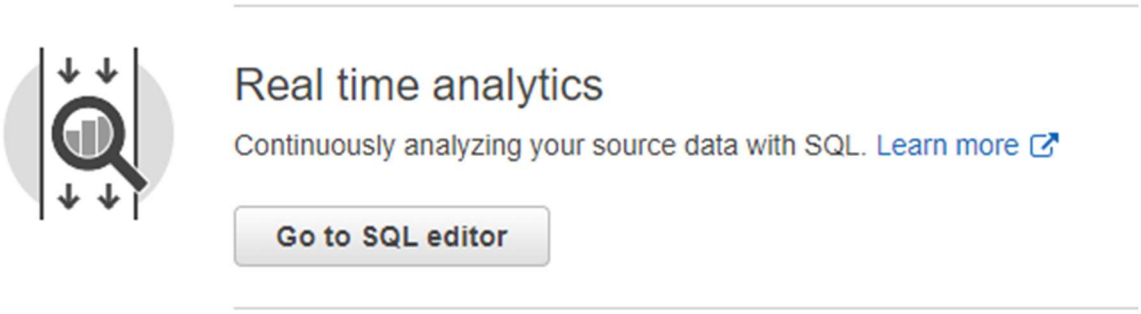
Now that we have our clickstream data sent to our Data stream sessionclicks in Kinesis, we need to process these clickstream data in order to form sessions, which is called sessionization.

To achieve this, we have created a Kinesis Data Analytics application called sessionization-app and connected our data input stream sessionclicks to it as shown below.

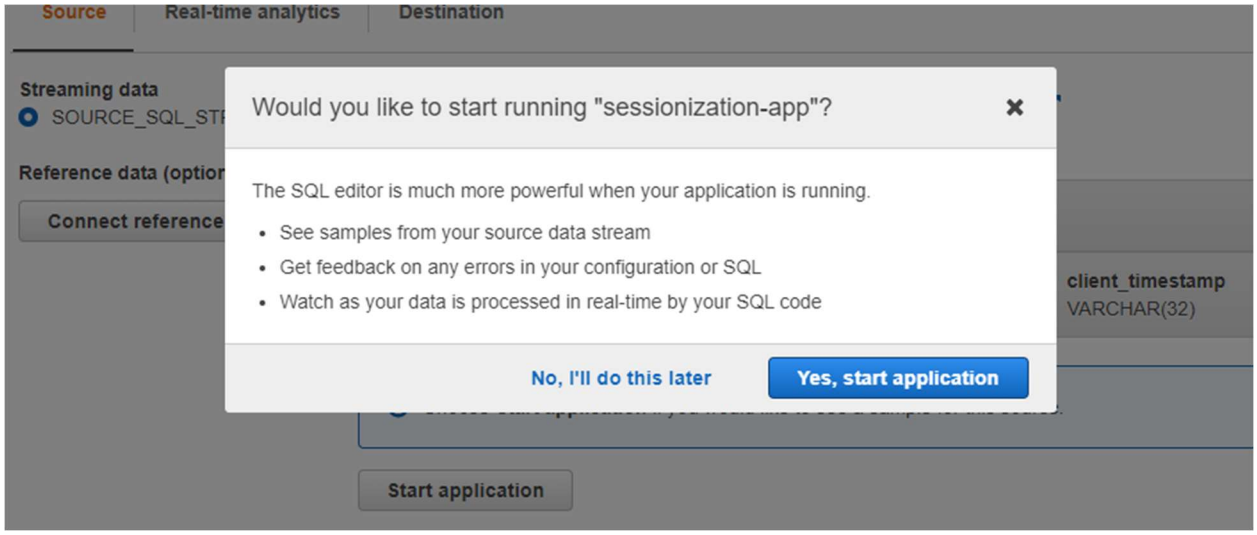



The reason we chose Kinesis Data Analytics for processing our clickstream data is because performing sessionization in Kinesis Data Analytics takes less time and gives us a lower latency between the session generation.

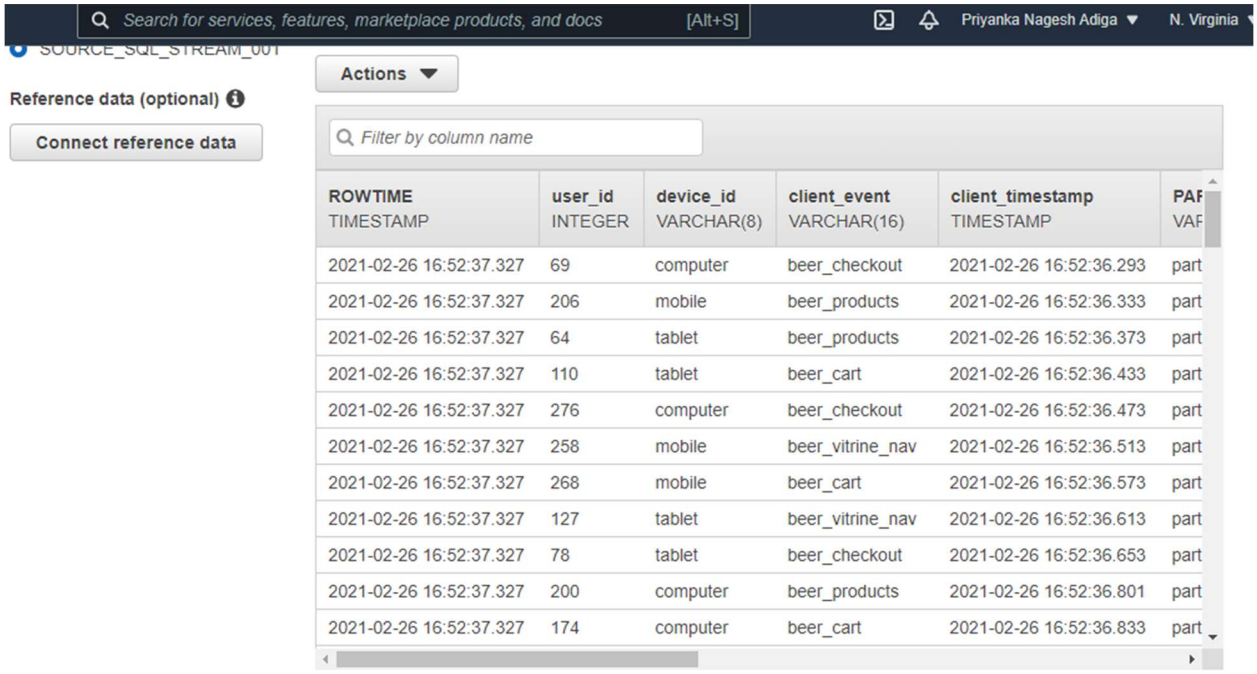
Now to run our application and process our data using SQL, we can navigate to the SQL editor in the Real time analytics section as shown below:



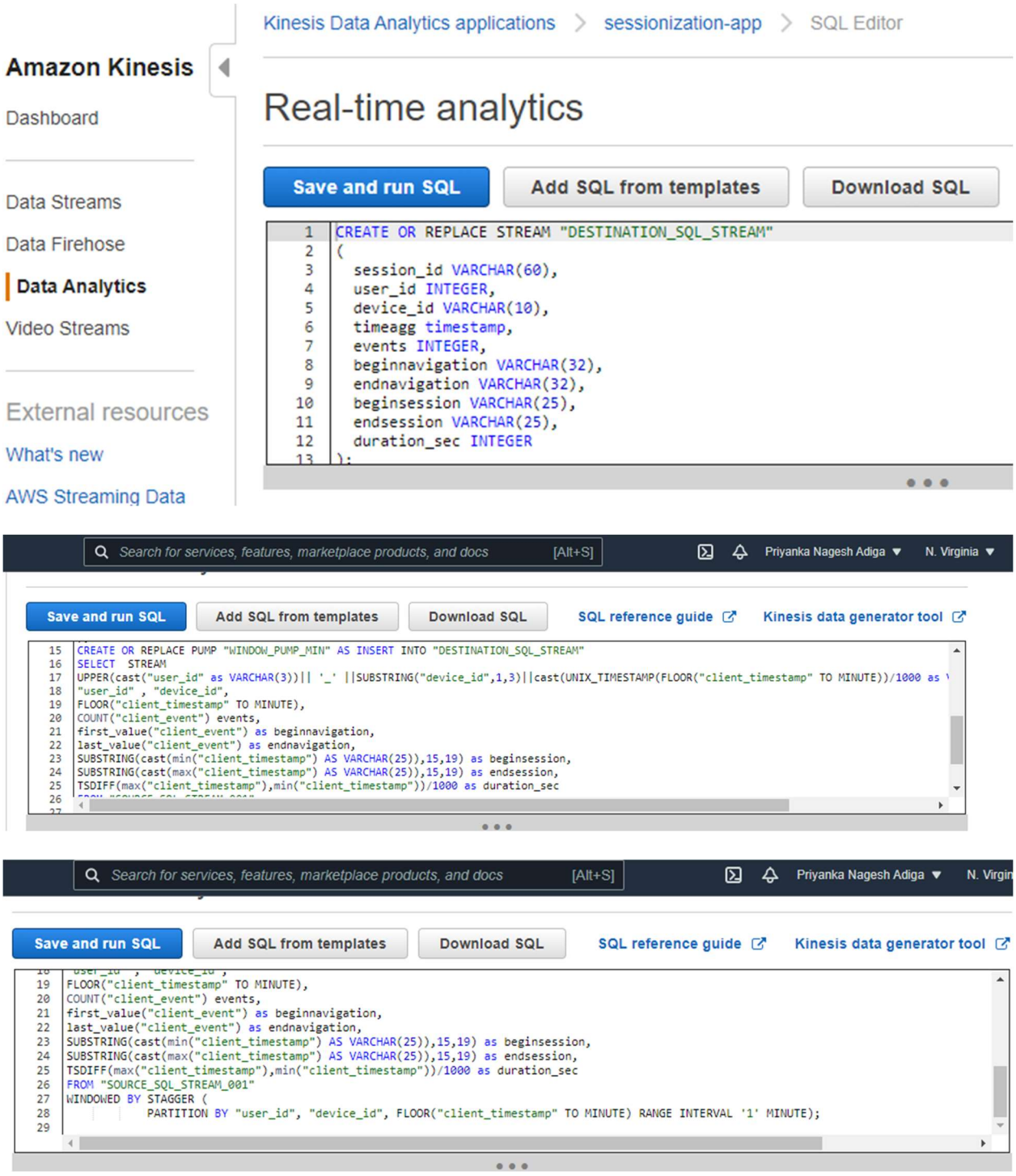

It immediately gives a pop up asking if we want to start the application, where we can click yes.



