**CSP 554 - PROJECT REPORT**

****

**BUILDING A DATA PIPELINE USING DYNAMODB AND AWS KINESIS**

Rajesh Kumar Bandaru (A20446254)

Mohammed Jawhar (A20449684)

Sai Charan Akkena (A20456762)

**ABSTRACT**

A Data Pipeline is a series of data processing steps. It can be a piece of software that eliminates many manual steps and enables automated flow of data. It has three important components:

1.  A Source

2.  Processing step/s

3.  A Sink

The data pipeline in this project enables the flow of data from a database to an analytics platform. As the volume of data being generated every day is increasing exponentially the term “big data” implying huge data came into existence. This “big data” can be used to make predictive analytics, real-time reporting, and alerting.

The architecture of a data pipeline requires many considerations. Does the pipeline need to handle streaming data or batch data? Does it need to satisfy specified conditions? What type of processing needs to be done? Where does the data need to go, where is the data coming from?

There are different types of architectures for data pipelines. They are:

**1.** **Batch based pipeline**: This kind of architecture is used when there are a large number of data points that have to be pushed into a warehouse. It is most useful when we want to move large volumes of data at regular interval.

**2. Streaming data pipeline:** This kind of architecture is used when there is a need to process the data point as it is generated.

**3.  Lambda based pipeline**: This is the combination of both batch and streaming data pipeline. This architecture is popular in big data environments where we can process both real-time data and historical data.

In this project we used Lambda based architecture to process the data of the Permanent magnet synchronous motor (PMSM) of Power Electronics and Electrical Devices. The readings of the data are taken batch wise, and we need a real time analysis to give alerts if anything goes wrong with the machine.

**PROJECT DETAILS**

1.  **Project Topic:** Buildinga Data Pipeline using DynamoDB and AWS Kinesis.

2.  **Dataset**: Recordings from a Permanent Magnet Synchronous Motor (PMSM)

3.  **About the Dataset**:

The data set comprises several sensor data collected from a permanent magnet synchronous motor (PMSM) deployed on a test bench. The dataset was taken from Kaggle offered by Paderborn university. The readings are as follows:

1.  Ambient temperature.

2.  Coolant temperature.

3.  Motor speed.

4.  Current components.

5.  Voltage components.

6.  Stator yoke temperature.

7.  Stator tooth temperature.

8.  Stator winding temperature.

9.  Permanent magnet surface temperature.

These readings are measured by appropriate sensors attached to the motor. There are more than 990,000 readings taken over 72 sessions. The data is available as a csv file.

4.  **Project Goals**:

* Create a DynamoDB table.
* Ingest data into the DynamoDB Table.
* An automated Data pipeline is created using AWS Cloud Formation.
* Enable Kinesis Data Streaming for DynamoDB.
* Transform the data as needed using lambda function and kinesis data firehose.
* Analyzing the data using Amazon Athena if needed.  A Quick Sight dashboard for analysis.

5.  **Technologies**: Amazon DynamoDB, Amazon Kinesis Data Streams, Amazon Kinesis Data Firehose, Amazon Athena, Amazon Quick Sight.

**LITERATURE REVIEW**

* **Data Pipeline**

A data pipeline is used to move data from one end to another end and transform data in between for analysis at the other end. It consists of a source, a data transformation step, and a sink. In this project, the pipeline allows users to move data from a DynamoDB table to Athena for querying and Quick Sight for visual analysis. It is also called as a series of data processing steps. Since streaming should be considered while building a data pipeline, we make use of AWS Kinesis services to move data from source to sink. We assume our data is generated from the cloud and stored in the DynamoDB table (source).

* **DynamoDB**

The data for this project is stored in the DynamoDB table. DynamoDB is amazon’s NoSQL database service and delivers single - digit millisecond performance at any scale. It is a fully managed, multi-region, multi-active, durable database with built-in security backup and restore and in-memory caching for internet scale applications. DynamoDB is serverless with no servers to provision, patch or manage and no software to install, maintain or operate. It automatically scales tables up and down to adjust for capacity and maintain performance.  It supports ACID transactions for building business-critical applications at scale.

* **AWS Kinesis**

For collecting the data from DynamoDB and further processing AWS Kinesis is used. It is used to analyze real time data so that timely insights can be taken and react fast as per the situation. The application logs of our data set can be ingested real-time for analysis in Quick Sight. Amazon kinesis can ingest, buffer and process streaming data in real-time so that insights can be derived in seconds or minutes instead of hours or days. It can handle data from hundreds of thousands of sources with very low latencies.

* **Amazon Athena**

It is a query service offered by amazon to query data in S3 buckets interactively. The querying can be done in Standard SQL. It is a serverless service. The data streamed from DynamoDB is queried for necessary data needed for analysis in Quick Sight. Athena uses presto with ANSI SQL support and works with various standard data formats. Athena is fast as it executes queries in parallel, so more results will come back in less time.

* **Quick Sight**

The data extracted from Athena is just the data in another format. It should be converted into something visually soothing so that we can get insights from them. Quick Sight does the exact same. It is a scalable, serverless service built for the cloud. It can easily create interactive dashboards which are machine learning powered.

**OBJECTIVE**

The objective of the project is to create a data pipeline to make analysis on the PMSM dataset. It is a way to monitor the incoming data. To delete or alter the schema as the company’s requirements and to take the necessary information from the dataset for performing analysis. It helps in giving alerts if readings are out of range or if anything is wrong with the machine based on the sensor data collected. This helps Power Electronics and Electrical Devices to take measures according to the situation.

**ARCHITECTURE**

**Diagram

Description automatically generated**

**OVERVIEW OF FLOW**

The data of PMSM machine is uploaded into an input source batch-wise. A lambda function is used as a trigger to upload data into DynamoDB to store historical data. Once the data is uploaded into DynamoDB, the DynamoDB stream is activated to stream data through Amazon kinesis. The data from Kinesis is delivered by a delivery stream called Kinesis Data Firehose to an S3 Bucket.

A lambda function is used as a trigger to deliver the data to S3 whenever new data is added to DynamoDB. Data from S3 bucket is crawled using a Crawler for it to be available for querying in Athena. Data crawled from S3 is stored in Glue Data Catalog. It is accessed by Athena by mentioning the database name in Athena. The data queried is then visualized in Quick Sight.

**METHODOLOGY**

The input source in our case is a S3 bucket. In real life, data is sent from IOT devices which measures the appropriate readings of the PMSM machine.

