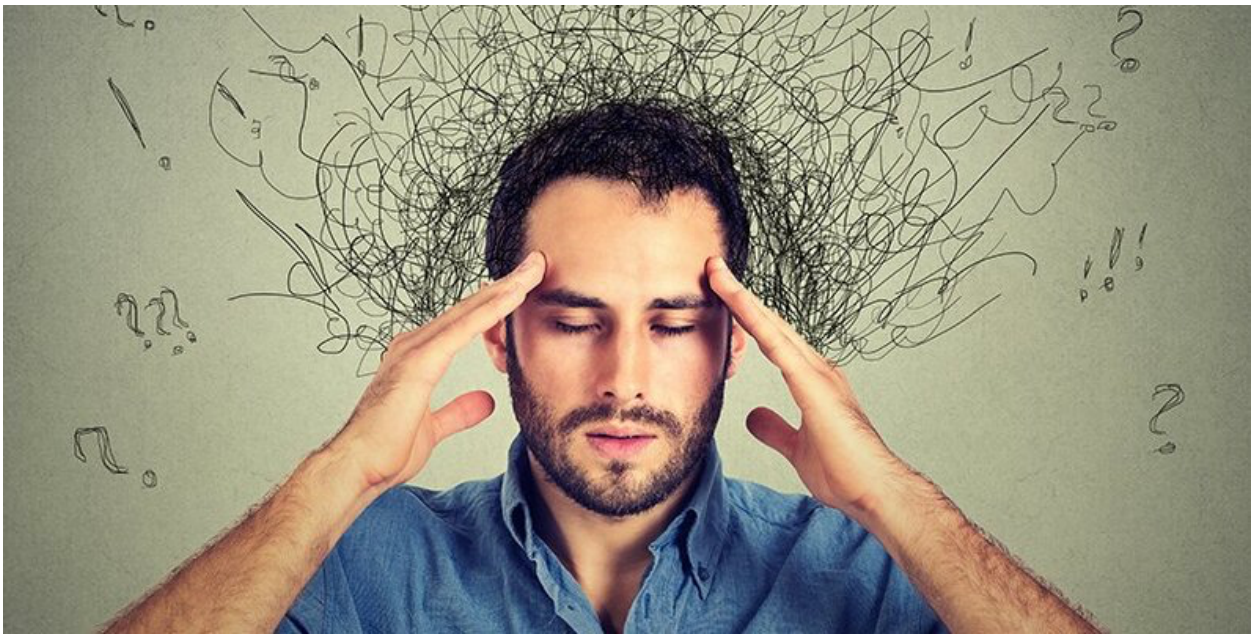
การออกแบบระบบวิเคราะห์ความเครียดจากข้อมูล Smart Watch บนคลาวด์แพลตฟอร์ม



โดย

6609036055 นันติยะ ศรีโคตร

6709036195 อัญชนา พิทักษ์วงษ์

6709036179 ธนวัสส์ ทั่งทอง

6709036146 อภิวิญช์ สุขวัฒณประเสริฐ

a. ที่มาและความสำคัญ เป้าหมายของโปรเจค

ในยุคที่ผู้คนต้องเผชิญกับความเร่งรีบและแรงกดดันจากการทำงานและการใช้ชีวิตประจำวัน ปัญหาเรื่อง “ความเครียด” กลายเป็นหนึ่งในปัญหาสุขภาพที่พบได้บ่อย และอาจส่งผลกระทบในระยะยาวต่อ ทั้งสุขภาพกายและใจ การเข้าใจและติดตามระดับความเครียดจึงมีความสำคัญต่อการป้องกันและดูแลสุขภาพในเชิงรุก

ด้วยความก้าวหน้าของเทคโนโลยีอุปกรณ์สวมใส่ (Wearable Devices) เช่น Smart watch ที่สามารถเก็บข้อมูลสุขภาพจากผู้ใช้ได้อย่างต่อเนื่อง เช่น อัตราการเต้นของหัวใจ จำนวนก้าว ระยะเวลาการนอน และระดับออกซิเจนในเลือด ทำให้เกิดแหล่งข้อมูลขนาดใหญ่ที่สามารถนำมาวิเคราะห์ เพื่อหาปัจจัยที่สัมพันธ์กับความเครียดของผู้ใช้งาน ได้

ชุดข้อมูล Smartwatch Health Data (Uncleaned) ที่นำมาใช้ในโปรเจคนี้ เป็นชุดข้อมูลจำลองข้อมูลที่ได้จาก Smart watch ที่เลียนแบบข้อมูลจริง สะท้อนสภาพข้อมูลจริงที่ได้จาก Smart watch ซึ่งมักมีปัญหาในด้านคุณภาพของข้อมูล เช่น ค่าที่หายไป (Missing), ค่าที่ผิดปกติ (Outlier), และค่าที่ไม่สอดคล้องกัน (Inconsistency) ส่งผลให้ไม่สามารถนำมาใช้งานได้โดยตรง

ดังนั้นการสร้างระบบที่สามารถจัดการกับข้อมูลสุขภาพเหล่านี้ให้พร้อมใช้งานทางวิเคราะห์และนำมาใช้ร่วมกับเทคโนโลยีคลาวด์จึงเป็นแนวทางที่เหมาะสม โดยเฉพาะการใช้บริการบน AWS ที่สามารถจัดการข้อมูลขนาดใหญ่ได้อย่างมีประสิทธิภาพ และรองรับการประมวลผลในแต่ละขั้นตอนของ Data Pipeline ตั้งแต่การนำเข้าข้อมูล (Ingestion) การจัดเก็บ การแปลงข้อมูล (ETL) การวิเคราะห์ไปจนถึงการแสดงผลผ่าน Dashboard

เป้าหมายของงานวิจัยนี้ คือการสร้างระบบ
วิเคราะห์และพยากรณ์ระดับความเครียดจากข้อมูลสุขภาพที่ได้จากอุปกรณ์ Smartwatch โดยสร้างระบบ Data Pipeline บน AWS ที่สามารถจัดการกับข้อมูลสุขภาพที่ยังไม่สะอาด และนำมาวิเคราะห์ โดย Machine Learning เพื่อคาดการณ์ระดับความเครียดในแต่ละวันของผู้ป่วย พร้อมทั้งนำเสนอข้อมูลผ่านแดชบอร์ด เพื่อให้แพทย์สามารถใช้ข้อมูลเหล่านี้ในการวางแผนรักษาและดูแลสุขภาพของผู้ป่วยได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

b. ชุดข้อมูลและรายละเอียดของชุดข้อมูลที่จะใช้

งานวิจัยนี้ใช้ชุดข้อมูล Smartwatch Health Data (Uncleaned) ซึ่งเผยแพร่โดย Mohammed Arfath R บนเว็บไซต์ Kaggle (https://www.kaggle.com/datasets/mohammedarfathr/smartwatch-health-data-uncleaned) โดยชุดข้อมูลนี้เป็นชุดข้อมูลที่ได้จากการจำลองข้อมูลการใช้งานสมาร์ตวอทช์ของผู้ใช้จริง ซึ่งถูกจัดเก็บในรูปแบบ Uncleaned หรือข้อมูลดิบ (Raw Data)
ที่ยังไม่ได้ผ่านกระบวนการทำความสะอาดหรือจัดรูปแบบที่เหมาะสมสำหรับการนำไปใช้วิเคราะห์โดยตรง ซึ่งถือเป็นจุดเริ่มต้นที่ดีในการฝึกการสร้างระบบ Data Pipeline เพื่อจัดการกับข้อมูลจริงที่มีคุณภาพหลากหลาย