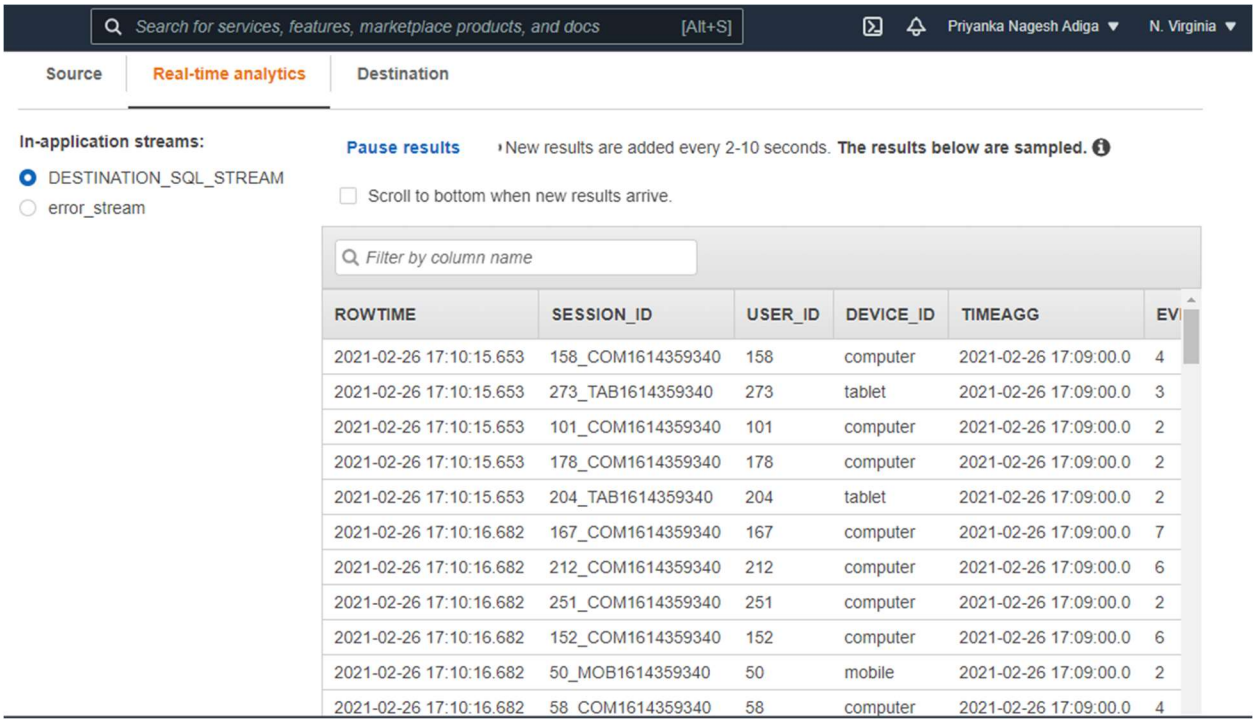

After the Application starts, we can our clickstream data from our source data stream:



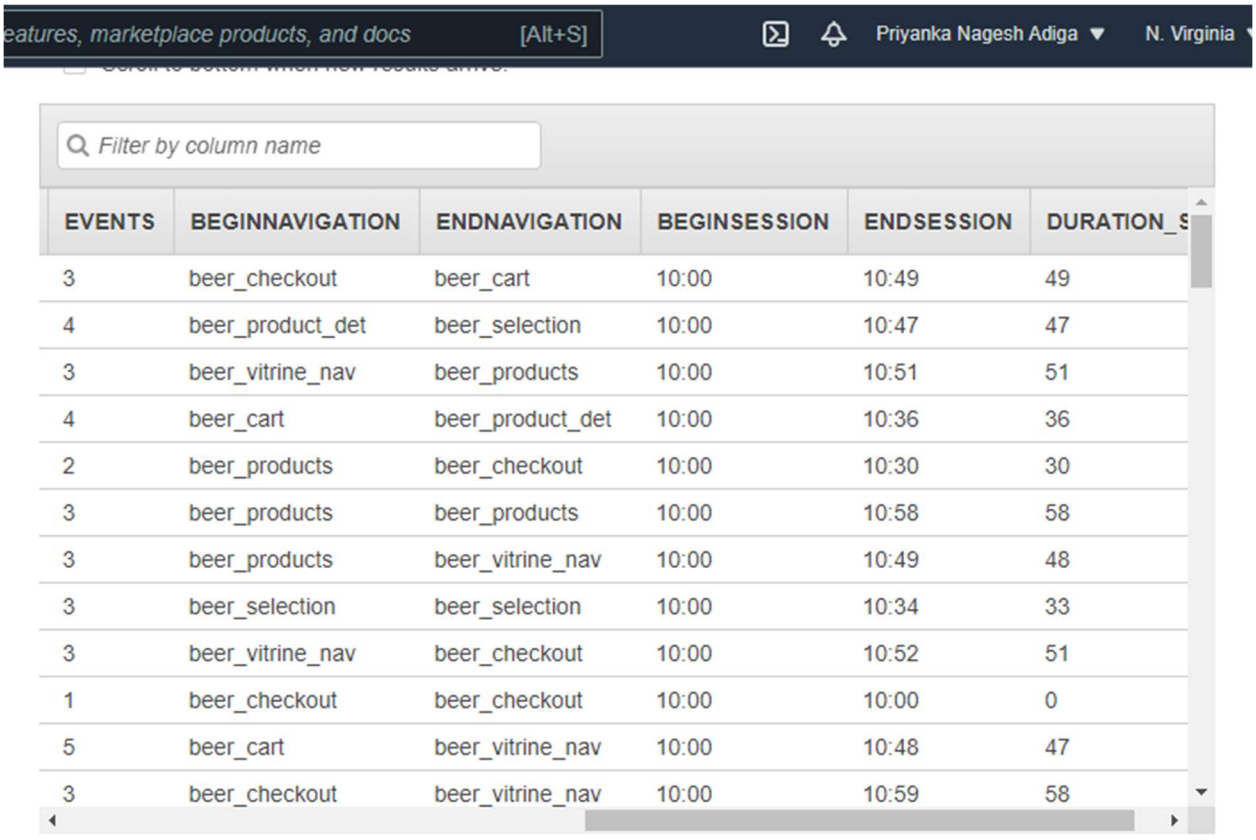

However, we want to group them into sessions, to achieve that we will create a destination stream using the following query that makes use of window SQL functions to group our clickstream data into sessions as shown below:




Amazon Kinesis Data Analytics SQL queries in our application code execute continuously over in-application streams. The specified bounded queries using a window defined in terms of time or rows helps us group the sessions based on lag time.



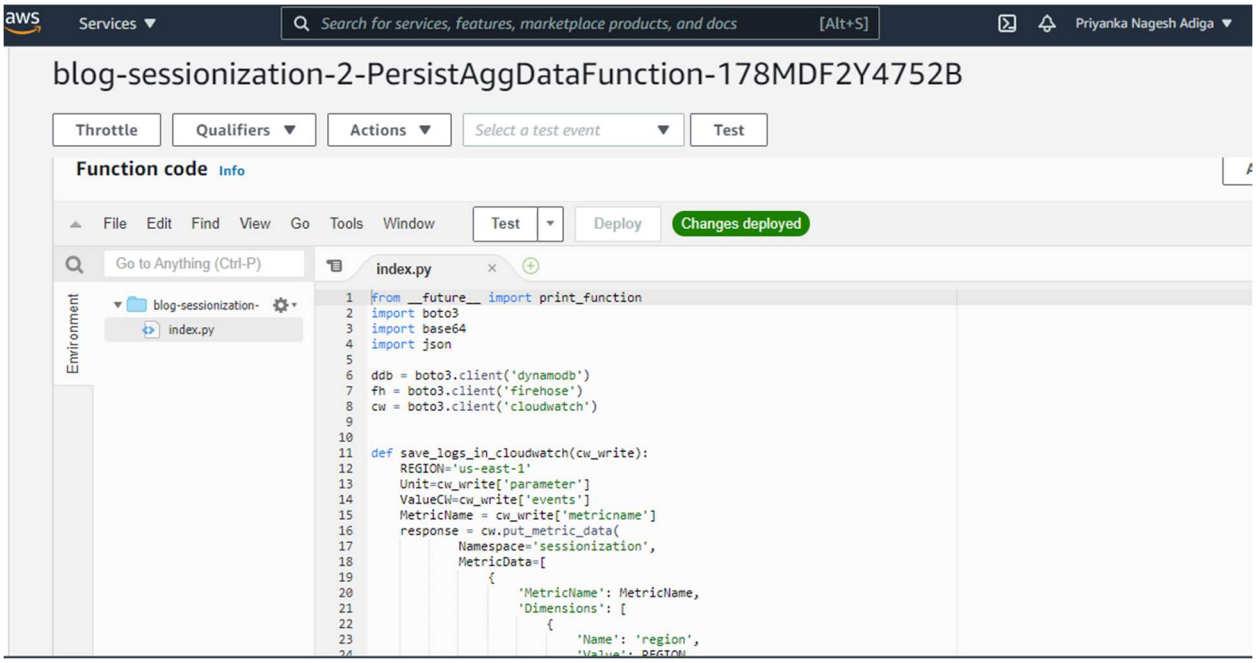

| ROWTIME | SESSION_ID | USER_ID | DEVICE_ID | TIMEAGG | EV |
|---|---|---|---|---|---|
| 2021-02-26 17:10:15.653 | 158_COM1614359340 | 158 | computer | 2021-02-26 17:09:00.0 | 4 |
| 2021-02-26 17:10:15.653 | 273_TAB1614359340 | 273 | tablet | 2021-02-26 17:09:00.0 | 3 |
| 2021-02-26 17:10:15.653 | 101_COM1614359340 | 101 | computer | 2021-02-26 17:09:00.0 | 2 |
| 2021-02-26 17:10:15.653 | 178_COM1614359340 | 178 | computer | 2021-02-26 17:09:00.0 | 2 |
| 2021-02-26 17:10:15.653 | 204_TAB1614359340 | 204 | tablet | 2021-02-26 17:09:00.0 | 2 |
| 2021-02-26 17:10:16.682 | 167_COM1614359340 | 167 | computer | 2021-02-26 17:09:00.0 | 7 |
| 2021-02-26 17:10:16.682 | 212_COM1614359340 | 212 | computer | 2021-02-26 17:09:00.0 | 6 |
| 2021-02-26 17:10:16.682 | 251_COM1614359340 | 251 | computer | 2021-02-26 17:09:00.0 | 2 |
| 2021-02-26 17:10:16.682 | 152_COM1614359340 | 152 | computer | 2021-02-26 17:09:00.0 | 6 |
| 2021-02-26 17:10:16.682 | 50_MOB1614359340 | 50 | mobile | 2021-02-26 17:09:00.0 | 2 |
| 2021-02-26 17:10:16.682 | 58_COM1614359340 | 58 | computer | 2021-02-26 17:09:00.0 | 4 |