Graphical user interface, text, application

Description automatically generated

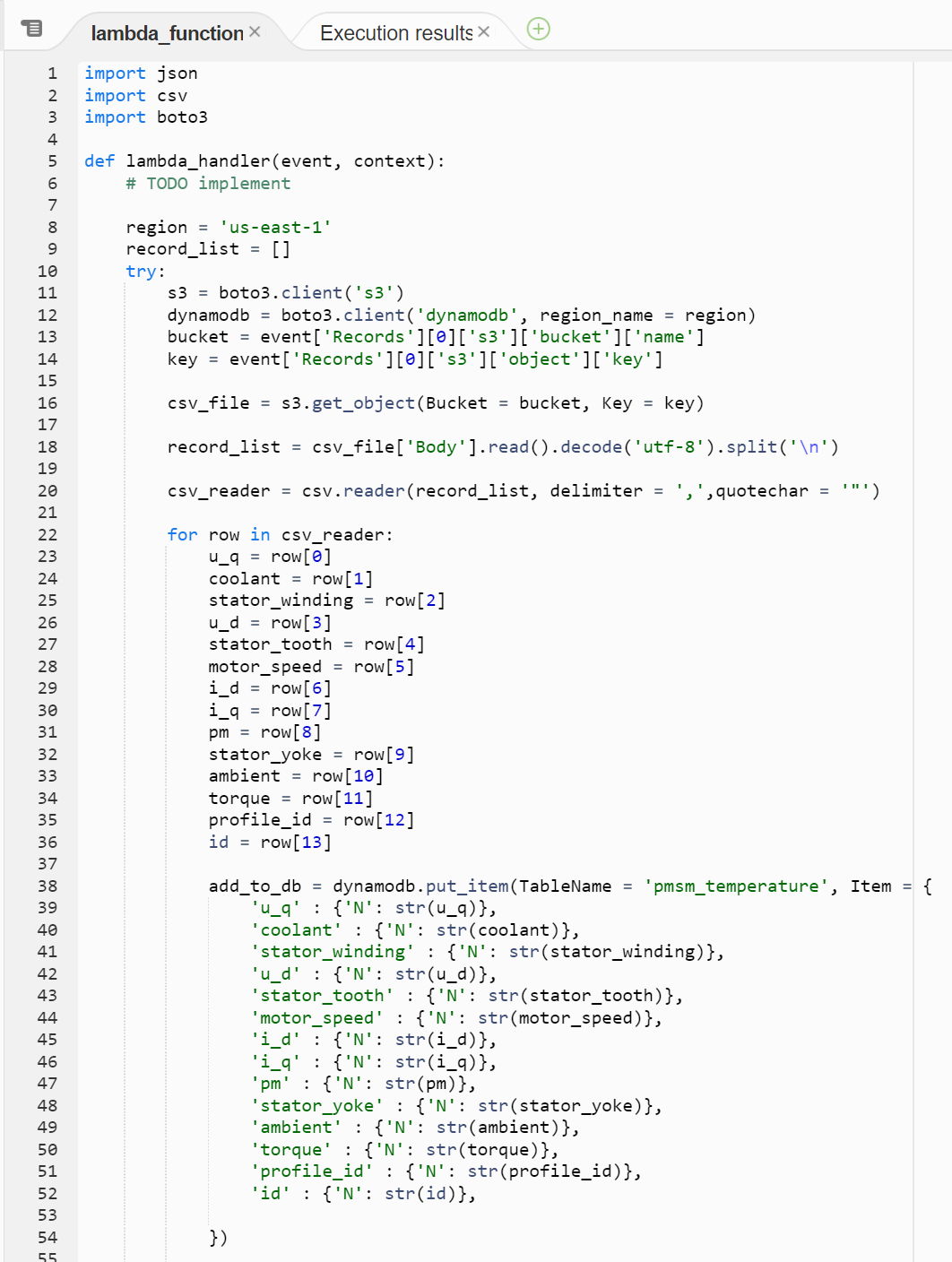
The above marked part in red is the data coming from the sensors of the machine. This is the first batch of data that is generated. The data generated batch wise is sent to the bucket regularly. Once the data is uploaded into a s3 bucket(in our case it’s a .csv file), a trigger is used to invoke the lambda function and inserts the data into DynamoDB. A trigger is a Lambda resource or a resource in another service that we configure to invoke our function in response to lifecycle events, external requests, or on a schedule. The function can have multiple triggers. Each trigger acts as a client invoking our function independently.

Graphical user interface, application

Description automatically generated

**The Trigger**

In the Trigger Configuration the s3 bucket serves as the event source and the Event Type in the Trigger is ‘PUT’ with the suffix given as “.csv” which means that whenever as “.csv” file is uploaded into the s3 bucket, the lambda function named “**bigdataprojectcsvtodynamodb’** gets triggered, and the data is inserted into the DynamoDB table. The data is used for historical analysis as well as real time analysis. Making the data use for both purposes makes it fall under the category of lambda architecture of Data Pipeline.

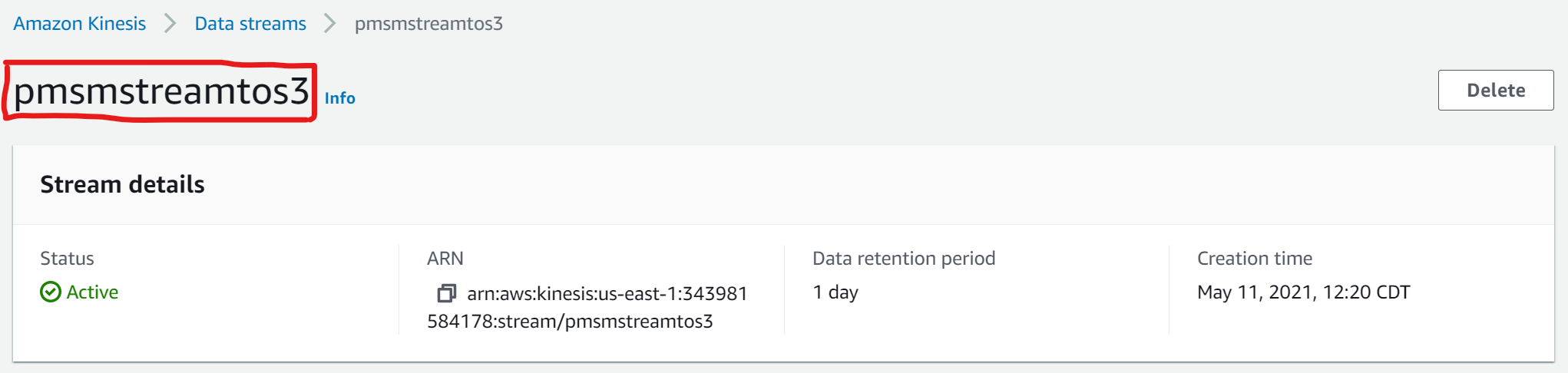


Graphical user interface

Description automatically generated

The DynamoDB table created is called **‘pmsm\_temperature’** indicated by the red box above and the primary key / partition key is the ID of the reading indicated by orange box.

A kinesis Data stream is needed to stream the data from DynamoDB whenever a new batch of data is added to it, so a kinesis data stream ‘**pmsmstreamtos3’** is created. The status of the stream is ‘active’ meaning it is ready to stream the data.



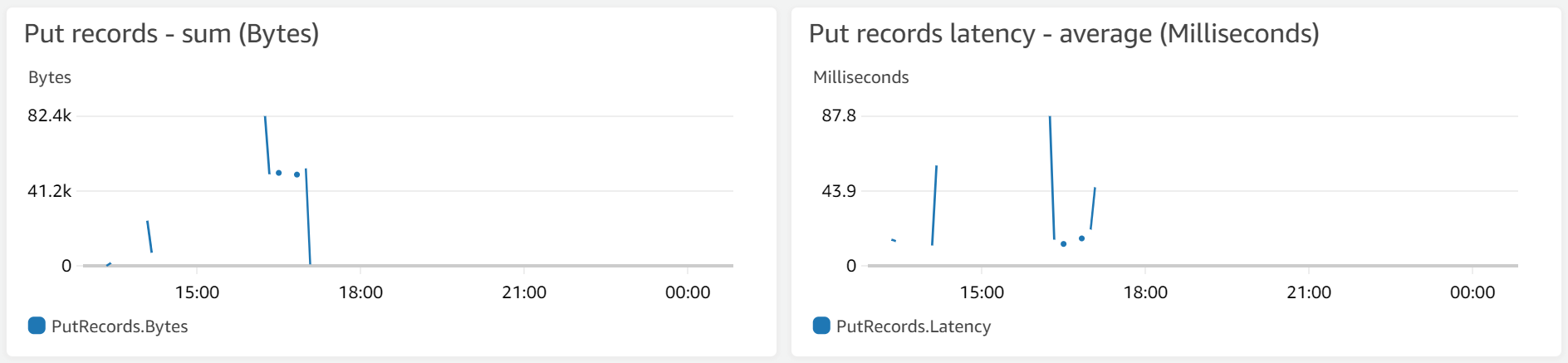
To stream the data in DynamoDB, Stream should be enabled. This is done as follows:

Graphical user interface, text, application, email

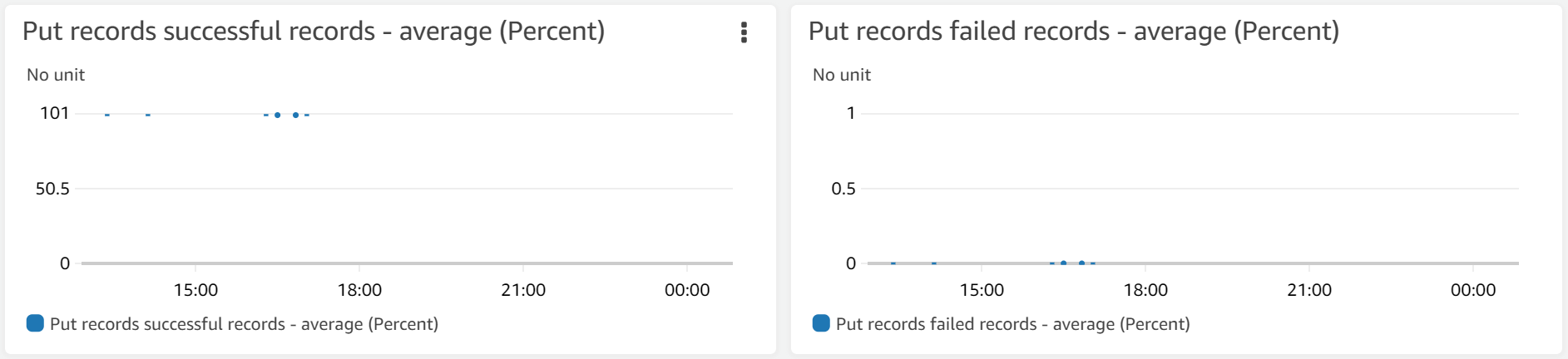
Description automatically generated

The red box above indicates that streaming is enabled as ‘Stream enabled’ is set to ‘Yes’ through ‘pmsmstreamtos3’ which was created in the previous step. In the orange box, ‘View type’ is set to ‘New image’ which means new data which is added to DynamoDB is only sent through streaming for real-time analysis. The old and new data is stored in DynamoDB table for historical analysis later.