ข้อมูลในชุดนี้แสดงผลลัพธ์รายวันของผู้ใช้งานและมีข้อมูลทั้งหมดจำนวน 10,000 แถว
โดยในแต่ละแถวของข้อมูล (Record) แสดงค่าทางสุขภาพในแต่ละวัน เช่น อัตราการเต้นของหัวใจ จำนวนก้าว ระยะเวลาการนอน รวมถึงค่าที่ประเมินว่าเป็นระดับความเครียด ซึ่งมีช่วงค่าตั้งแต่ 1–10
และข้อมูลบางส่วนยังพบว่ามีค่า null หรือค่าที่ไม่สมเหตุสมผล (Outlier)
สะท้อนถึงลักษณะของข้อมูลจริงที่เกิดขึ้นจากอุปกรณ์ IoT ซึ่งข้อมูลประกอบด้วย 7 ฟีเจอร์ ดังแสดงในตารางที่ 1

ฟีเจอร์	**คำอธิบาย**
user_id	หมายเลขประจำตัวของผู้ใช้ (ไม่ระบุตัวตน)
heart_rate	อัตราการเต้นของหัวใจ (หน่วย: BPM)
blood_oxygen	ปริมาณออกซิเจนในเลือด (หน่วย: %)
step_count	จำนวนก้าวที่เดินในแต่ละวัน
sleep_duration	ระยะเวลาการนอน (หน่วย: ชั่วโมง)
activity_level	ระดับกิจกรรมในแต่ละวัน (เช่น Sedentary, Moderate, Highly Active)
stress_level	ระดับความเครียดที่รายงาน (ช่วงคะแนน 1–10)

ตารางที่ 1 แสดงความหมายของฟีเจอร์

ลักษณะของข้อมูลจัดอยู่ในรูปแบบ **Time Series** ซึ่งสะท้อนพฤติกรรมสุขภาพของผู้ใช้ในแต่ละวัน ทำให้สามารถนำมาใช้วิเคราะห์ความเปลี่ยนแปลงและแนวโน้มได้เป็นอย่างดี อย่างไรก็ตาม
เนื่องจากเป็นข้อมูลที่ยังไม่ผ่านการทำความสะอาด จึงพบปัญหาทั่วไป เช่น

- ค่าที่หายไป (Missing Values)
- ค่าที่ผิดปกติ (Outliers)
- การสะกดค่าที่ไม่สอดคล้องกัน (เช่น “Highly_Active” และ “Highly Active”)
- ข้อมูลที่ไม่สอดคล้อง

ด้วยเหตุนี้ ระบบที่พัฒนาขึ้นจะต้องสามารถจัดการข้อมูลเหล่านี้ให้พร้อมต่อการใช้งาน
ทั้งในด้านการเตรียม ข้อมูลเพื่อฝึกโมเดล Machine Learning
และการแสดงผลข้อมูลเชิงสุขภาพได้อย่างถูกต้องแม่นยำ
ซึ่งทำให้ชุดข้อมูลนี้เหมาะสมอย่างยิ่งสำหรับการใช้ในโครงการวิเคราะห์และพยากรณ์ระดับความเครียดบนระบบ Cloud-Based Data Pipeline

c. การวิเคราะห์คุณลักษณะ 5V ในการประมวลผลชุดข้อมูล

การประมวลผลข้อมูลในงานวิจัยนี้อ้างอิงจากหลัก 5V ของ Big Data ได้แก่ Volume, Velocity, Variety, Veracity และ Value ซึ่งช่วยให้สามารถออกแบบระบบ Data Pipeline ได้อย่างเหมาะสมตามลักษณะของข้อมูลและการใช้งานจริง ดังรายละเอียดต่อไปนี้:

1. Volume (ปริมาณข้อมูล)

ชุดข้อมูล Smartwatch Health Data (Uncleaned) ที่ใช้ในโครงการมีจำนวน 10,000 แถว ซึ่งถูกนำมาใช้เพื่อการฝึกโมเดล Machine Learning และออกแบบโครงสร้างของระบบ Data Pipeline เท่านั้น แม้ว่าชุดข้อมูลจะไม่ใหญ่มากนักในเชิงปริมาณ แต่เมื่อพิจารณาว่าแต่ละแถวประกอบด้วยหลายฟีเจอร์ด้านสุขภาพ เช่น อัตราการเต้นของหัวใจ จำนวนก้าว ระยะเวลาการนอน และระดับความเครียด ข้อมูลดังกล่าวก็มีความหลากหลายเชิงมิติสูง และมีแนวโน้มที่จะขยายปริมาณมากขึ้นเมื่อใช้กับข้อมูล streaming จริงในระบบ ดังนั้นระบบจึงถูกออกแบบให้สามารถรองรับการจัดเก็บและประมวลผลข้อมูลในระดับ Big Data โดยใช้ Amazon S3 เป็น Data Lake ที่สามารถขยายตัวได้ง่ายเมื่อมีข้อมูลจำนวนมากในอนาคต

2. Velocity (ความเร็วของข้อมูล)

งานวิจัยนี้ได้ออกแบบระบบรองรับข้อมูลที่ไหลเข้ามาอย่างต่อเนื่อง (Data Streaming) โดยจำลองการรับข้อมูลจาก Smartwatch จริงผ่าน AWS Cloud9 และส่งข้อมูลทุก 3 วินาทีเข้าสู่ระบบผ่าน AWS Kinesis Firehose จากนั้นข้อมูลจะถูกจัดเก็บไว้ใน S3 Bucket โฟลเดอร์ raw/ ก่อนที่จะถูกนำไปแปลงในขั้นตอน ETL ซึ่งทำงานทุก 24 ชั่วโมงโดยอัตโนมัติผ่าน AWS EventBridge และ AWS Glue ดังนั้นแม้ว่าระบบจะประมวลผลแบบ Batch รายวันเพื่อความเหมาะสมกับลักษณะข้อมูลสุขภาพ เช่นการนอนหลับที่ต้องวัดแบบรายวัน แต่โครงสร้างของระบบก็พร้อมขยายสู่การรองรับข้อมูลแบบ Real-Time ได้ในอนาคตหากต้องการความถี่ที่สูงขึ้น

3. Variety (ความหลากหลายของข้อมูล)

ข้อมูลมีหลายประเภท ได้แก่:

- ข้อมูลเชิงตัวเลข (heart rate, step count, sleep duration)
- ข้อมูลเชิงหมวดหมู่ (activity level)
- ข้อมูลเวลา (timestamp)
- ข้อมูลที่อาจอยู่ในรูปแบบ string หรือ missing และข้อมูลที่มีความไม่สอดคล้องกัน เช่น activity level ที่สะกดแตกต่างกัน

4. Veracity (ความถูกต้องและเชื่อถือได้)

ข้อมูลที่ใช้มีข้อจำกัดในเรื่องคุณภาพ เช่น

- ค่าที่ขาดหาย (Missing values)
- ค่าที่ผิดปกติ (Outliers) เช่น Heart Rate สูงเกิน 220 bpm หรือระดับออกซิเจนต่ำกว่าปกติ

- ข้อมูลที่ไม่สอดคล้องกัน เช่น ข้อมูลที่สะกดผิด หรือข้อมูลซ้ำ (Duplicate)

ขั้นตอนการตรวจสอบและทำความสะอาด (Data Cleansing) จึงมีความจำเป็น โดยใช้ AWS Glue เพื่อจัดการข้อมูลให้อยู่ในรูปแบบที่เหมาะสม เช่น เติมค่าที่หายไป (Imputation), แปลงค่าผิดปกติ หรือคัดกรองข้อมูลที่ไม่สมเหตุสมผล ก่อนนำไปวิเคราะห์ เพื่อให้สามารถพยากรณ์ได้อย่างแม่นยำ

5. Value (คุณค่าของข้อมูล)

ข้อมูลที่ผ่านการประมวลผลสามารถนำมาใช้ในการพัฒนาโมเดล Machine Learning บน AWS SageMaker เพื่อพยากรณ์ระดับความเครียดของผู้ป่วย และนำไปแสดงผลผ่านแดชบอร์ดใน AWS QuickSight ซึ่งมีระบบรหัสสีที่ช่วยให้ผู้ใช้สามารถมองเห็นสถานะความเครียดของผู้ป่วยในแต่ละวันได้อย่างชัดเจน ช่วยสนับสนุนการรักษาและการวางแผนดูแลสุขภาพได้อย่างมีประสิทธิภาพ
อีกทั้งระบบยังสามารถต่อยอดไปสู่การแจ้งเตือน การวิเคราะห์เชิงสถิติ
และการสร้างระบบแนะนำสุขภาพได้ในอนาคต

d.สถาปัตยกรรม Data Pipeline พร้อมการประเมินการออกแบบซึ่งอิงจาก AWS Well Architected Framework: Data Analytics Lens

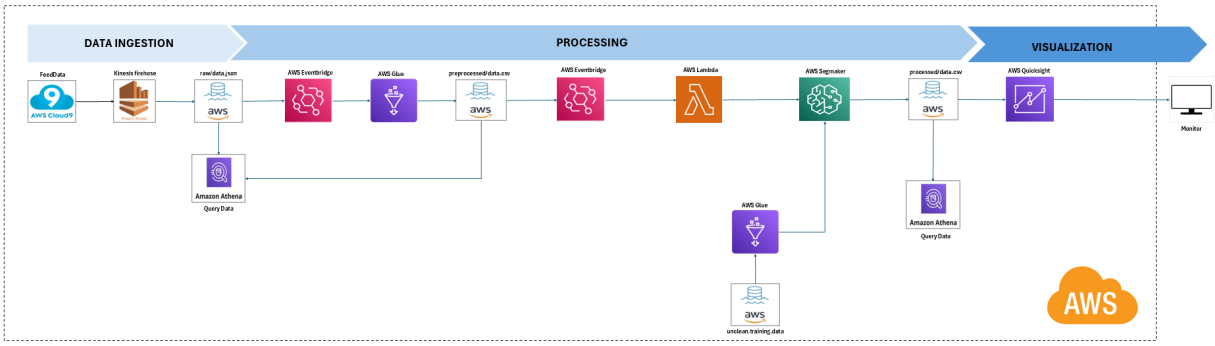
จุดแข็ง:

- ใช้บริการแบบ managed (เช่น AWS Glue, Athena, QuickSight) ทำให้ลดภาระด้านการดูแลระบบ
- Automate ด้วย Glue Crawler และ Lambda (จัดการ pipeline ได้ง่าย)

ข้อเสนอแนะ:

- เพิ่ม CloudWatch เพื่อติดตามการทำงานแบบ Real-time
- ใช้ Step Functions เพื่อ orchestration ที่ชัดเจนในขั้นตอน processing
- จัดทำ runbooks และ playbooks สำหรับทีมเมื่อเกิดเหตุขัดข้อง

e. รายละเอียดของ Data Pipeline Implementation ในแต่ละขั้นตอน



รูปที่ 1 แสดง **Pipe Line** *ของโครงการ*

Ingestion data

ในโปรเจคนี้ได้นำชุดข้อมูลจากเว็บไซต์ชื่อ “Smartwatch Health Data (Uncleaned)” มาใช้ ชุดข้อมูลประกอบด้วยฟีเจอร์หลัก ได้แก่ User ID, Heart Rate (BPM), Blood Oxygen Level (%), Step Count, Sleep Duration (hours), Activity Level และ Stress Level โดยจะนำข้อมูลชุดนี้ไปใช้ในการเทรนนิ่งโมเดลเพื่อทำนายค่า Stress Level ของข้อมูลใหม่ที่ได้รับผ่านระบบ Streaming ในโปรเจคนี้ กำหนดให้คอลัมน์ Stress Level เป็น target ที่ต้องทำนาย อย่างไรก็ตาม จุดด้อยของชุดข้อมูลนี้คือ ข้อมูลที่มาและวิธีการคำนวณค่า Stress Level ไม่ได้ถูกระบุอย่างละเอียดจากผู้จัดทำชุดข้อมูลเดิมซึ่งทำให้มีข้อจำกัดด้านความน่าเชื่อถือและการตรวจสอบความถูกต้องของข้อมูล (data validation) อย่างไรก็ตาม ผู้จัดทำโปรเจคนี้ใช้ชุดข้อมูลดังกล่าวเป็นตัวอย่างเบื้องต้นในการพัฒนาโมเดลเท่านั้น

เมื่อดาวน์โหลดชุดข้อมูล “unclean_smartwatch_health_data” มาแล้ว จะทำการตรวจสอบเบื้องต้นเพื่อดูความเหมาะสมของข้อมูล เช่น

1. ตรวจสอบโครงสร้างข้อมูล (Schema)
    - ดูว่าชุดข้อมูลมีคอลัมน์อะไรบ้าง และแต่ละคอลัมน์มีชนิดข้อมูล (data type) อย่างไร พบว่าชุดข้อมูลมีช่องว่างหรืออักขระพิเศษที่ไม่จำเป็น
2. ตรวจสอบข้อมูลที่ขาดหาย (Missing Values)
    - ตรวจพบว่ามีคอลัมน์ที่มีค่า null หรือ missing มากเกิน
    - ประเมินแล้วว่าควรจะเติมค่า (imputation) หรือกำจัดแถวที่ขาดข้อมูล
3. ตรวจสอบความถูกต้องของข้อมูล (Data Validity)
    - พบว่ามีค่าที่ผิดปกติ (outliers) หรือค่าที่ไม่สมเหตุสมผล เช่น ค่า Heart Rate สูงเกินจริง (>220 bpm)