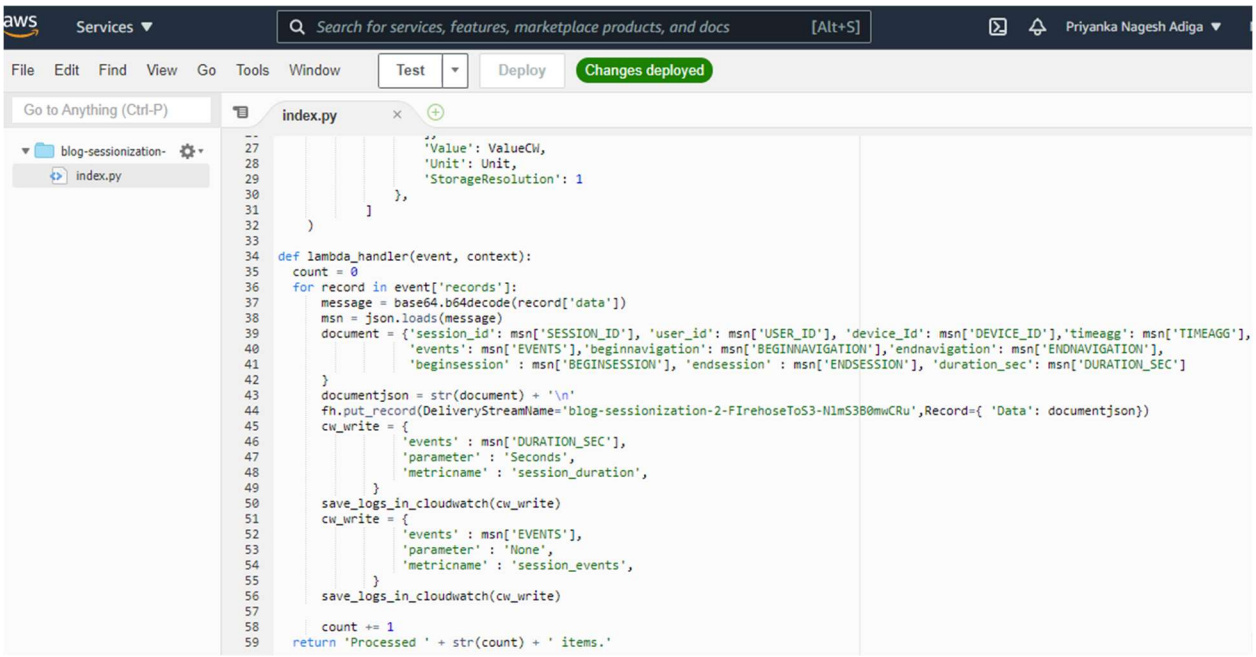



| EVENTS | BEGINNAVIGATION | ENDNAVIGATION | BEGINSESSION | ENDSESSION | DURATION_S |
|---|---|---|---|---|---|
| 3 | beer_checkout | beer_cart | 10:00 | 10:49 | 49 |
| 4 | beer_product_det | beer_selection | 10:00 | 10:47 | 47 |
| 3 | beer_vitrine_nav | beer_products | 10:00 | 10:51 | 51 |
| 4 | beer_cart | beer_product_det | 10:00 | 10:36 | 36 |
| 2 | beer_products | beer_checkout | 10:00 | 10:30 | 30 |
| 3 | beer_products | beer_products | 10:00 | 10:58 | 58 |
| 3 | beer_products | beer_vitrine_nav | 10:00 | 10:49 | 48 |
| 3 | beer_selection | beer_selection | 10:00 | 10:34 | 33 |
| 3 | beer_vitrine_nav | beer_checkout | 10:00 | 10:52 | 51 |
| 1 | beer_checkout | beer_checkout | 10:00 | 10:00 | 0 |
| 5 | beer_cart | beer_vitrine_nav | 10:00 | 10:48 | 47 |
| 3 | beer_checkout | beer_vitrine_nav | 10:00 | 10:59 | 58 |

The data continues to stream as long as the Application is running. The output of the SQL functions created our streaming data with session ID and aggregate time as shown above.

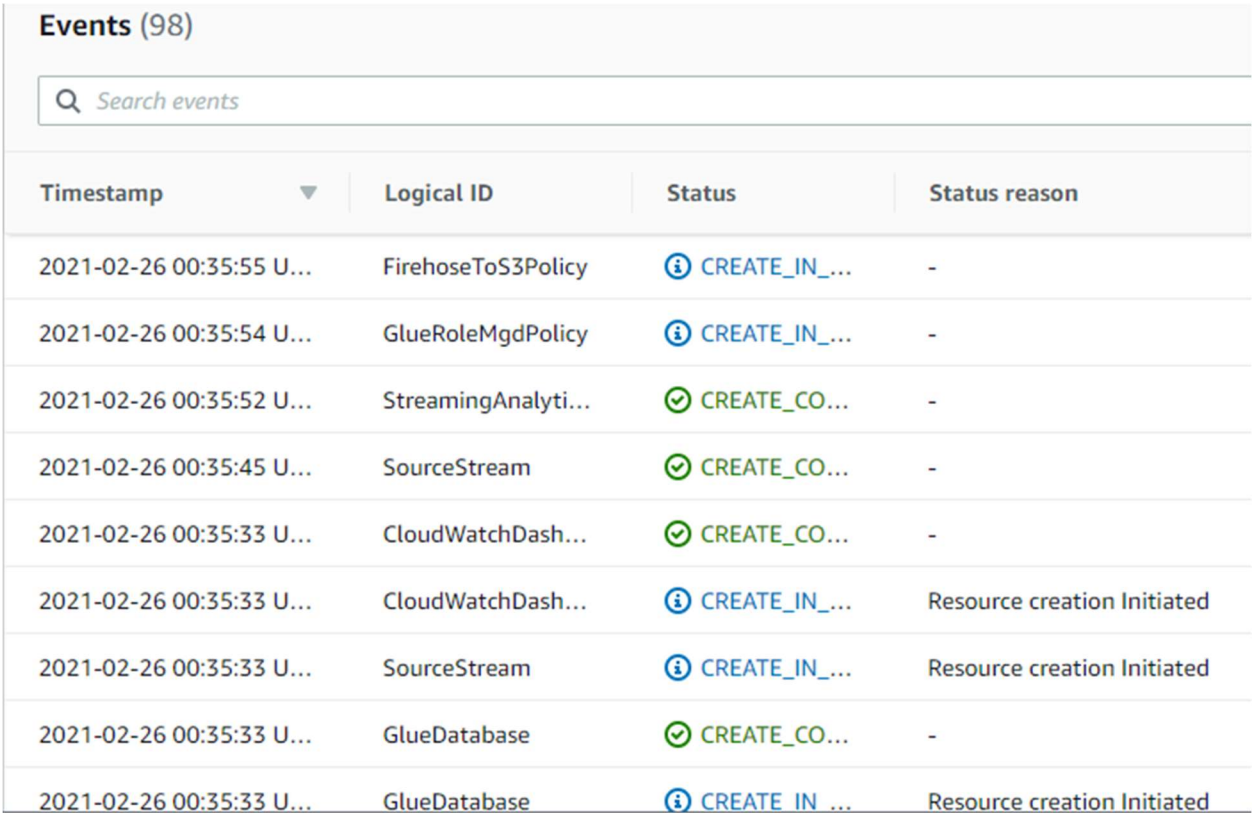
**Step 3** Now we need to add a destination Lambda function that can continuously stream this data to the Kinesis Firehose delivery system. For this, we will add the following Lambda function :




This Lambda function saves the logs in the Cloud watch which can be visualized and uses put_record function to write the data into the delivery stream.

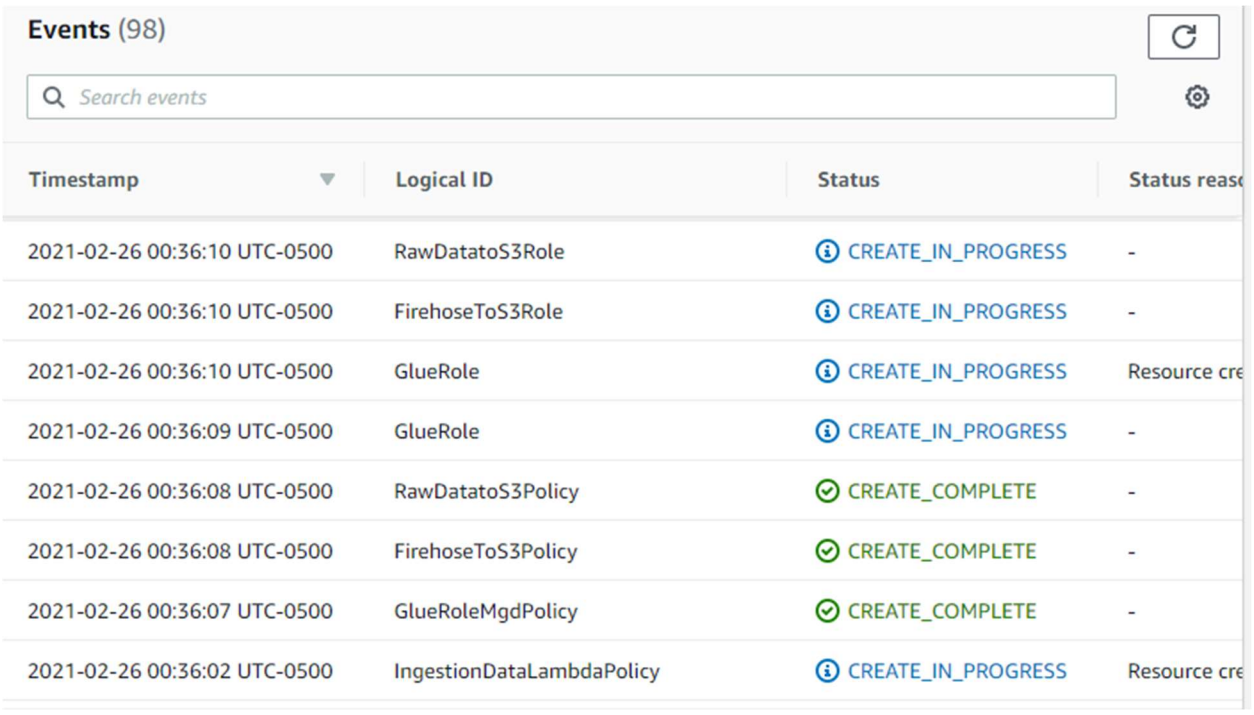
The data is then stored into the S3 bucket from the Kinesis Firehose whose event sequence can be seen as below. We also get the Glue database and the crawler created.