Now, that the stream is enabled data is sent from DynamoDB through Kinesis Stream. This can be seen in the ‘**Monitoring’** section of Kinesis Data Stream.

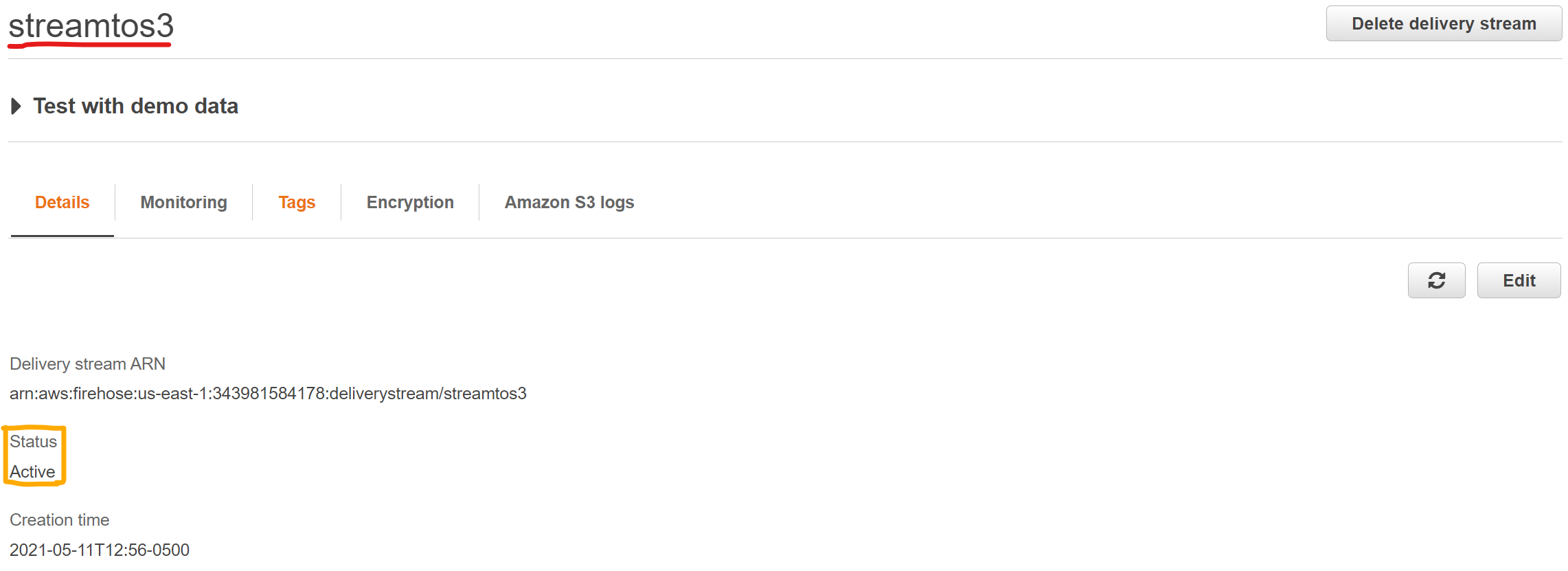


The above graphs show around 82.4k bytes of data was streamed and there is a latency of 87.8 ms.

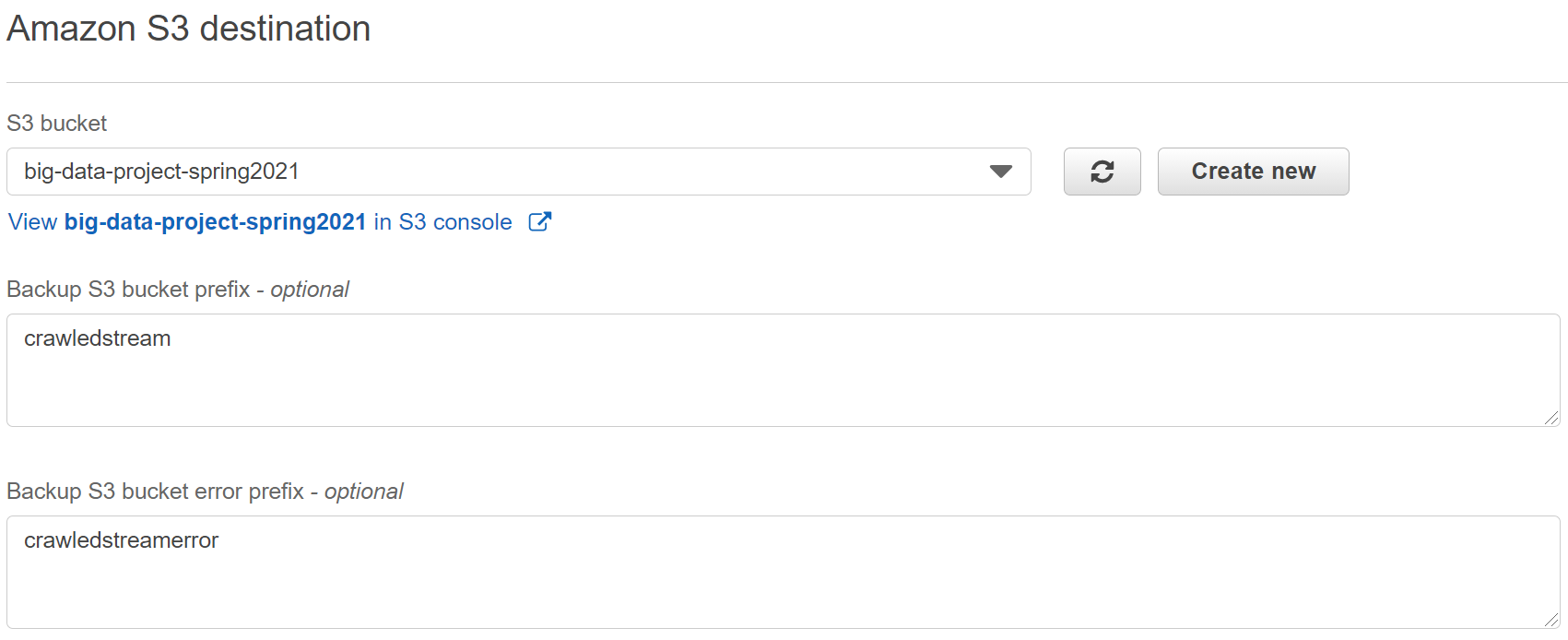


The above graphs show around 101 units of data were successfully streamed and almost 0 units of data were failed.