- ตรวจสอบช่วงค่าของแต่ละฟีเจอร์ว่าตรงตามความเป็นจริง เช่น Step Count ต้องเป็นจำนวนเต็มบวก

4. ตรวจสอบความสม่ำเสมอของข้อมูล (Consistency)
    - ตรวจสอบว่าข้อมูลประเภทตัวเลขอยู่ในรูปแบบเดียวกัน เช่น มีหน่วยเดียวกันหรือไม่
    - ตรวจพบว่ามีคำที่ไม่สอดคล้องกัน เช่น Activity Level ที่ผิดสะกด หรือมีรูปแบบไม่เหมือนกัน (Highly_Active vs Highly Active)
5. ตรวจสอบข้อมูลซ้ำ (Duplicate Records)
    - ตรวจสอบว่ามีข้อมูลแถวใดซ้ำกันหรือไม่ และพิจารณาว่าควรลบหรือไม่
6. ตรวจสอบการกระจายตัวของข้อมูล (Data Distribution)
    - ดูการกระจายตัวของตัวแปรสำคัญ เช่น Step Count, Stress Level เพื่อเข้าใจลักษณะข้อมูลเบื้องต้น
7. ตรวจสอบความสัมพันธ์ของข้อมูล (Correlation)
    - วิเคราะห์ความสัมพันธ์เบื้องต้นระหว่างฟีเจอร์ต่างๆ กับ target เช่น ความสัมพันธ์ระหว่าง Activity Level กับ Stress Level

เมื่อตรวจสอบข้อมูลเบื้องต้นไปแล้ว ทำการ Cleaning ชุดข้อมูลชุดนี้ด้วยการสร้าง GlueJob ชื่อ preprocessing-data-trainning โดยการสร้าง python script โดยมีรายละเอียดดังนี้

1. Casting Data Type (Safe Casting)
    - แปลงค่าคอลัมน์ที่ควรเป็นตัวเลข เช่น Heart Rate, Blood Oxygen, Step Count, Sleep Duration ให้เป็นชนิดตัวเลข Float หรือ Int
    - ใช้ฟังก์ชันตรวจสอบก่อนแปลง เพื่อป้องกัน error กรณีข้อมูลมีรูปแบบไม่ถูกต้อง เช่น ค่าที่ไม่ใช่ตัวเลข
2. จัดการกับค่าผิดปกติและข้อมูลหาย
    - กรองข้อมูลให้เฉพาะช่วงค่าที่สมเหตุสมผลโดยช่วง HEART RATE ให้อยู่ในช่วง 40 ถึง 150 BPM อ้างอิงจาก (TARGET HEART RATES CHART, N.D.) และกรอง BLOOD OXYGEN LEVEL ให้อยู่ในช่วง 82 ถึง 100 % อ้างอิงจาก (PULSE OXIMETRY, N.D.) เพื่อกรองค่า OUTLIER และข้อมูลผิดปกติที่อาจส่งผลเสียต่อโมเดลและวิเคราะห์ ข้อมูลที่ไม่ได้อยู่ในช่วงจะทำการตัดแถวข้อมูลนั้นออก (FILTERING) หรือ แทนค่าผิดปกติด้วยค่าเฉลี่ย / มัธยฐาน (IMPUTATION) และใส่ค่าผิดปกติเป็น NULL จากนั้นเติมค่า MISSING ด้วยวิธีเติมค่าเฉลี่ยของข้อมูลจริงในแต่ละคอลัมน์
    - กรองข้อมูลแถวที่คอลัมน์สำคัญเช่น USER ID หรือ ACTIVITY LEVEL เป็นค่า NULL

- เติมค่าที่หายไป (MISSING VALUES) โดยใช้ค่าเฉลี่ย (MEAN) ของคอลัมน์นั้น ๆ เพื่อรักษาข้อมูลให้สมบูรณ์

3. แก้ไขค่าที่สะกดผิดและไม่สม่ำเสมอ
   - เช่น Activity Level ที่สะกดผิด (Highly_Active ⟶ Highly Active, Actve ⟶ Active, Seddentary ⟶ Sedentary) เพื่อให้ข้อมูลสอดคล้องและเหมือนกันทั้งหมด
4. แปลงค่าคอลัมน์ Activity Level เป็นดัชนีเชิงตัวเลข (Indexing)
   - ใช้ StringIndexer เพื่อแปลงค่าตัวอักษร Activity Level เป็นตัวเลข ซึ่งง่ายต่อการนำไปใช้ในโมเดล Machine Learning
5. เพิ่มขั้นตอนการแปลง Activity Level ตาม WHO Guidelines จาก Step Count (*WHO Guidelines on Physical Activity and Sedentary Behaviour*, 2020) เหตุผลที่ต้องการแปลงค่า Activity Level ตาม WHO Guidelines เนื่องจากมีความไม่แน่นอนของ raw Activity Level ของ dataset ดั้งเดิม เช่น step count ต่ำกว่า 100 แต่ activity level คือ Highly_Active ไม่สอดคล้องในความเป็นจริง ดังนั้น ทางผู้จัดทำโปรเจคจึงต้องการหามาตรฐานที่น่าเชื่อถือได้มากำหนดข้อมูลของ Activity level โดยกำหนดตาม WHO Guidelines คำนวนหา Activity level จาก Step Count เพื่อแบ่งระดับกิจกรรมทางกายได้ชัดเจนและเป็นมาตรฐานสากล ใช้เกณฑ์ในการกำหนดดังนี้

< 5,000 ก้าว/วัน	Sedentary
5,000-7,499 ก้าว/วัน	Low Active
7,500-9,999 ก้าว/วัน	Somewhat Active
≥ 10,000 ก้าว/วัน	Active หรือสูงกว่า

ตารางที่ 2 แสดงระดับของ **Activity Level**

ช่วยให้การนิยาม Activity Level เป็นไปตามเกณฑ์วิทยาศาสตร์และสุขภาพที่ได้รับการยอมรับมากยิ่งขึ้น อาจจะช่วยเพิ่มคุณภาพของฟีเจอร์ และให้สอดคล้องกับมาตรฐานจริง

6. เมื่อตรวจสอบคอลัมที่จะเลือกเป็น label ในการเทรนนิ่งโมเดลแล้วพบว่า stress level มีค่าปรากฏอยู่ช่วงระดับ 1-5

Results (10)