**Events** (98)

Search events

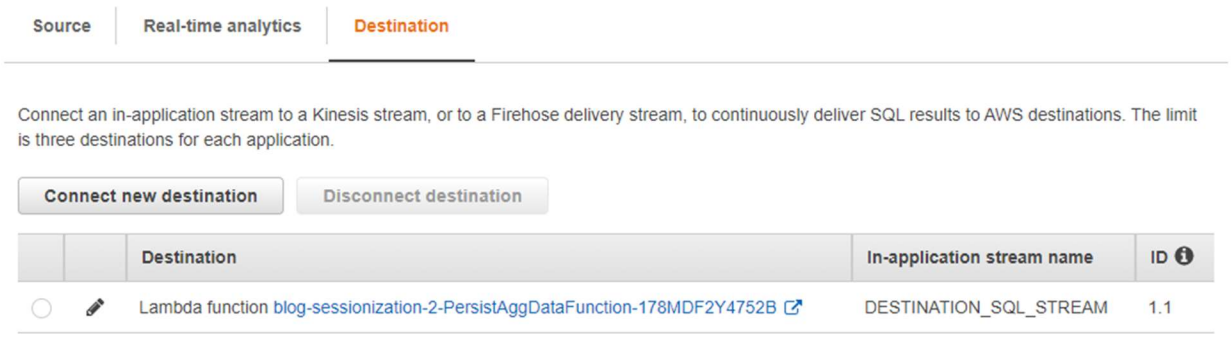
| Timestamp | Logical ID | Status | Status reason |
|---|---|---|---|
| 2021-02-26 00:35:55 U... | FirehoseToS3Policy | CREATE_IN_... | - |
| 2021-02-26 00:35:54 U... | GlueRoleMgdPolicy | CREATE_IN_... | - |
| 2021-02-26 00:35:52 U... | StreamingAnalyti... | CREATE_CO... | - |
| 2021-02-26 00:35:45 U... | SourceStream | CREATE_CO... | - |
| 2021-02-26 00:35:33 U... | CloudWatchDash... | CREATE_CO... | - |
| 2021-02-26 00:35:33 U... | CloudWatchDash... | CREATE_IN_... | Resource creation Initiated |
| 2021-02-26 00:35:33 U... | SourceStream | CREATE_IN_... | Resource creation Initiated |
| 2021-02-26 00:35:33 U... | GlueDatabase | CREATE_CO... | - |
| 2021-02-26 00:35:33 U... | GlueDatabase | CREATE_IN_... | Resource creation Initiated |



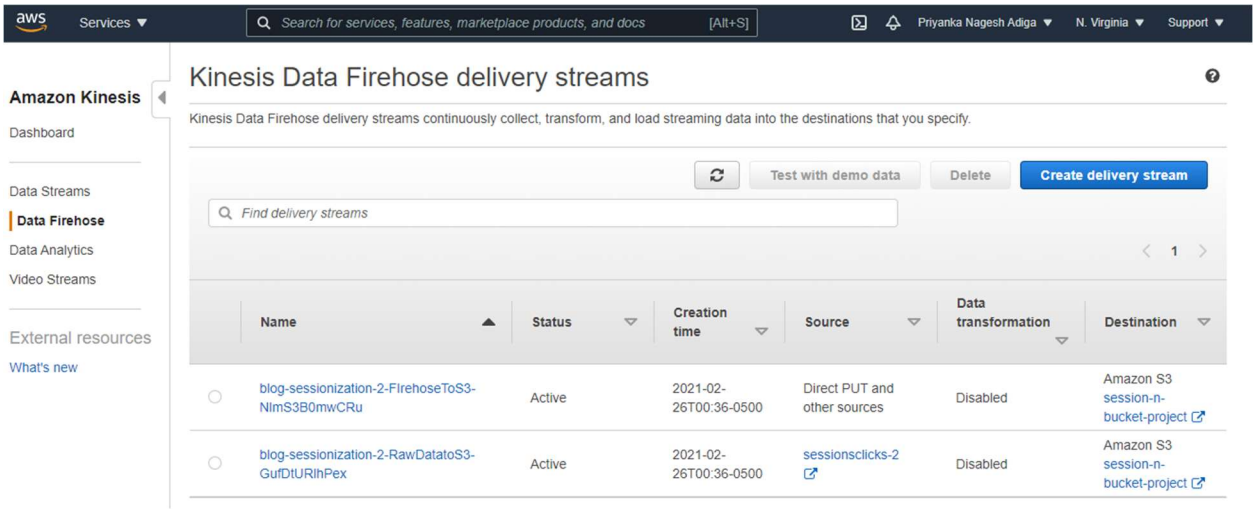
**Events** (98)

Search events

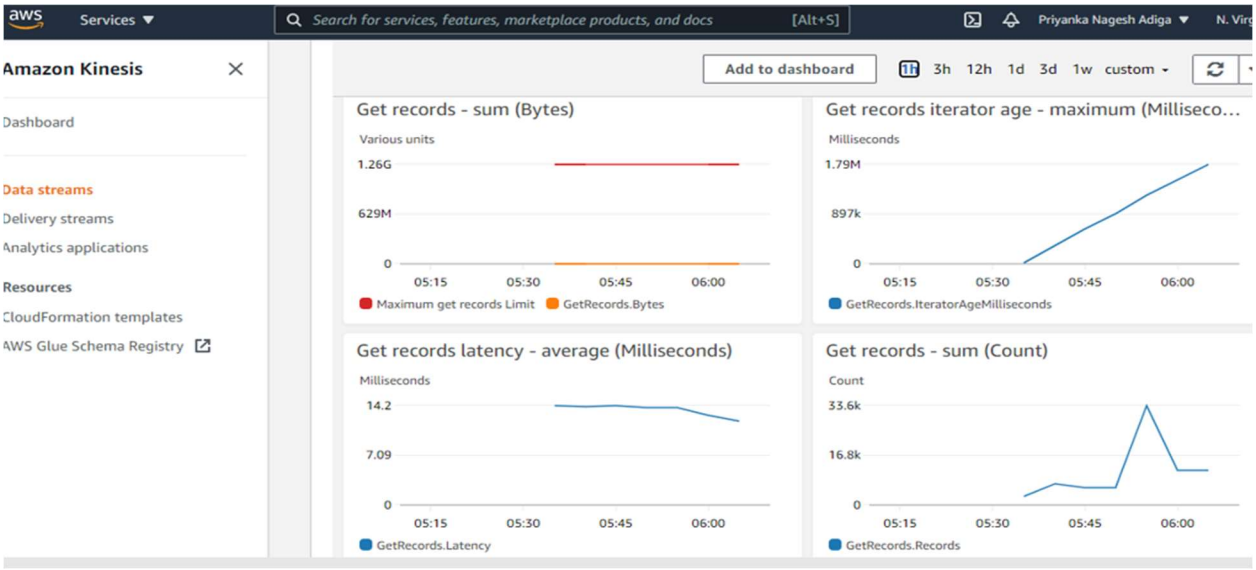
| Timestamp | Logical ID | Status | Status reas |
|---|---|---|---|
| 2021-02-26 00:36:10 UTC-0500 | RawDatatoS3Role | CREATE_IN_PROGRESS | - |
| 2021-02-26 00:36:10 UTC-0500 | FirehoseToS3Role | CREATE_IN_PROGRESS | - |
| 2021-02-26 00:36:10 UTC-0500 | GlueRole | CREATE_IN_PROGRESS | Resource cre |
| 2021-02-26 00:36:09 UTC-0500 | GlueRole | CREATE_IN_PROGRESS | - |
| 2021-02-26 00:36:08 UTC-0500 | RawDatatoS3Policy | CREATE_COMPLETE | - |
| 2021-02-26 00:36:08 UTC-0500 | FirehoseToS3Policy | CREATE_COMPLETE | - |
| 2021-02-26 00:36:07 UTC-0500 | GlueRoleMgdPolicy | CREATE_COMPLETE | - |
| 2021-02-26 00:36:02 UTC-0500 | IngestionDataLambdaPolicy | CREATE_IN_PROGRESS | Resource cre |




The raw data and the processed data are both sent to the S3 buckets using two different Kinesis Data Firehose delivery streams created as shown below.

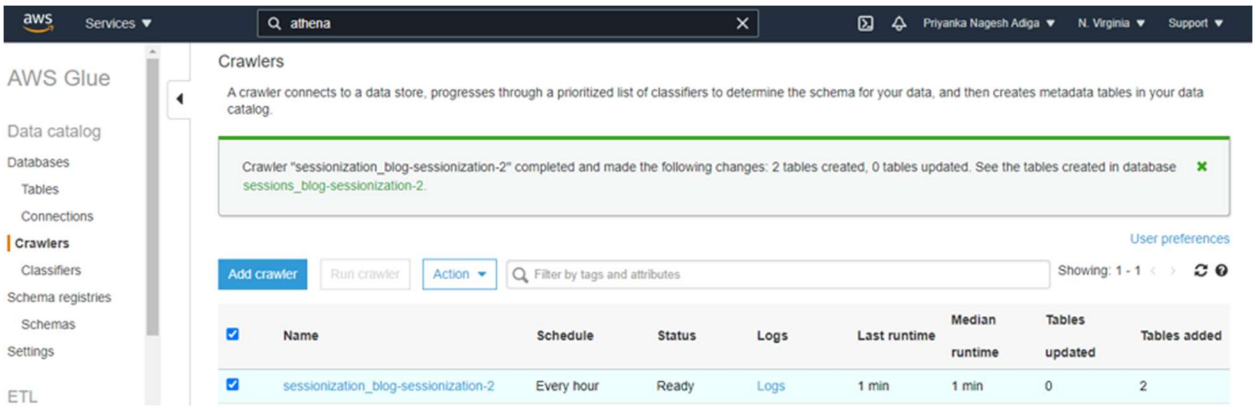



We can see CloudWatch events for the sequence of runs and also the log group visualization dashboard as shown below.