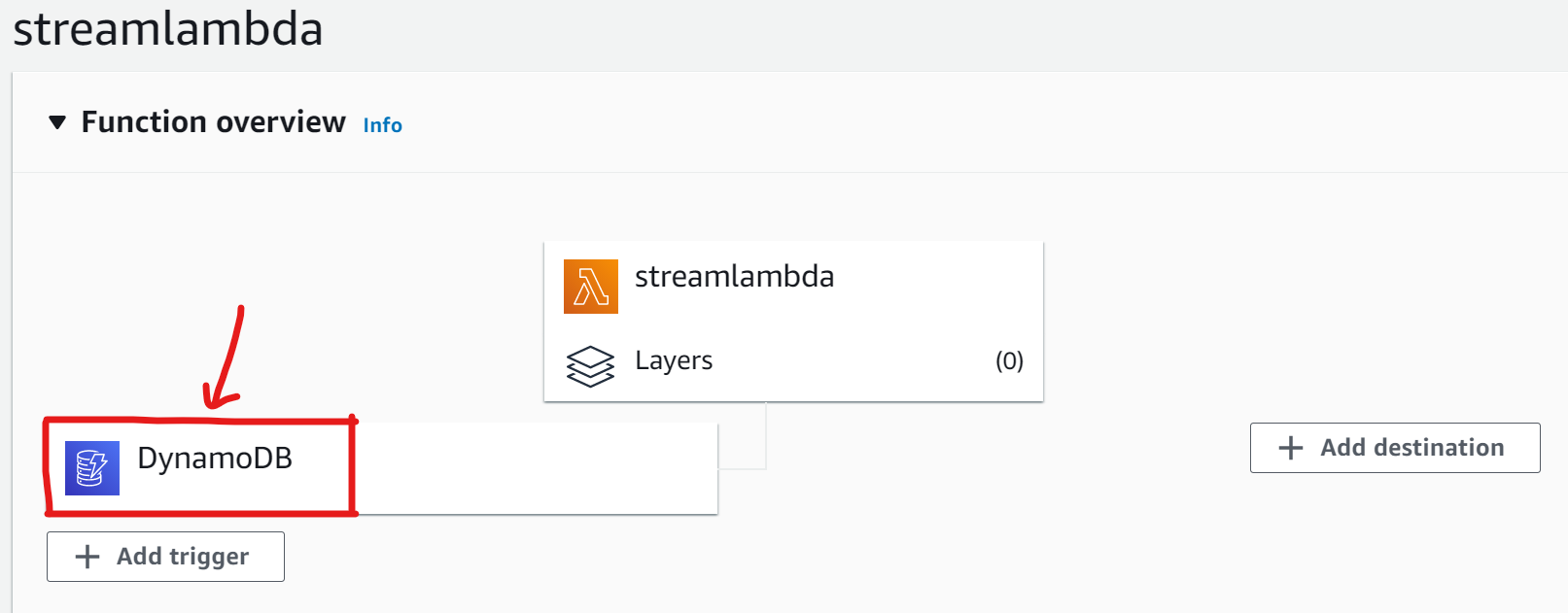
Now that the data is streaming, it should be delivered to a S3 bucket for querying in Athena. Now, Amazon Data Firehose does the job. For this a delivery stream should be created.



A delivery stream named ‘**streamtos3’** is created and the ‘Status’ of this stream is ‘Active’ (indicated by orange box). The delivery stream delivers the data to S3 bucket as mentioned before. The destination of this stream is **‘big-data-project-spring2021’**.



It is not always that the delivery stream sends data to an S3 bucket. Whenever new data is added to DynamoDB the delivery stream should be activated. This can be done through another lambda function.



**The Trigger**

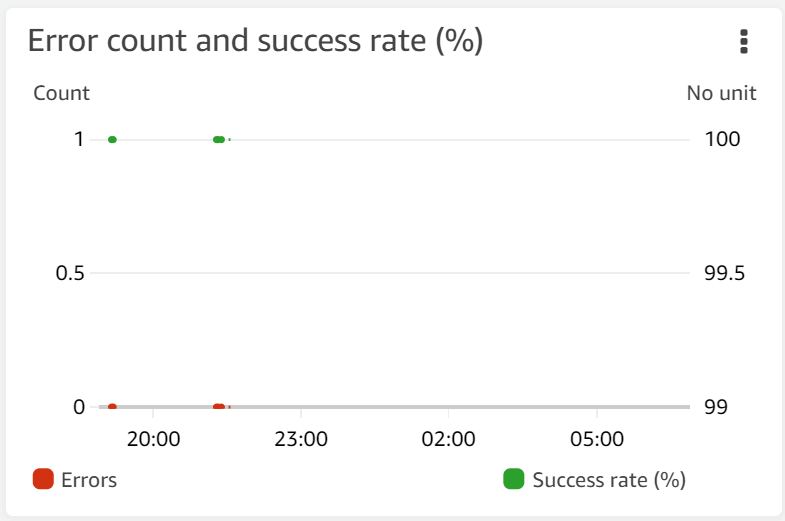
The ‘**pmsm\_temperature’** table of DynamoDB is given as a trigger to the lambda function by setting the **Batch window** to 5 meaning data will start streaming when 5 or more units of data is loaded into the table.

Graphical user interface, text, application

Description automatically generated

The lambda function ‘**streamlambda’** is triggered when more than 5 units of data is loaded into the DynamoDB table. Streaming happens only when data is inserted or modified in the table (indicated by red box above) through the delivery stream ‘**streamtos3’** (indicated by orange box) which we created in the previous step.

The success rate and error count of streaming can be seen from the ‘Metrics’ section of lambda function.

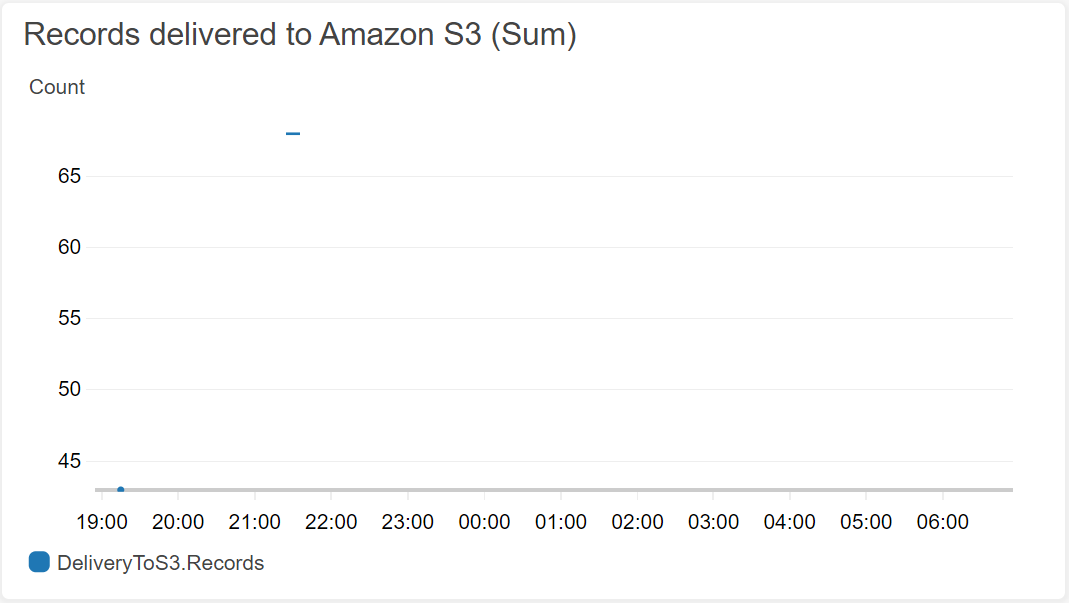


By the above graph, we can say that all the new data in **‘pmsm\_temperature’** table is delivered successfully as there is 100% success rate and 0% errors.

‘Monitoring’ section of Kinesis Firehose shows the amount of data delivered to the S3 bucket.