Search rows

#	stress_level	count
1	0	0
2	1	4512
3	2	1352
4	3	945
5	4	675
6	5	1590
7	6	0
8	7	0
9	8	0
10	9	0

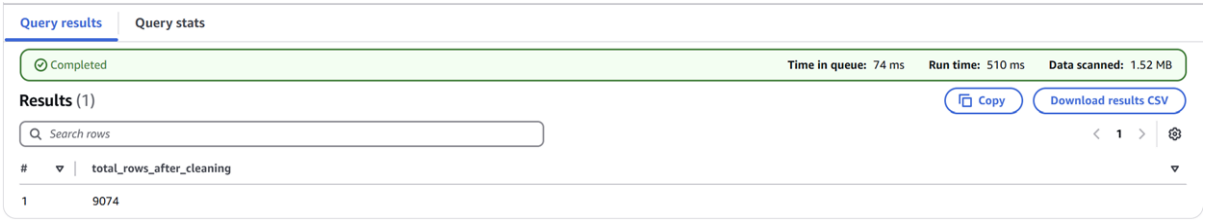
รูปที่ 2 แสดง **Stress level 1-5**

จึงปรับค่าช่วงของ Stress level ให้อยู่ในช่วง 0-4 สาเหตุที่เริ่มที่ 0 เพราะให้ง่ายต่อการนำไปเทรนนิ่งตัวโมเดล อ้างอิงช่วงความเครียดจาก (Three Studies Supporting the Initial Validation of the Stress Numerical Rating Scale-11 (Stress NRS-11): A Single Item Measure of Momentary Stress for Adolescents and Adults, n.d.)

ค่าดั้งเดิม (1–10)	ค่าปรับใหม่ (0–4)	ความหมายใหม่ (สำหรับตีความ)
1–2	0	ต่ำมาก (Very low)
3–4	1	ต่ำ (Low)
5–6	2	ปานกลาง (Moderate)
7–8	3	สูง (High)
9–10	4	สูงมาก (Very high)

รูปที่ 3 แสดงการกำหนดค่าระดับความเครียดของโครงการ

7. บันทึกข้อมูลในการเทรนนิ่งโมเดลที่ผ่านการ Clean แล้วส่งออกเป็นไฟล์ JSON บันทึกไว้ที่ s3://stress-predict-project/data-training-preprocessed-result/ จากรูปภาพแสดงถึงข้อมูลหลังจากการทำ Cleaning Data เรียบร้อยแล้วเหลืออยู่ 9074 rows จาก 10000 rows



รูปที่ 4 แสดงสถานะของ Query *ข้อมูลเพื่อส่งไปยัง* S3

การเทรนนิ่งโมเดลและสร้าง Endpoint

เพื่อเรียกใช้โมเดลเมื่อมีข้อมูลเข้ามาใหม่ เริ่มจากการสร้าง Jupyter Notebook ใน sagemaker เพื่อใช้ script python เทรนนิ่งโมเดลจากชุดข้อมูลที่ผ่านการครีนนิ่งแล้วจาก

```
    num_round=100
)

# เริ่มเทรนโมเดล โดยชี้ไปที่ path ข้อมูล S3
xgb.fit({'train': TrainingInput(train_data_path, content_type='csv')})
```

```
[05/28/25 06:03:11] INFO     Ignoring unnecessary instance type: None.                    image_uris.py:530

                    INFO     SageMaker Python SDK will collect telemetry to help us better telemetry_logging.py:91
                             understand our user's needs, diagnose issues, and deliver
                             additional features.
                             To opt out of telemetry, please disable via TelemetryOptOut
                             parameter in SDK defaults config. For more information, refer
                             to
                             https://sagemaker.readthedocs.io/en/stable/overview.html#confi
                             guring-and-using-defaults-with-the-sagemaker-python-sdk.

                    INFO     Creating training-job with name:                             session.py:1042
                             sagemaker-xgboost-2025-05-28-06-03-11-137

2025-05-28 06:03:11 Starting - Starting the training job.....
```

จากนั้นสร้าง Endpoint เพื่อให้ **โมเดลของคุณพร้อมให้บริการแบบ Real-time inference (REST API)**

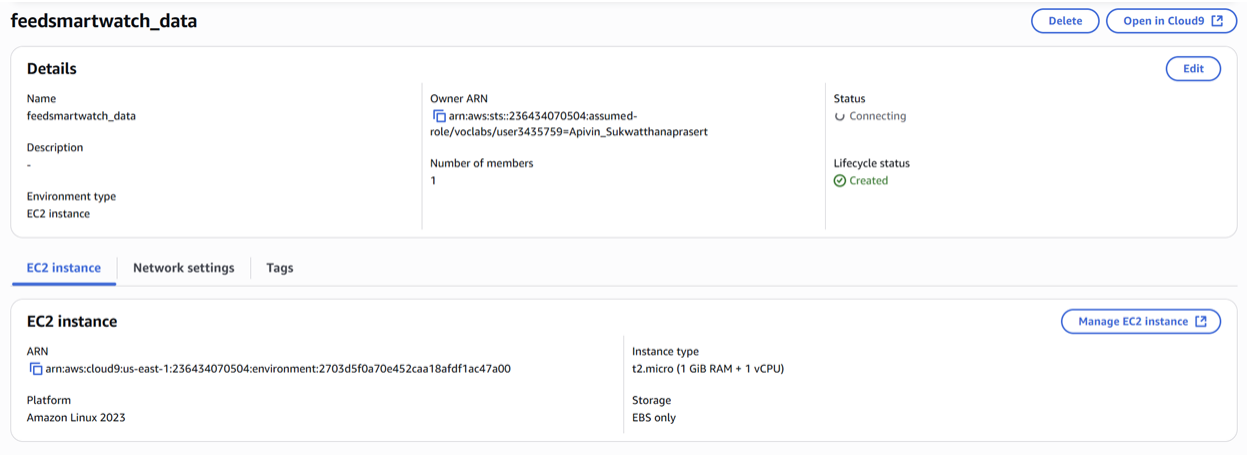
```
print(f"Deployed endpoint: {endpoint_name}")
```

```
[05/28/25 06:05:04] INFO     Ignoring unnecessary instance type: None.                    image_uris.py:530

                    INFO     Creating model with name: sagemaker-xgboost-2025-05-28-06-05-04-712    session.py:4094

[05/28/25 06:05:05] INFO     Creating endpoint-config with name xgboost-endpoint          session.py:6019

                    INFO     Creating endpoint with name xgboost-endpoint                 session.py:4841

--------!Deployed endpoint: xgboost-endpoint
```

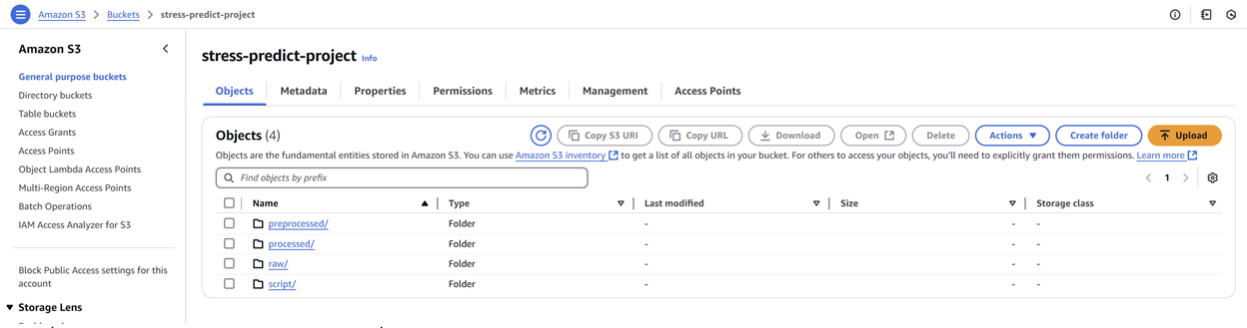
รายละเอียดของ Data Pipeline Implementation ในแต่ละขั้นตอน