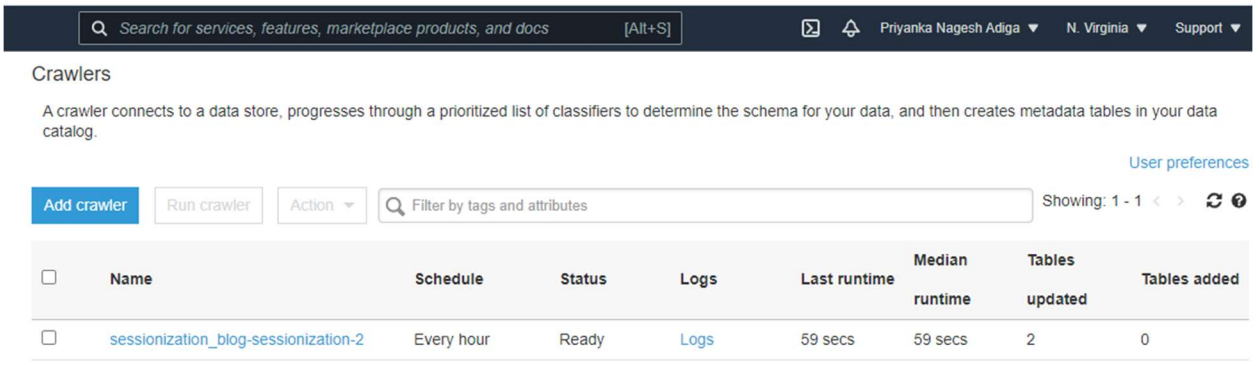


**Step 4:** Run the crawler to create Tables to connect to Athena.

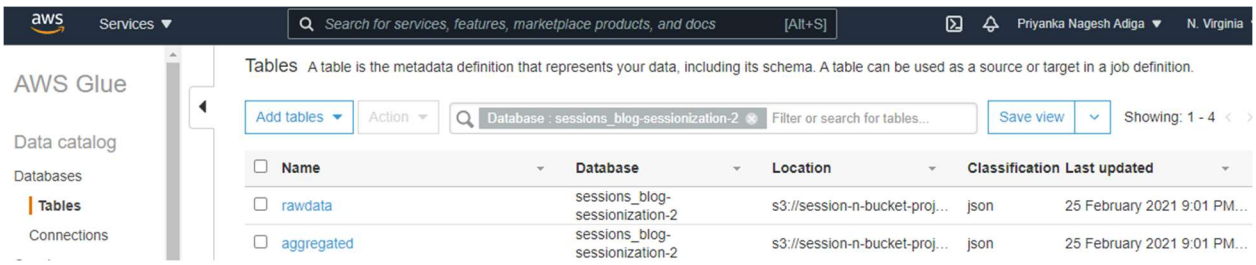
After sending the data to our S3 bucket and creating a database in the Glue Data catalog, the next step is to run the Glue crawler that automatically discovers the schema from the S3 bucket and creates the tables in the Glue database which can be connected to Amazon Athena to run queries and create views for the analysis of our clickstream data.



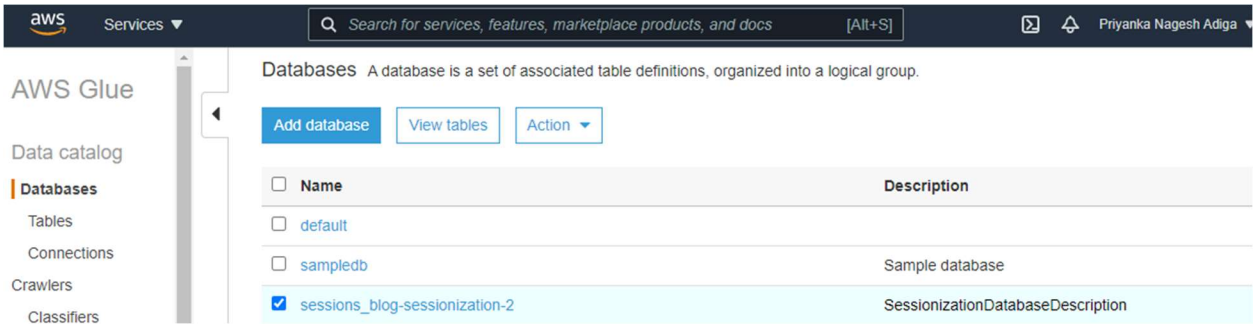

As we can see in the snippet above, after successfully running for the first time, the Crawler added 2 tables to the database and is scheduled to run every hour to discover tables.




As we can see, in the subsequent runs, the Crawler updates the raw data and aggregate data tables in the S3 bucket as shown below:



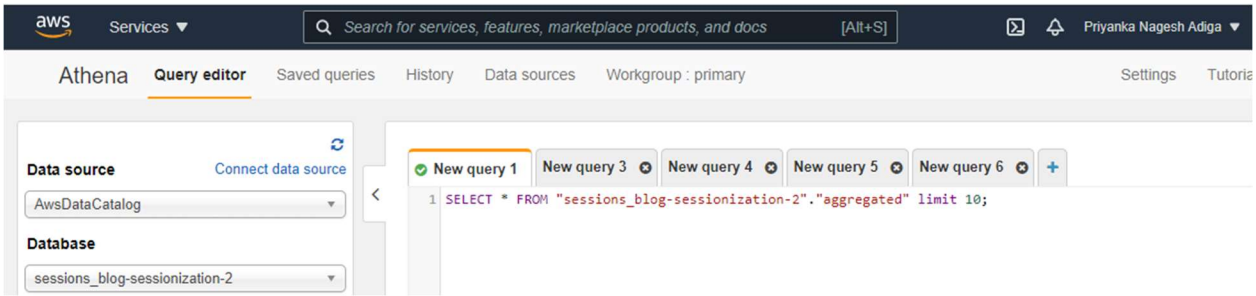
The highlighted database below contains the above two tables created by the Crawler:



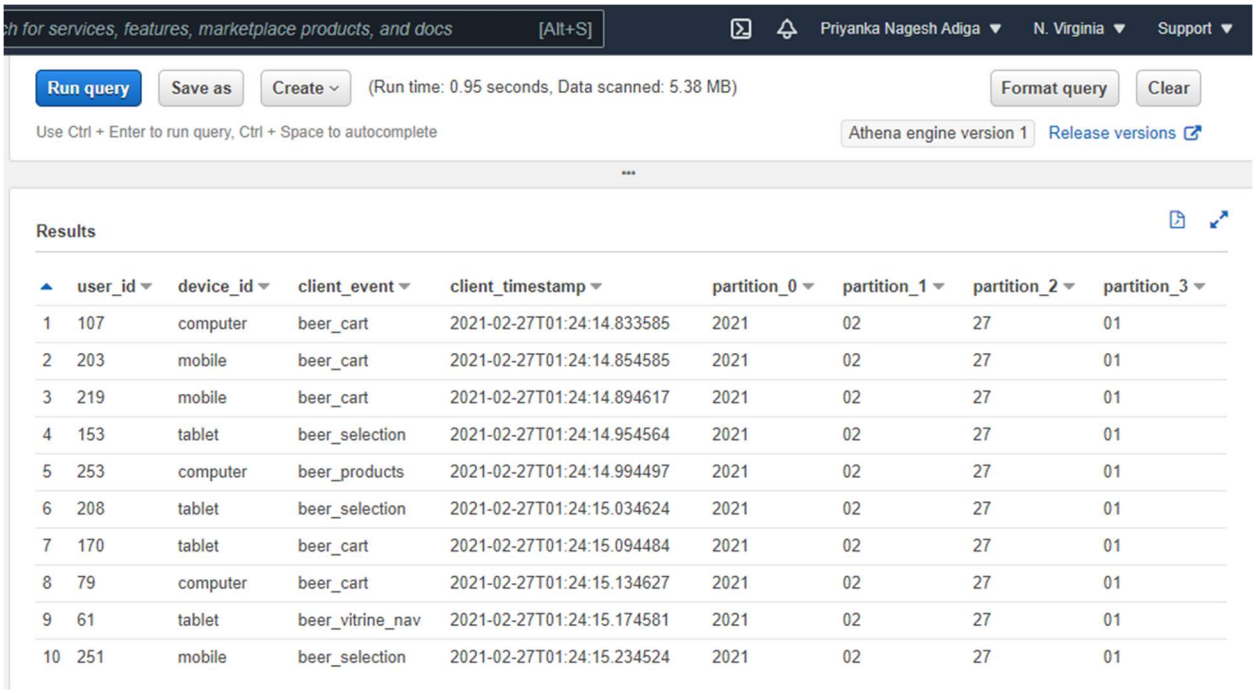

The CloudWatch dashboard can be viewed to track and monitor real time session activity with the metrics such as session duration and session events as shown below.


**Step 5:** Analyze the raw and aggregated clickstream data using Athena.

We can now explore our clickstream data and analyze it using Amazon Athena which is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.

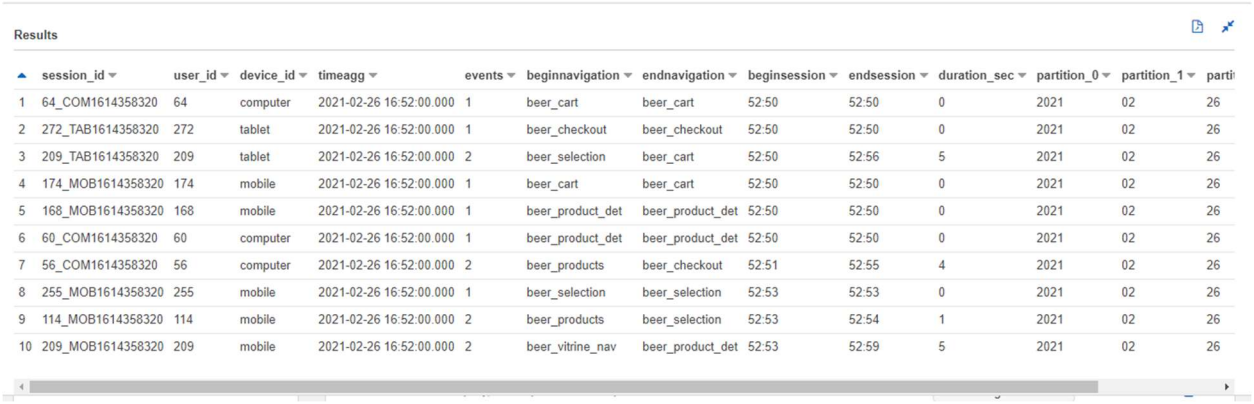


As we can see above, the query shows a green tick after successfully executing and shows the results below including the amount of data scanned as shown below.