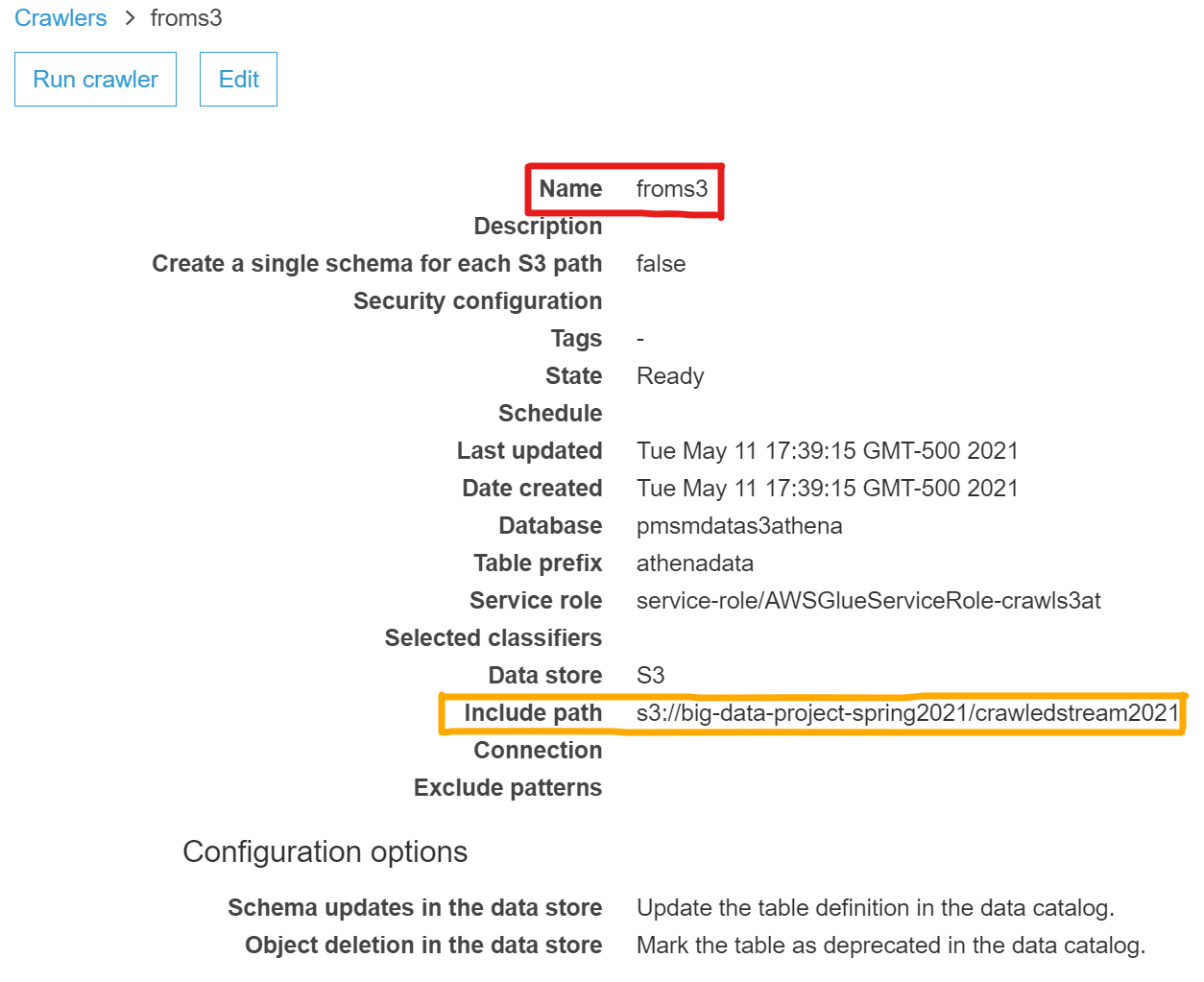


The above graph indicates there is 100% success in delivering the data to S3 bucket justifying the 100% success rate of streaming of lambda function.



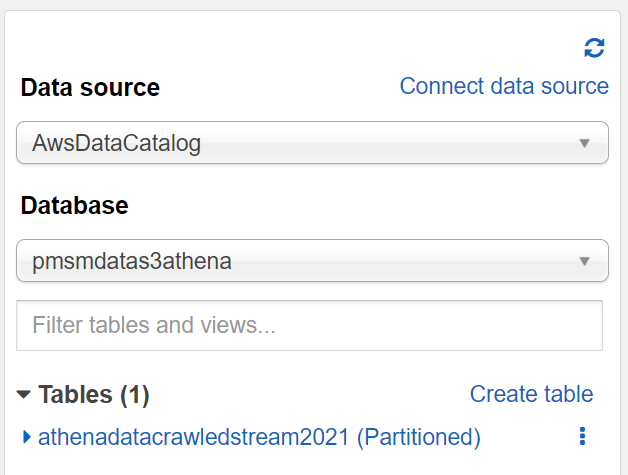
The above graph indicated that more than 65 records are delivered to S3 bucket.

The data stored in the S3 bucket in parquet form. To query in Athena, the data has been crawled using the AWS Glue Crawler.

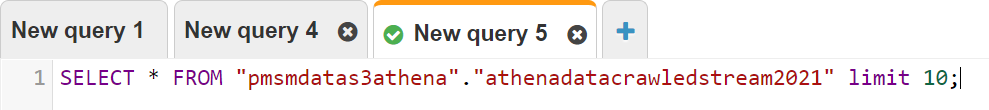


A crawler ‘**froms3’** (indicated by the red box above) is created mentioning the s3 bucket (orange box) where the DynamoDB data was delivered by firehose delivery stream. This crawls the data from S3 and enables it to be queried by Athena.

To query data in Athena, it has to be connected to a data source. The data source is the AWS Data Catalog where the database created by the crawler in the previous step exists.

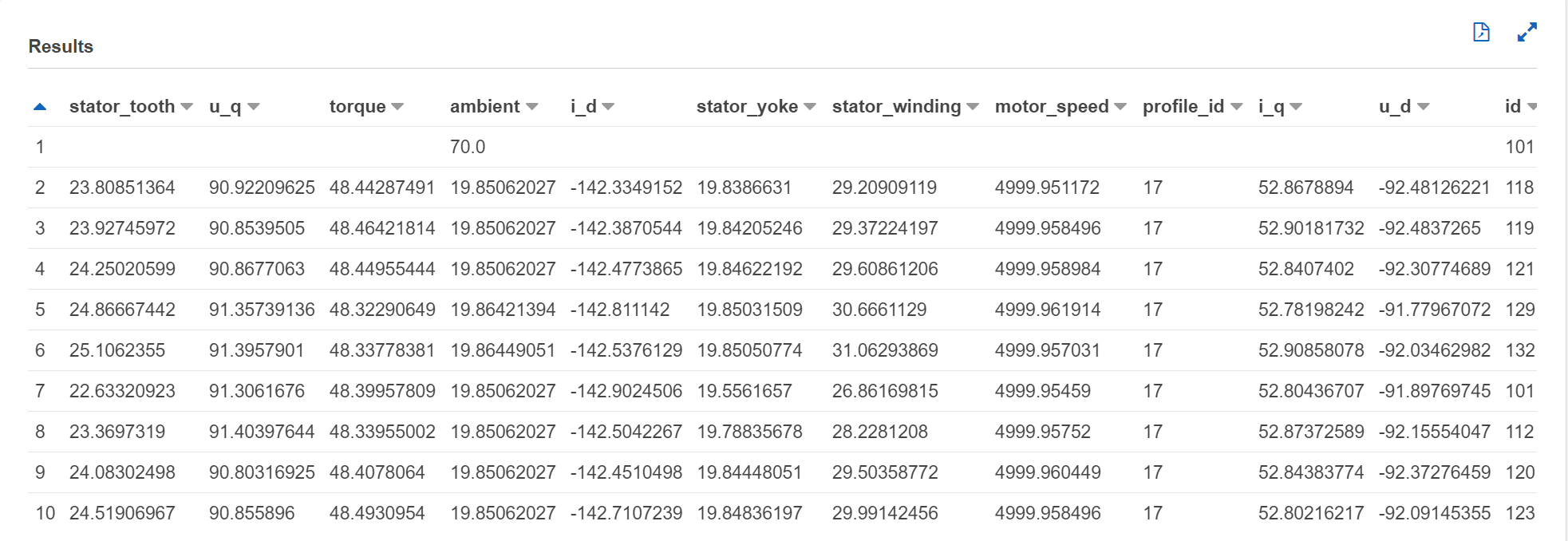


The query,

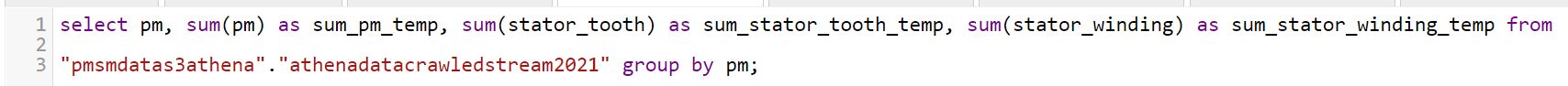


*Query 1*

Gives the following output,



Some queries necessary to do analysis in Quick Sight are:



*Query 2: To compare temperature of permanent magnet, stator tooth and stator winding*