1.AWS Could9

เริ่มต้องที่การ Steam Data จะเริ่มต้นจากการสร้างที่จัดเก็บข้อมูล โปรเจคนี้ใช้ S3 Bucket เพื่อจัดเก็บข้อมูลทั้งหมดในโปรเจคนี้ ทำการสร้าง Bucket โปรเจคกำหนดชื่อว่า stress-predict-project ต่อไปทำการสร้างโฟรเดอร์เพื่อเก็บข้อมูลในขั้นตอนต่างๆ แบ่งเป็น raw/ เป็นโฟรเดอร์ที่รับการป้อนข้อมูลจาก Cloud9 preprocessed/ รับข้อมูลที่ผ่านการทำ ETL แล้ว processed/ รับข้อมูลที่ผ่านการ predict ด้วย ML แล้วต่อไปจะทำการสร้างเครื่องมือในการ steam data ในโปรเจคนี้เลือกใช้ AWS firehose



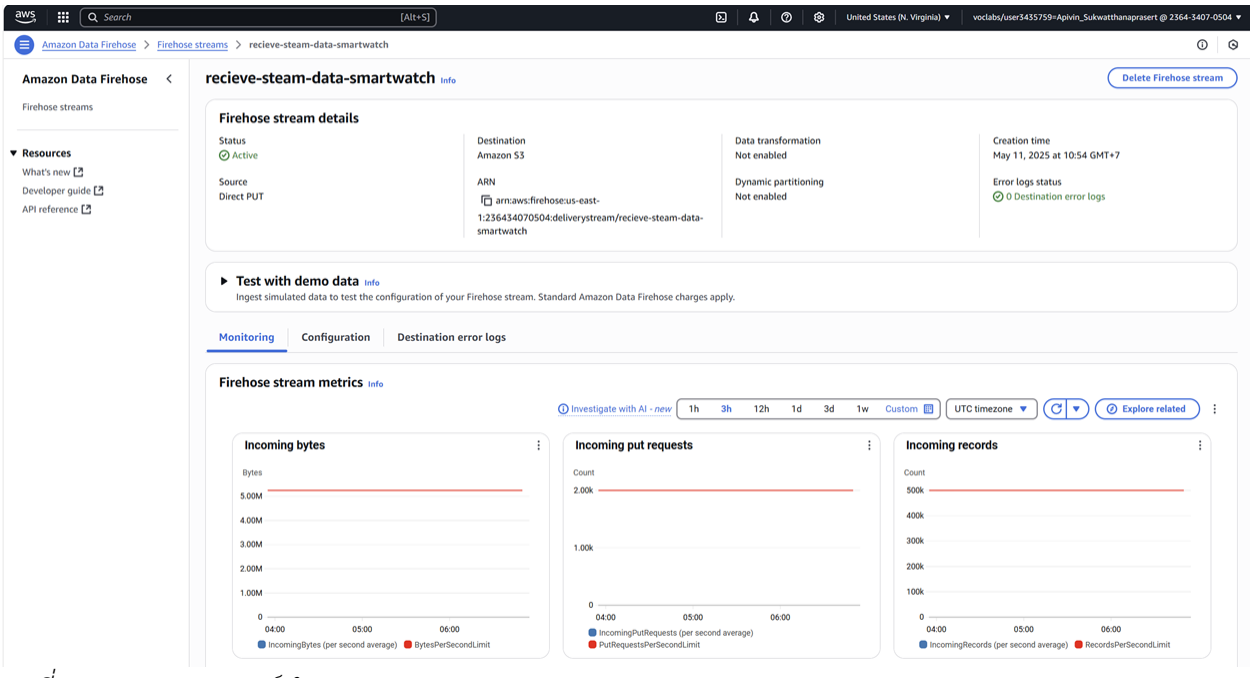
รูปที่ 5 แสดงการตั้งค่าการนำข้อมูลเข้าจาก Smart Watch



รูปที่ 6 แสดงการนำข้อมูลมาเก็บที่ S3

2. AWS Firehose

สร้างเครื่องมือ steam data ด้วย AWS firehose โปรเจคนี้กำหนดชื่อของ firehose ว่า recieve-steam-data-smartwatch เพื่อ steam data เข้าไปที่ S3 ไปที่ path : s3://stress-predict-project/raw/ ที่สร้างเอาไว้แล้ว



รูปที่ 7 แสดงสถานการณ์ทำงานของ AWS Firehose

3. AWS Cloud9

ต่อไปสร้างเครื่องที่ใช้ในการสร้างข้อมูลจำลองของ smartwatch โปรเจคนี้ใช้ Cloud9 ในการสร้างข้อมูลจำลองของ smart watch ขึ้นมาด้วยการสร้างสภาพแวดล้อมกำหนดชื่อว่า feedsmartwatch_data จากนั้นเข้าไปที่หน้า Cloud9 IDE เพื่อสร้างสคลิป Python ใช้ป้อนข้อมูลเข้า S3 bucket : raw/ ผ่าน AWS firehose โดยมีสคลิปดังนี้ :

```
import boto3
import json
import time
import random
import datetime
```

```
# Firehose setup
firehose = boto3.client('firehose', region_name='us-east-1')
stream_name = 'recieve-steam-data-smartwatch'

# Mapping activity score to label
def get_activity_level(step_count):
    if step_count < 5000:
        return "Sedentary"
    elif 5000 <= step_count <= 7499:
        return "Low active"
    elif 7500 <= step_count <= 9999:
        return "Moderately active"
    elif 10000 <= step_count < 12500:
        return "Highly active"
    else:
        return "Very highly active"

# เริ่มจากวันที่จำลอง (วันแรกที่ใช้)
simulated_date = datetime.date(2025, 5, 12)
last_refresh_time = datetime.datetime.utcnow()  # เวลาเริ่มจับเวลา 2 นาที

# ส่งข้อมูลทุก 3 วินาที
while True:
    now = datetime.datetime.utcnow()

    # ตรวจสอบว่าเกิน 2 นาทีหรือยัง ⟶ เปลี่ยนวัน
    if (now - last_refresh_time).total_seconds() >= 120:
        simulated_date += datetime.timedelta(days=1)
        last_refresh_time = now
        print("Date incremented:", simulated_date)
```

```
    # Generate smartwatch data
    step_count = random.randint(4000, 13000)
    smartwatch_data = {
        "UserId": str(random.randint(1500, 1600)),
        "bpm": str(round(random.uniform(55.0, 110.0), 3)),  # Updated BPM range
        "blood oxygen level": round(random.uniform(82.0, 100.0), 3),  # Updated blood oxygen level range
        "Step Count": str(step_count),  # Step count range between 4000 to 13000
        "Sleep Duration": str(round(random.uniform(1.0, 10.0), 3)),
        "Activity Level": get_activity_level(step_count),  # Updated activity level logic based on step count
        "timestamp": simulated_date.strftime('%Y-%m-%d')  # เฉพาะวัน
    }

    # Send data to Firehose
    response = firehose.put_record(
        DeliveryStreamName=stream_name,
        Record={'Data': json.dumps(smartwatch_data) + '\n'}
    )

    print(f"Sent: {smartwatch_data}")
    time.sleep(3)
```