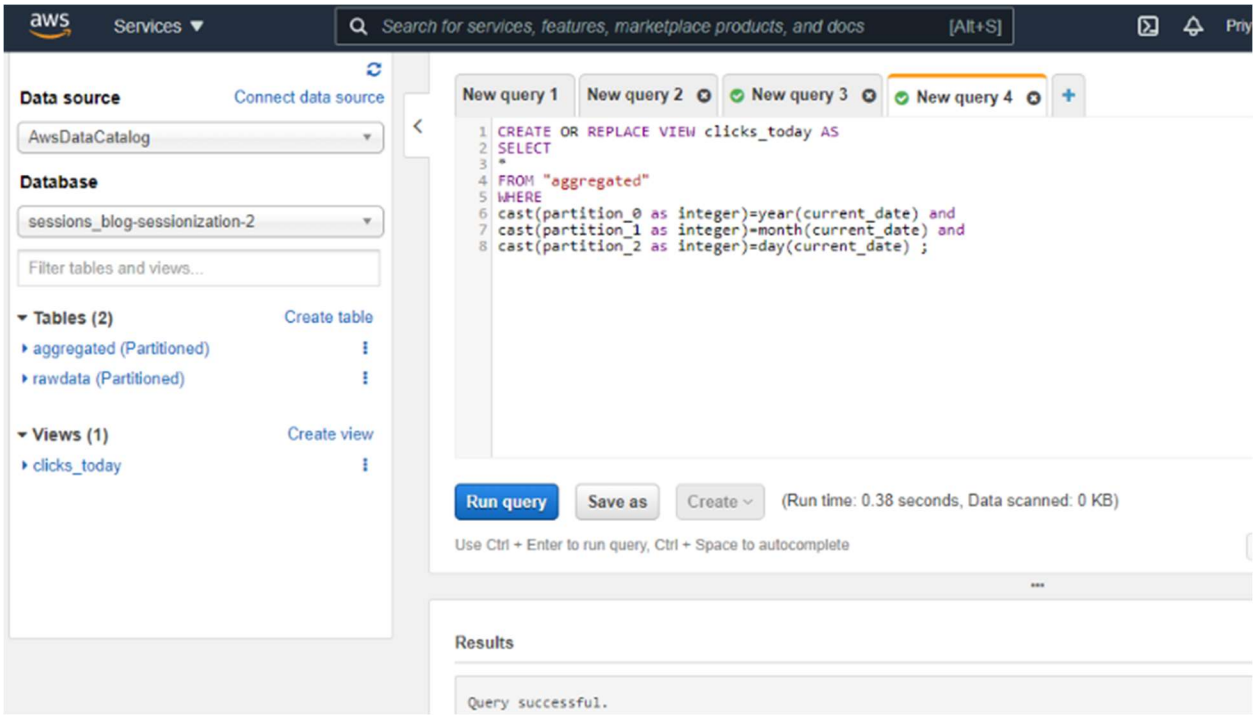

| | user_id | device_id | client_event | client_timestamp | partition_0 | partition_1 | partition_2 | partition_3 |
|---|---|---|---|---|---|---|---|---|
| 1 | 107 | computer | beer_cart | 2021-02-27T01:24:14.833585 | 2021 | 02 | 27 | 01 |
| 2 | 203 | mobile | beer_cart | 2021-02-27T01:24:14.854585 | 2021 | 02 | 27 | 01 |
| 3 | 219 | mobile | beer_cart | 2021-02-27T01:24:14.894617 | 2021 | 02 | 27 | 01 |
| 4 | 153 | tablet | beer_selection | 2021-02-27T01:24:14.954564 | 2021 | 02 | 27 | 01 |
| 5 | 253 | computer | beer_products | 2021-02-27T01:24:14.994497 | 2021 | 02 | 27 | 01 |
| 6 | 208 | tablet | beer_selection | 2021-02-27T01:24:15.034624 | 2021 | 02 | 27 | 01 |
| 7 | 170 | tablet | beer_cart | 2021-02-27T01:24:15.094484 | 2021 | 02 | 27 | 01 |
| 8 | 79 | computer | beer_cart | 2021-02-27T01:24:15.134627 | 2021 | 02 | 27 | 01 |
| 9 | 61 | tablet | beer_vitrine_nav | 2021-02-27T01:24:15.174581 | 2021 | 02 | 27 | 01 |
| 10 | 251 | mobile | beer_selection | 2021-02-27T01:24:15.234524 | 2021 | 02 | 27 | 01 |

The above table is the result from the Select * operation on the raw data and the below table is the result from Select * operation on the aggregated data.

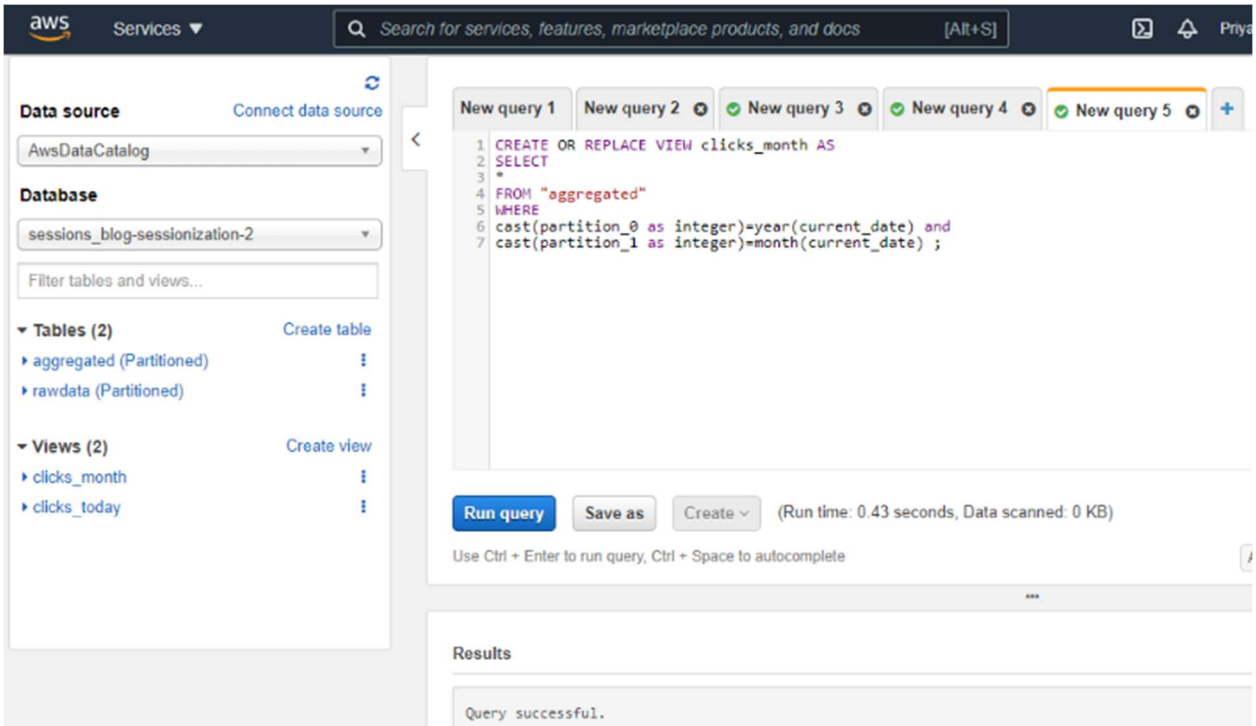



| | session_id | user_id | device_id | timeagg | events | beginnavigation | endnavigation | beginsession | endsession | duration_sec | partition_0 | partition_1 | parti |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 64_COM1614358320 | 64 | computer | 2021-02-26 16:52:00.000 | 1 | beer_cart | beer_cart | 52:50 | 52:50 | 0 | 2021 | 02 | 26 |
| 2 | 272_TAB1614358320 | 272 | tablet | 2021-02-26 16:52:00.000 | 1 | beer_checkout | beer_checkout | 52:50 | 52:50 | 0 | 2021 | 02 | 26 |
| 3 | 209_TAB1614358320 | 209 | tablet | 2021-02-26 16:52:00.000 | 2 | beer_selection | beer_cart | 52:50 | 52:56 | 5 | 2021 | 02 | 26 |
| 4 | 174_MOB1614358320 | 174 | mobile | 2021-02-26 16:52:00.000 | 1 | beer_cart | beer_cart | 52:50 | 52:50 | 0 | 2021 | 02 | 26 |
| 5 | 168_MOB1614358320 | 168 | mobile | 2021-02-26 16:52:00.000 | 1 | beer_product_det | beer_product_det | 52:50 | 52:50 | 0 | 2021 | 02 | 26 |
| 6 | 60_COM1614358320 | 60 | computer | 2021-02-26 16:52:00.000 | 1 | beer_product_det | beer_product_det | 52:50 | 52:50 | 0 | 2021 | 02 | 26 |
| 7 | 56_COM1614358320 | 56 | computer | 2021-02-26 16:52:00.000 | 2 | beer_products | beer_checkout | 52:51 | 52:55 | 4 | 2021 | 02 | 26 |
| 8 | 255_MOB1614358320 | 255 | mobile | 2021-02-26 16:52:00.000 | 1 | beer_selection | beer_selection | 52:53 | 52:53 | 0 | 2021 | 02 | 26 |
| 9 | 114_MOB1614358320 | 114 | mobile | 2021-02-26 16:52:00.000 | 2 | beer_products | beer_selection | 52:53 | 52:54 | 1 | 2021 | 02 | 26 |
| 10 | 209_MOB1614358320 | 209 | mobile | 2021-02-26 16:52:00.000 | 2 | beer_vitrine_nav | beer_product_det | 52:53 | 52:59 | 5 | 2021 | 02 | 26 |

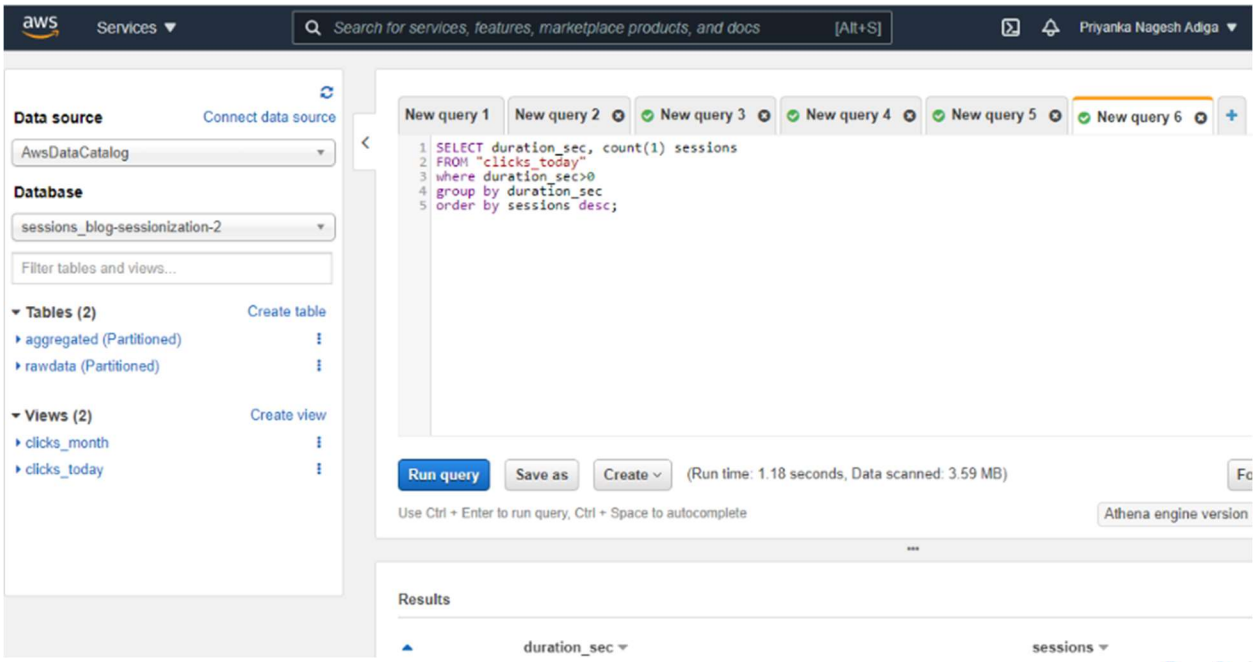
Now, we can create views to analyze our data better, such as creating a view from the aggregated table that can pull out all the clicks for today as shown below:




As we can see, the results say Query successful and we get a view created on the left side of the panel, similarly, we can create a view to pull out all the clicks of the month as shown below.