*Query 3: count of readings in each batch*



*Query 4*



*Query 5*

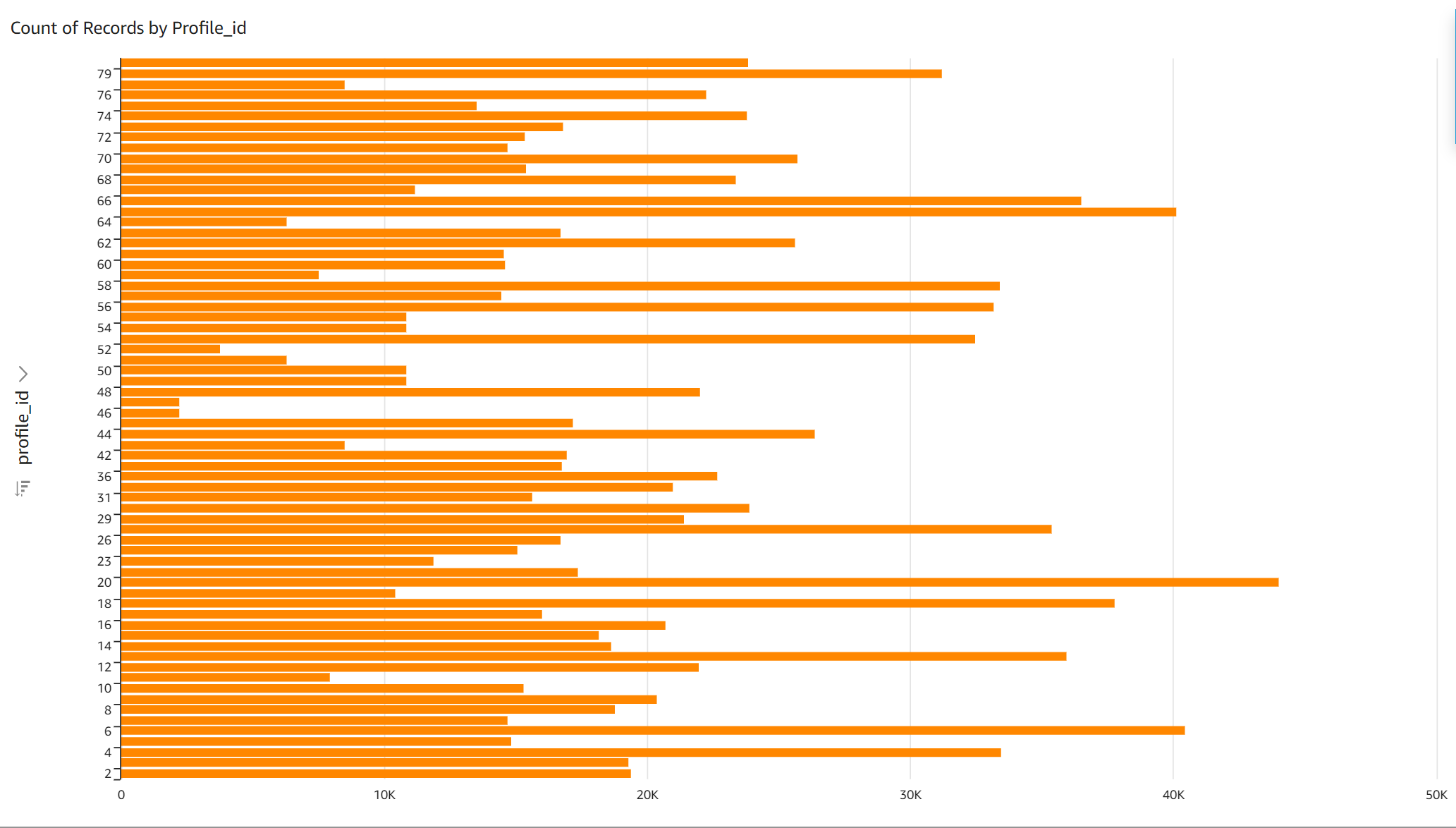


*Query 6*

Queries 4, 5 and 6 are needed for comparative analysis on each other.

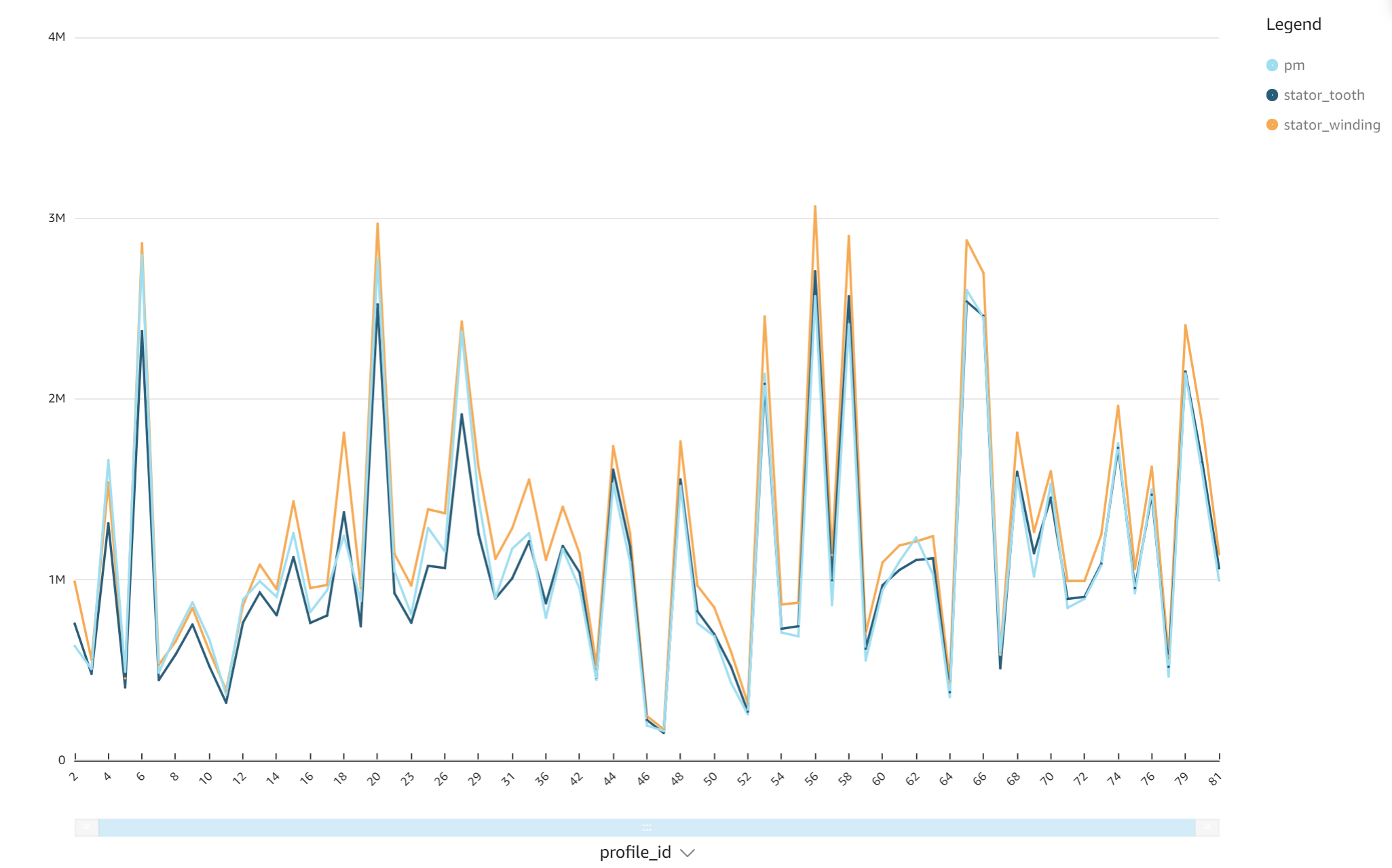
**QUICK SIGHT ANALYSIS**

There are batches of readings taken from PMSM machine, the below graph shows the total number of reading taken for every batch.

****

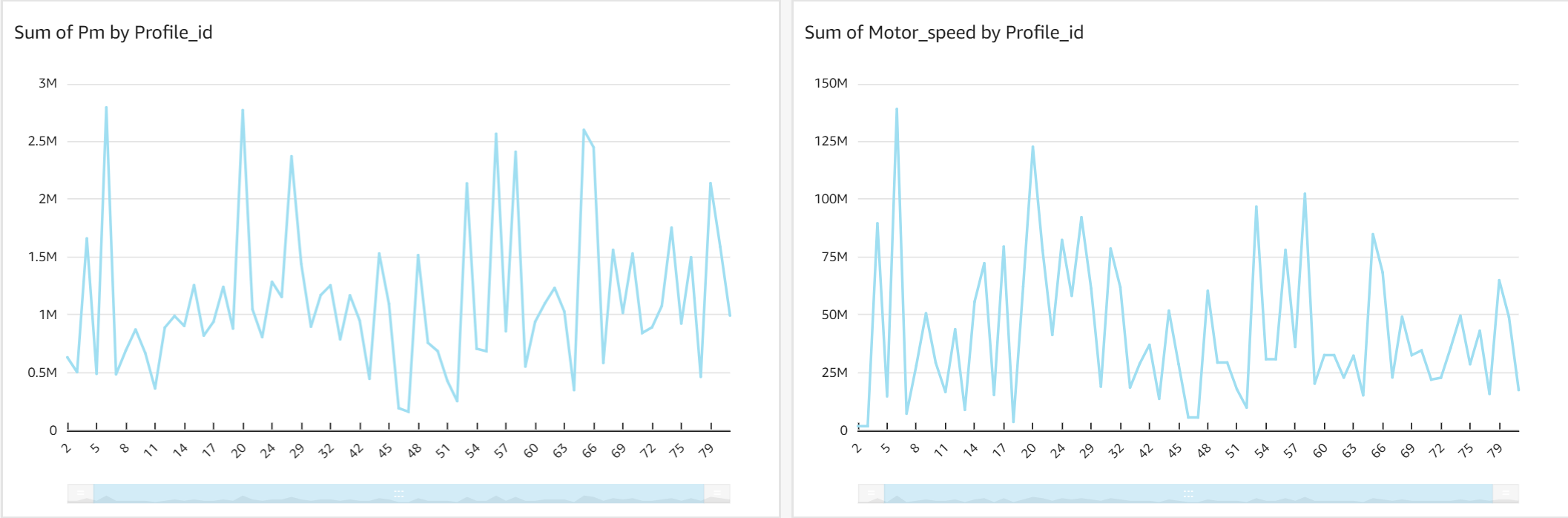
*Fig: Count of readings for each batch*

The temperature of every part of machine is important to make sure the machine functions effectively. Every part should maintain equal range of temperatures.