สคลิปนี้จะทำงานทุก 3 วินาทีเพื่อป้อนข้อมูลเข้า Kinesis Forehose Name : 'recieve-steam-data-smartwatch' จะมีโครงสร้างของข้อมูลเป็นรูปแบบ .json มีโครงสร้างเป็น “{"UserId": "1541", "bpm": "69.578", "blood oxygen level": 99.419, "Step Count": "7925.395", "Sleep Duration": "6.368", "Activity Level": "Moderate", "timestamp": "2025-05-12"}” ประกอบด้วยส่วนต่างๆ ดังนี้

1. UserId: XXXX คือ smartwatchID ของผู้ใช้งาน
   ข้อมูลที่ป้อนเข้าไปจะมีไอดีที่แตกต่างกันหรือจะมีไอดีที่ซ้ำบ้างในบางไอดี ไอดีจะถูกสุ่มและสร้างใหม่ทุกๆ 3 วินาที
2. Bpm (ค่า Heart rate) จะถูกสร้างถูกสุ่มและสร้างใหม่ทุกๆ 3 วินาที การสุ่มจะอยู่ในช่วง 55 ถึง 110 เพราะเป็นค่าอยู่ระหว่างคนปกติทั่วไปกับคนที่การเต้นของหัวใจไม่ปกติ
   เนื่องจากอยากจะได้เคสของคนที่มีการเต้นของหัวใจที่ไม่ปกติด้วยอ้างอิงจาก (*Target Heart Rates Chart*, n.d.)
3. Blood oxygen level คือ ระดับออกซิเจนในเลือดจะถูกสร้างถูกสุ่มและสร้างใหม่ทุกๆ 3 วินาที การสุ่มจะอยู่ในช่วง 82 ถึง 100 กำหนดแบบนี้เพราะต้องการทั้งช่วงปกติและไม่ปกติด้วย
   อ้างอิงช่วงระดับออกซิเจนจาก (*Pulse Oximetry*, n.d.)

สถานะ	ช่วงค่า SpO_2 (%)	คำอธิบาย
ปกติ (Normal)	95% – 100%	ร่างกายได้รับออกซิเจนเพียงพอ
ต่ำกว่าปกติเล็กน้อย	90% – 94%	อาจเริ่มมีปัญหาในการแลกเปลี่ยนออกซิเจน
ภาวะขาดออกซิเจน (Hypoxemia)	< 90%	ต้องพบแพทย์ทันที หรือให้ออกซิเจนเพิ่มเติม
ระดับวิกฤต (Severe Hypoxemia)	< 85%	ภาวะฉุกเฉินทางการแพทย์ ต้องการการรักษาทันที

รูปที่ 8 แสดงสถานะของระดับเลือดในร่างกาย

4. Step Count การนับก้าวของแต่ละไอดีจะถูกสร้างถูกสุ่มและสร้างใหม่ทุกๆ 3 วินาที การสุ่มจะอยู่ในช่วง 4000 ถึง 13000 เป็นช่วงที่จะกำหนดถึงระดับ Activity Level ว่าอยู่ในระดับไหน ถ้าในช่วง <5000 ก้าวต่อวันกำหนดให้เป็น Sedentary ช่วง 5000 – 7499 ก้าวต่อวันกำหนดให้เป็น Low active ช่วง 7500 – 9999 ก้าวต่อวันกำหนดให้เป็น Moderately active ช่วงมากกว่าหรือเท่ากับ 10000 ก้าวต่อวันกำหนดให้เป็น Highly active ช่วงมากกว่าหรือเท่ากับ 12500 ก้าวต่อวันกำหนดให้เป็น Very highly active อ้างอิงจาก (*WHO Guidelines on Physical Activity and Sedentary Behaviour*, 2020)

ระดับกิจกรรม	จำนวนก้าวต่อวัน (โดยประมาณ)	คำอธิบาย
Sedentary (นั่งเฉยๆ)	< 5,000 ก้าว	แทบไม่มีการเคลื่อนไหว เช่น ทำงานออฟฟิศ นั่งเรียนทั้งวัน
Low Active	5,000 – 7,499 ก้าว	เดินบ้างในชีวิตประจำวัน เช่น ไปซื้อของ ทำงานบ้าน
Moderately Active	7,500 – 9,999 ก้าว	มีการเดินมากขึ้น หรือมีกิจกรรมที่ต้องเดินทั้งวัน
Highly Active	≥ 10,000 ก้าว	มีการออกกำลังกาย หรือทำงานที่ต้องเดิน/เคลื่อนไหวเยอะ
Very Highly Active	≥ 12,500 ก้าว	นักกีฬา หรือคนที่ใช้พลังงานร่างกายสูงมากในแต่ละวัน

รูปที่ 9 แสดงระดับกิจกรรมต่อวัน

5. Timestampg เป็นตัวกำหนดเวลาที่ข้อมูลถูกบันทึกเข้ามาข้อมูลที่ได้รับจากกระบวนการ ingestion จะถูกเก็บไว้ใน S3 bucket ที่ raw/ เพื่อเตรียมไปเข้าสู่กระบวนการ processing data ในกระบวนการนี้โปรเจ็คที่ทำเลือกใช้เครื่องดังนี้ AWS Event Bridge เพื่อจัดการกำหนดเวลาในการดึงข้อมูลต่อวัน เหตุผลที่มีการตั้งเวลาให้ทำงาน 1 วันต่อครั้งเพื่อรวมรวบข้อมูล Sleep Duration ให้ครบรอบต่อวันก่อนถึงจะนำข้อมูลมาใช้งานในกระบวนการ transform data โปรเจคนี้ใช้เครื่องมือของ AWS โดยเรียก GlueJob ในการทำ ETL ต่อไปคือขั้นตอนในกระบวนการ processing data สร้าง GlueJob เพื่อทำ ETL ข้อมูลจาก S3 Bucket raw/data.json
    1. สร้าง GlueJob โดยใช้ Script editor โดยกำหนดชื่อ GlueJob ว่า etl-raw-data
    2. กำหนด Role ให้กับ GlueJob และเลือกภาษาที่ใช้ในการสร้าง script โปรเจคนี้เลือกใช้ Python ในการสร้าง script ETL
    3. สร้าง script ด้วยภาษา Python เพื่อแปลงข้อมูล ข้อมูลที่ต้องการแปลงคือคอลัม Activity Level อยู่ในรูปแบบตัวอักษร ต้องการแปลงให้อยู่ในรูปแบบตัวเลขเพื่อบอกช่วงตัวเลขของ Activity level ซึ่งง่ายในการนำไปใช้พยากรณ์ด้วย Machine Learning จะกำหนดการแปลงข้อมูลดังนี้