Now that we have our views created as per clicks segregation, we can run all types of analysis on them based on the business requirement. For instance, if the manager wants to find out how many sessions occurred today and what were there durations, order by the highest number of sessions first, we can execute the following query to give us the results instantaneously.




Results

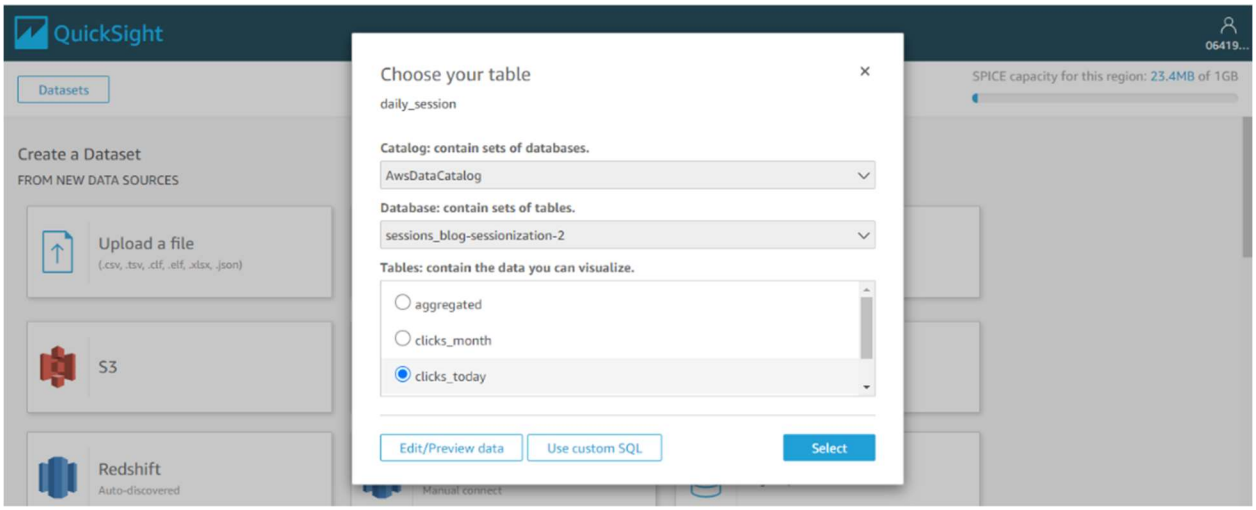
| | duration_sec | sessions |
|---|---|---|
| 1 | 33 | 486 |
| 2 | 28 | 423 |
| 3 | 29 | 345 |
| 4 | 25 | 343 |
| 5 | 19 | 324 |
| 6 | 36 | 319 |
| 7 | 32 | 315 |
| 8 | 21 | 301 |
| 9 | 30 | 293 |
| 10 | 22 | 292 |
| 11 | 16 | 280 |
| 12 | 40 | 268 |
| 13 | 37 | 267 |
| 14 | 35 | 265 |
| 15 | 44 | 255 |

The results keep getting updated everytime the Crawler runs and updates the table with new clickstream data. This period of updation can be set as per business need as running Crawlers incur costs. A schedule of every hour would mean that the results would get updated every hour, but it can be also done on ad hoc requirement by manually running the Crawler before schedule.

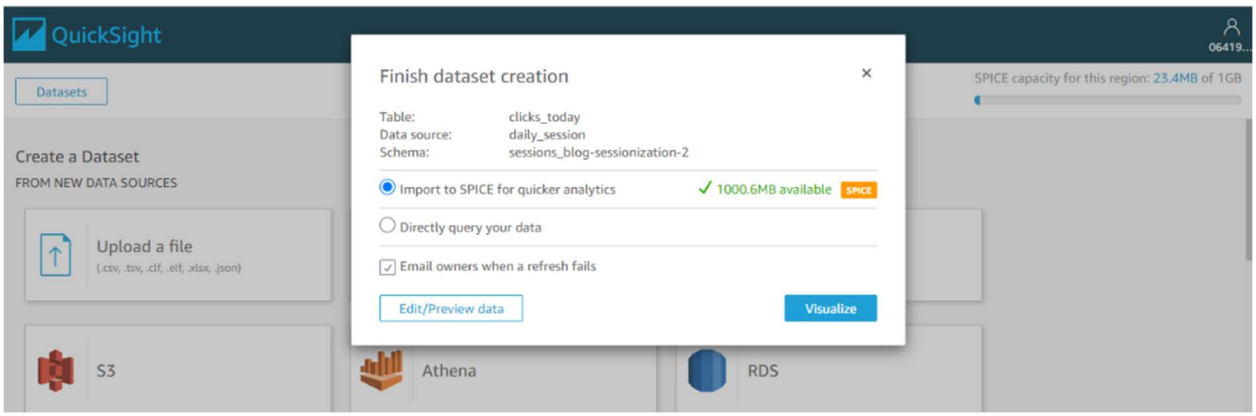
**Step 6:** Visualize the data using QuickSight and create a dashboard

Managers like to have their results visualized and grouped together in a dashboard. In order to achieve this, we can connect our Amazon Athena results to the Amazon QuickSight service where we can create our reports based on the business requirement.

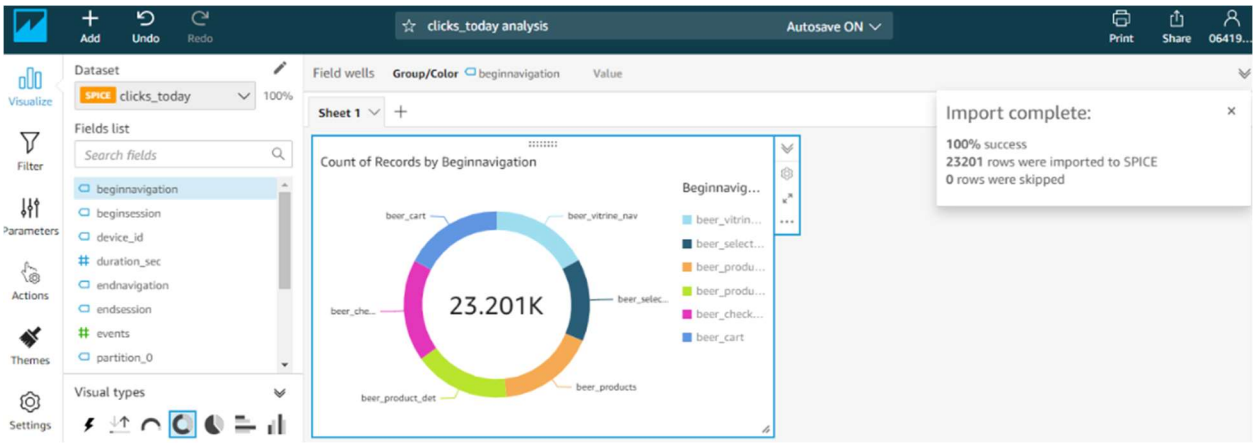
Continuing with the scenarios of clicks for today, we can chose this view from the tables section while creating a new dataset called daily session to visualize in QuickSight as shown below:



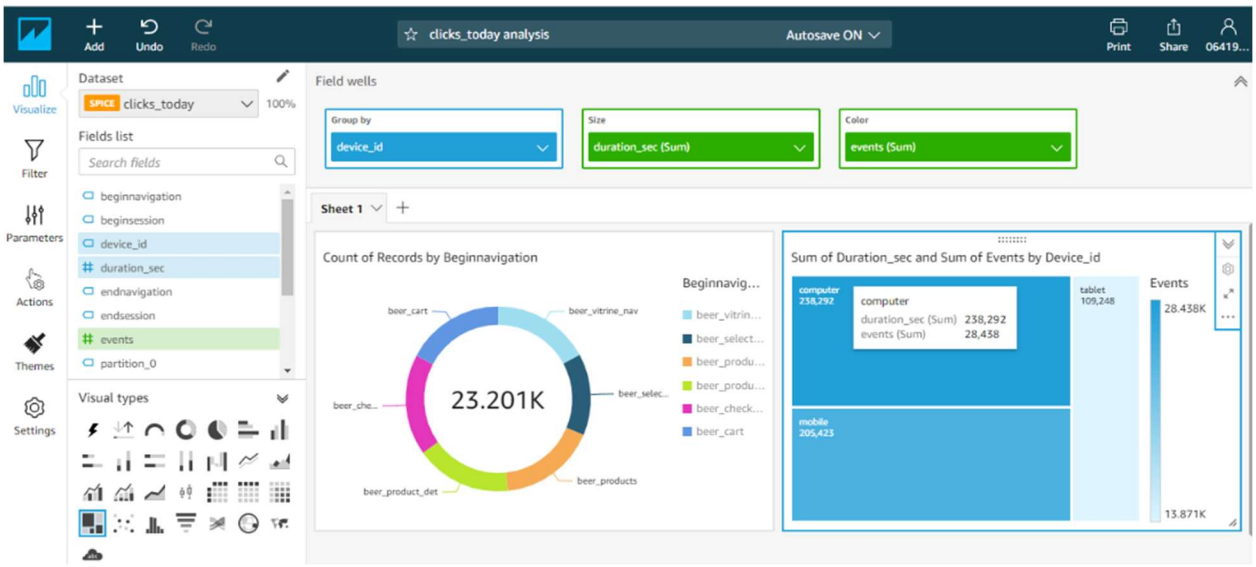

After choosing our table for the dataset, we can import it to SPICE for quicker analytics as shown in the snippet below:



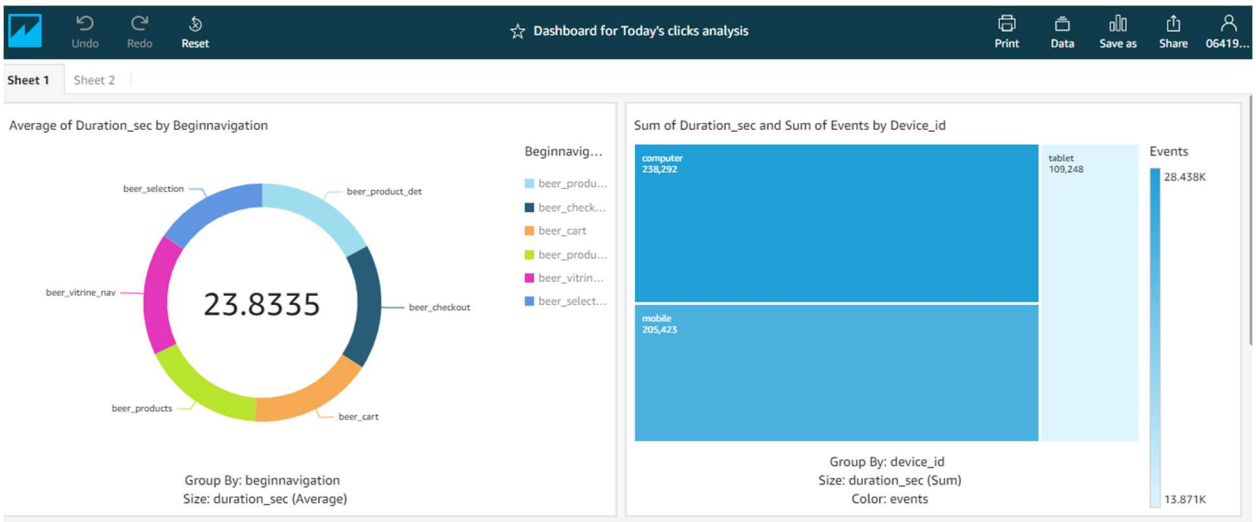

Once our dataset is ready, we can start creating our visualizations to answer the business questions that our managers want answered. QuickSight makes it very convenient to quickly create dashboard by just selecting the fields from the dataset and the visual type, which is the case of a donut chart for showing the count of begin navigation below.

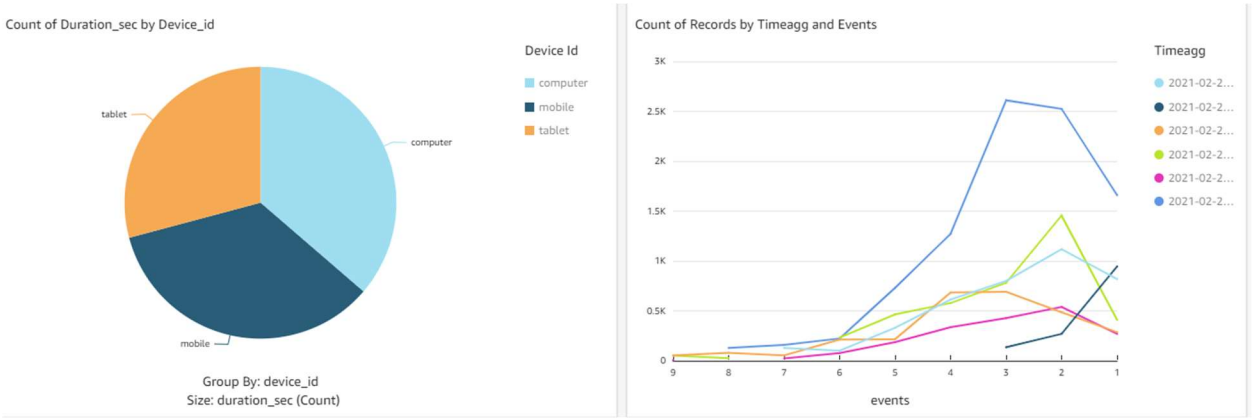


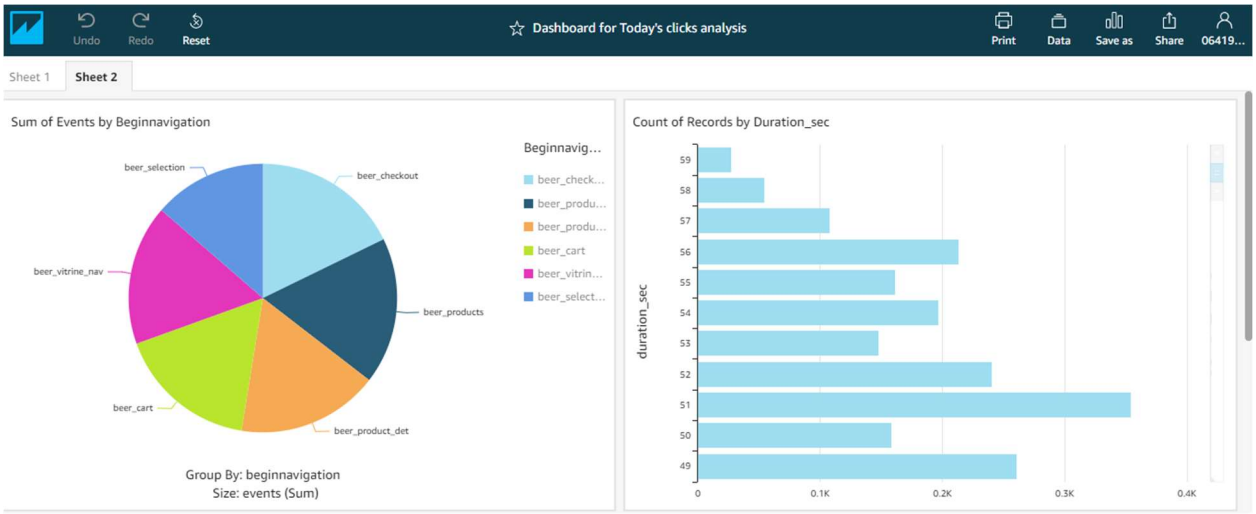

Similarly, we can add visuals on the same sheet or new sheets based on different requirements and to show different results. For example, if the manager wants to see which device is being used by the users for the maximum amount of time and what is the comparison of duration usage between the devices, we can add a tree map that takes the sum of durations in seconds, and groups them by each devices and color codes them in a scale of number of events as shown in the snippet below:

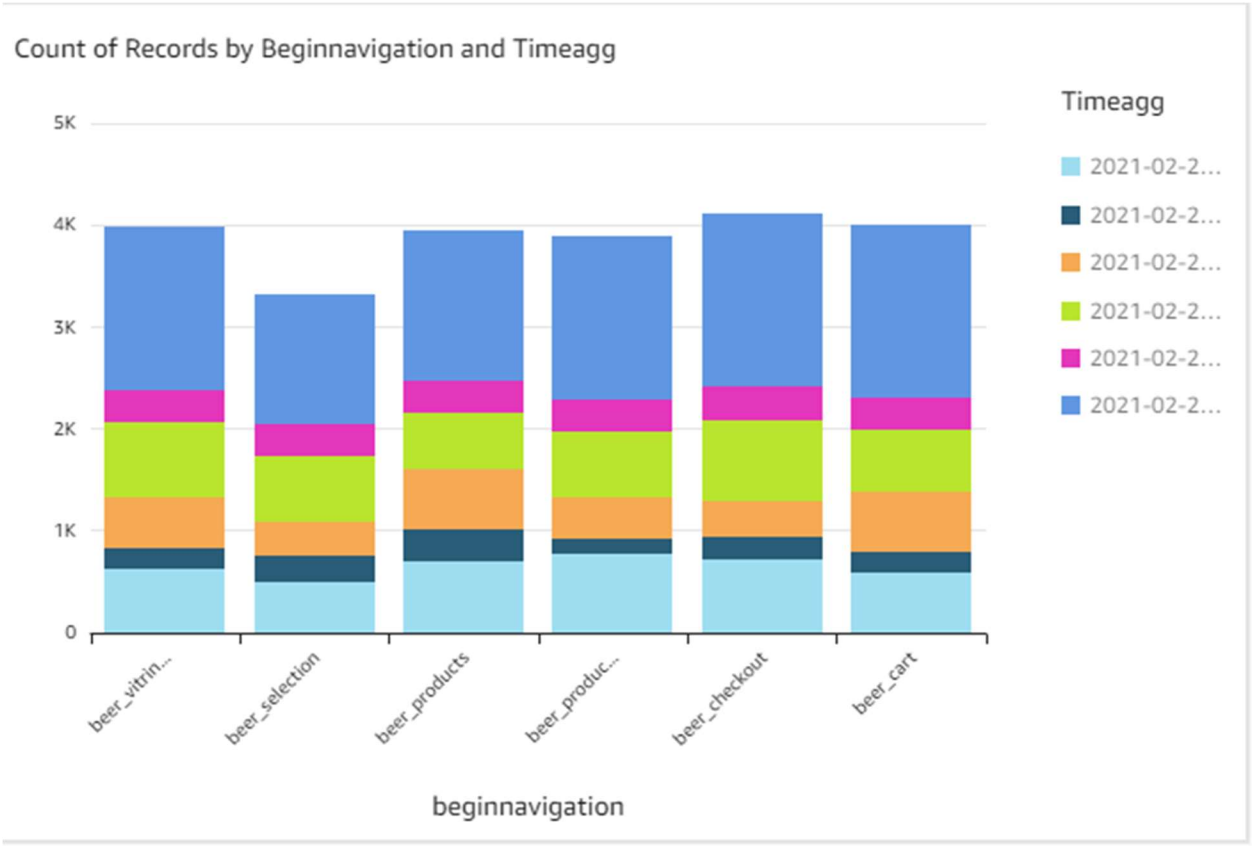



After publishing the analysis, our dashboard looks as below:






This interactive dashboard is now ready to be presentated to the client that will give analysis results based on real time clickstream data.

## Conclusion

In this project, we successfully implemented a clickstream analysis solution from an amalgamation of different AWS services. As a proof of concept, we generated a dummy clickstream streaming data using Kinesis that simulated real world clickstream events and processed the stream real time to identify sessions based on devices and lag time. We then fed this data to Athena using AWS Glue to analyze the data and gain business insights. Finally, these insights were visualized by QuickSight and delivered using a dashboard to the client to enable a real time data driven business decision ability to the company to maximize turnover.

## References

[1].Hugo Rozestraten, (February 07, 2019), Create real-time clickstream sessions and run analytics with Amazon Kinesis Data Analytics, AWS Glue, and Amazon Athena. Retrieved from : https://aws.amazon.com/blogs/big-data/create-real-time-clickstream-sessions-and-run-analytics-with-amazon-kinesis-data-analytics-aws-glue-and-amazon-athena/

[2].Cambridge Technology, AWS, (n.d.), Clickstream Analytics on AWS. Retrieved from : https://aws.amazon.com/quickstart/architecture/clickstream-analytics/