*Fig: Comparing temperature readings of the three components*

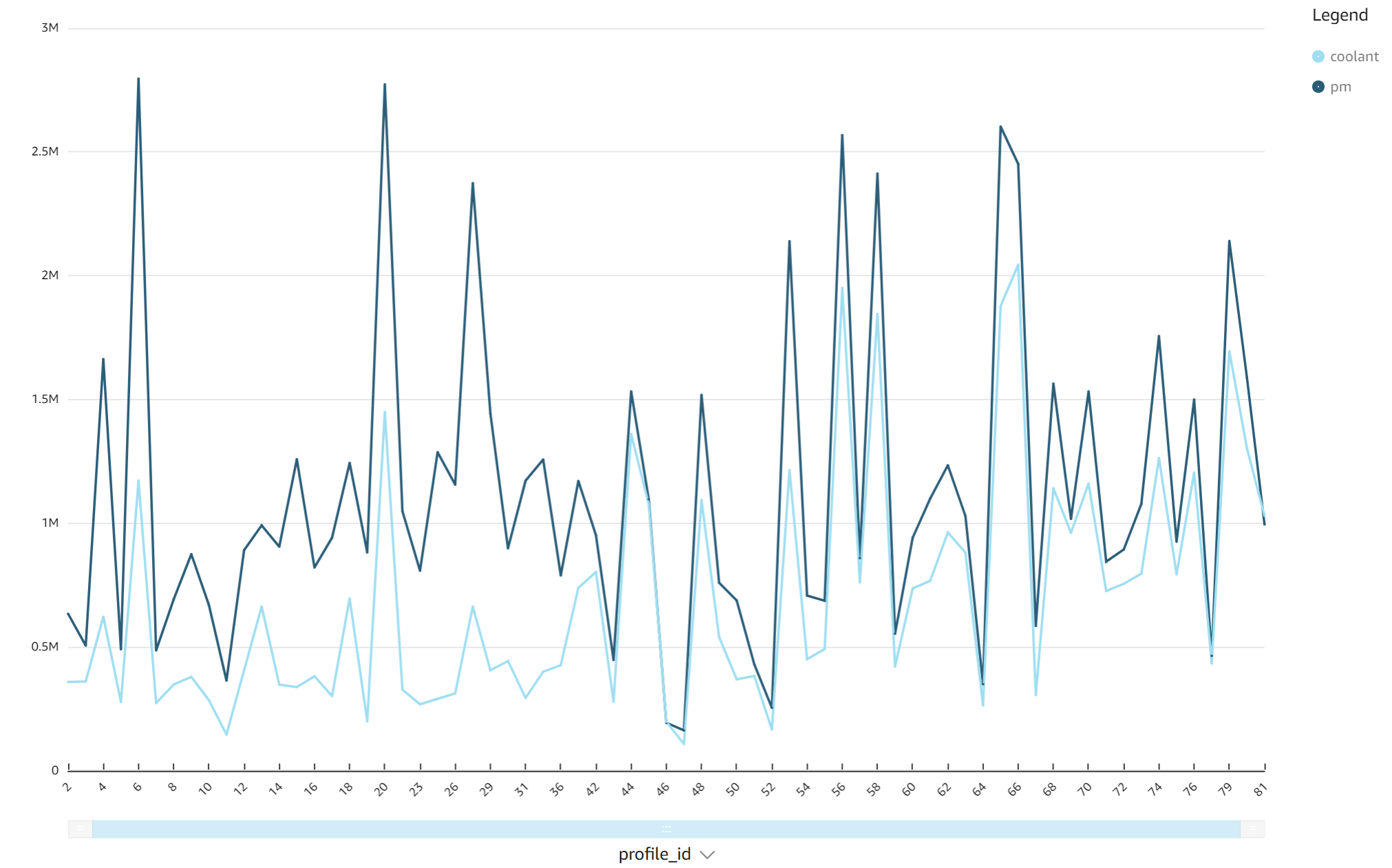
The temperature readings of permanent magnet, stator tooth and stator winding are plotted in the above graph. We can observe that there is an equal or same trend followed by all the three parts of the machine, maintaining the consistency. This says all three parts are working in harmony.



*Fig: temperature of permanent magnet according to motor speed*

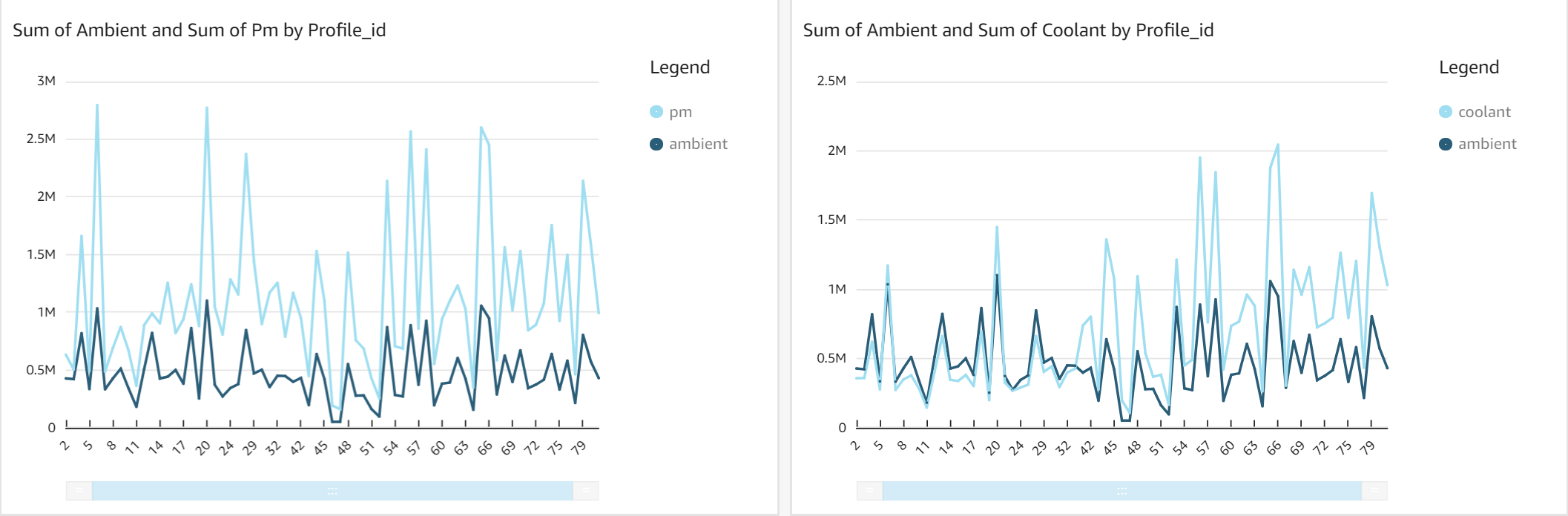
For the above figure, the left graph indicates temperature of permanent magnet and the right graph indicates the motor speed. In the initial batches, the temperature of permanent magnet is increasing along the motor speed. As it goes down the graph, even though the motor speed is decreasing temperature of permanent magnet is decreasing very slowly. It is because the temperature increased rapidly cannot decreased fast.

Now that the temperature is increasing, there is a need to cool the machine whenever a certain part is getting heated. The below graph shows that the temperature of the machine being maintained through a coolant when permanent magnet’s temperature is increasing.



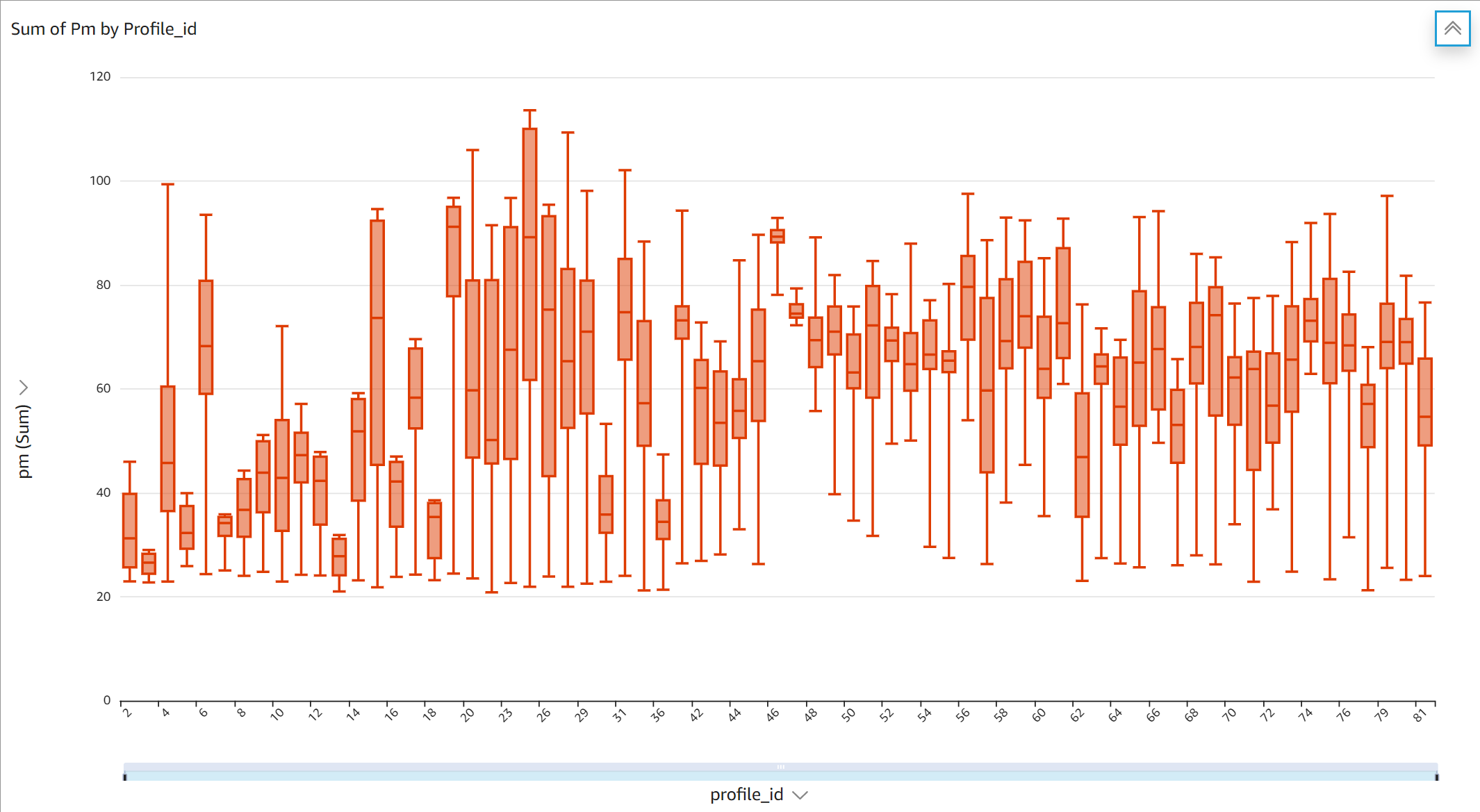
*Fig: Coolant temperature according to permanent magnet temperature*

Based on the permanent magnet temperature, we can set the coolant temperature appropriately. This data can be used to predict how much coolant temperature is to be set in future using time series analysis for future.



*Fig: temperature of permanent magnet according to ambient temperature (right) coolant temperature according to ambient temperature (left)*

The temperature of permanent magnet also depends on ambient (room) temperature (left graph) and the coolant temperature should be maintained according to the ambient temperature (right graph).



*Fig: Box plot for permanent magnet’s readings in every batch*

The above box plots of every batch of permanent magnet’s temperature readings shows there are no outliers. This indicates there is no suspicious increase or decrease in temperature of permanent magnet.

**COMPARATIVE ANALYSIS**

There are several reasons to reasons to consider DynamoDB over MongoDB and Kinesis Streaming over Kafka. The main reason would be the integration. As everything we choose is an AWS service, this makes it easy for a service to communicate with another service. Some of the other reasons are discussed below.

|  |  |
| --- | --- |
| DynamoDB | MongoDB |
| DynamoDB is a fully managed service meaning it reduces the time spent on operations, no hardware provisioning, no cluster scaling. The focus is entirely on application (data pipeline here). | MongoDB is exact opposite of this. It has an infrastructure to build, clusters to create and manage, replication of data and many more. It needs an operations person to take care of all this on top of an application designer. |
| DynamoDB has out of the box security. The security model is based on Identity and Access Management (IAM). This enables a secure access to AWS services and resources.  While building a Data Pipeline there will be so many leaking points as the data passes through various services. The data can be breached easily if there is no proper security. This is one of the strong reasons to choose DynamoDB over MongoDB. | MongoDB is secure, but the default configuration is not secure. It is potentially vulnerable to breaches as it does not provide out of the box security. |
| DynamoDB supports key-value queries. Queries involving graph traversals, aggregations support of other AWS services is required which increases the cost.  As our data pipeline, doesn’t need any of the graph traversal operations. It is better to stick with DynamoDB instead of managing the clusters in MongoDB. | The query language of MongoDB which is not key value helps in analyzing the data in many ways; Graph traversal, geospatial queries etc., |
| One of the disadvantages of being a fully managed service is we cannot tune the database elements such as index use, data models.  As the data pipeline doesn’t necessarily need a tuning to the database sticking to the fully managed service is better than posing a security threat to the pipeline as pipeline is vulnerable to it. | If necessary, MongoDB can obtain deeper levels of performance metrics granularity for optimization and tuning purposes. Some of the performance metrics include Resource Utilization, throughput metrics, Errors etc., |
| For projects with unpredictable demand, DynamoDB is the safest bet to cope with the growth of the market. | For small to medium scaled applications, MongoDB meets scalability and throughput requirements. |
| Netflix, Medium, New Relic are some of the companies which use DynamoDB. | Uber, Lyft, Code Academy are some of the companies which use MongoDB. |

|  |  |
| --- | --- |
| Kinesis | Kafka |
| Data is stored as Shards in Kinesis. | Data is stored as partitions in Kafka. |
| Kinesis supports a variety of SDKs like Java, Android, .NET, Go | Kafka supports only Java |
| Basic Skill level is required to create a Kinesis Data Stream making it easy to create one and stream data for a data pipeline | Advanced skill level is required to create a Kafka Steam. |
| Kinesis can write synchronously to 3 devices at a time making it easy for data pipeline to make different kind of analysis at same time. | Kafka has fewer limitations. It cannot do synchronous streaming. |
| Kinesis’s Data Retention Period is only 7 days. This doesn’t bother the pipeline built because all the data is being stored in DynamoDB table. | Kafka’s Data Retention period is configurable. |
| Kinesis is just pay and use | Kafka requires human support for cluster management and installation. |

**CONCLUSION**

A data pipeline is constructed making use of multiple AWS services to automate analysis process. A comparative analysis is made between DynamoDB and MongoDB, Kafka and Kinesis Stream to give advantages over one another in using them to construct a pipeline. Finally, visual analysis of readings is made using Quick Sight in real-time.

**REFERENCES**

[1] Dataset: <https://www.kaggle.com/wkirgsn/electric-motor-temperature>

[2] P. O Donovan, K. Leahy, K. Bruton, “An Industrial big data pipeline for data-driven analytics maintenance application in large-scale smart manufacturing facilities” Published in Springer Journal of Big Data, Article Number: 25, Year: 2015.

[3] Eveline van Stijn, David Hesketh, Yao-Hua Tan, Bram Klievink, Sietse Overbeek, Frank Heijmann, Markus Pikart, Tom Butterly, “The Data Pipeline” Conference paper for the United Nations Global Trade Facilitation Conference 2011.

[4] About Data Pipeline: <https://hazelcast.com/glossary/data-pipeline/>

[5] Stream Processing: <https://hazelcast.com/glossary/stream-processing/>

[6] Design Strategies for building a Data Pipeline:

<https://medium.com/@mrashish/design-strategies-for-building-big-data-pipelines-4c11affd47f3>