Activity Level Description	Activiti Level
Sedentary	1
Low	2
Moderately	3
Highly	4
Very Highly	5

ตารางที่ 3 แสดงระดับ **activity level**

4. แยกข้อมูลตามวันที่ นำข้อมูลที่แปลงแล้วไปเก็บที่ S3 bucket preprocessed/

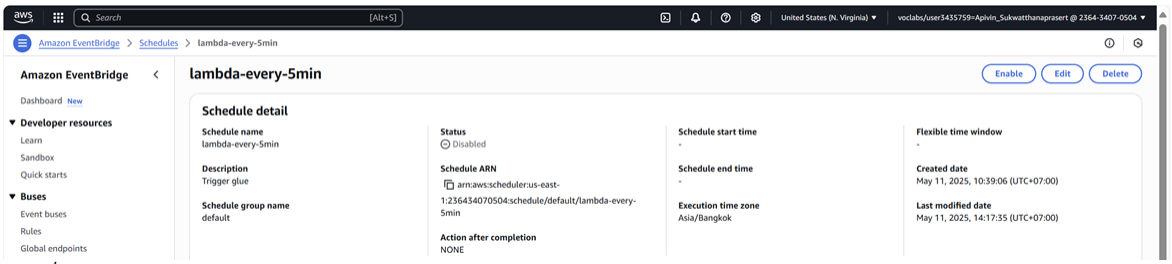
GlueJob Script :

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions
from pyspark.sql.functions import col, udf, to_date
from pyspark.sql.types import IntegerType

# Step 1: Glue Setup
args = getResolvedOptions(sys.argv, ['JOB_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# Step 2: Read JSON input from S3 (recursive lookup enabled)
input_path = "s3://stress-predict-project/raw/2025/"
df = spark.read.option("recursiveFileLookup", "true").json(input_path)

# Optional: Debug schema
df.printSchema()

# Step 3: UDF mapping Activity Level
def activity_to_num(level):
    if level == "Sedentary":
        return 1
```

```
    elif level == "Low active":
        return 2
    elif level == "Moderately active":
        return 3
    elif level == "Highly active":
        return 4
    elif level == "Very highly active":
        return 5
    else:
        return 0  # Default case in case an invalid level is encountered

map_udf = udf(activity_to_num, IntegerType())

# Step 4: Apply transformation
df_transformed = df.withColumn("Activity_Level_Num", map_udf(col("Activity Level")))
df_transformed = df_transformed.withColumn("date", to_date(col("timestamp")))

# Step 5: Write partitioned output
output_path = "s3://stress-predict-project/preprocessed/"
df_transformed.write.mode("overwrite").partitionBy("date").json(output_path)
job.commit()
```

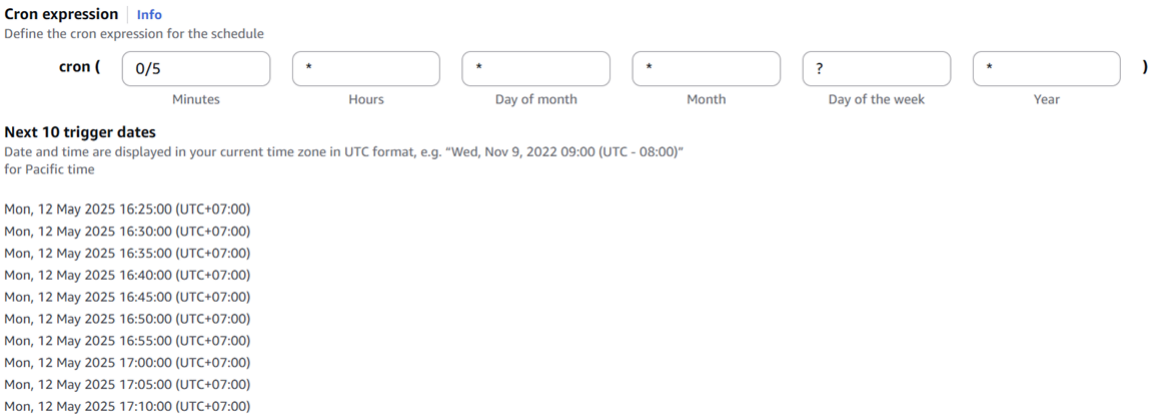
เมื่อสร้าง GlueJob เสร็จเรียบร้อยแล้ว ต่อไปคือการสร้างเครื่องมือในการเรียกใช้ GlueJob โดยมีการกำหนดเวลาเรียกใช้งานไว้ที่ 1 วันต่อครั้ง

1. จากการเรียกใช้งาน AWS EventBridge เพื่อสร้าง Schedules ขึ้นมา



รูปที่ 10 แสดงสถานการณ์ทำงานของ **AWS EventBridge**

2. กำหนดเวลาที่ต้องการให้ EventBridge เรียกใช้งาน GlueJob ในโปรเจคนี้เลือกใช้อยู่ที่ 24 ชม. แต่ใช่ช่วงสร้างโปรเจคเพื่อทดลองระบบ จึงกำหนดไว้ที่ 5 นาที เพื่อดูการทำงานของ EventBrigde



รูปที่ 11 แสดงการทำงานของ **Evenbrigde**

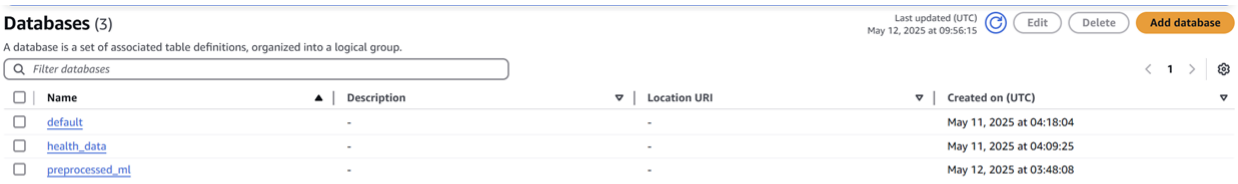
3. เลือก Target API เพื่อส่งคำสั่งไปให้ GlueJob ทำงานใน schedule นี้เลือก Target เลือก StartJobRun ใส่ข้อมูลในรูปแบบ JSON (JavaScript Object Notation) ที่ใช้ระบุค่าพารามิเตอร์ (parameters) ซึ่ง API จะนำไปใช้งานดังนี้

```
{
"JobName": "etl-raw-data"
}
```

Gluejob

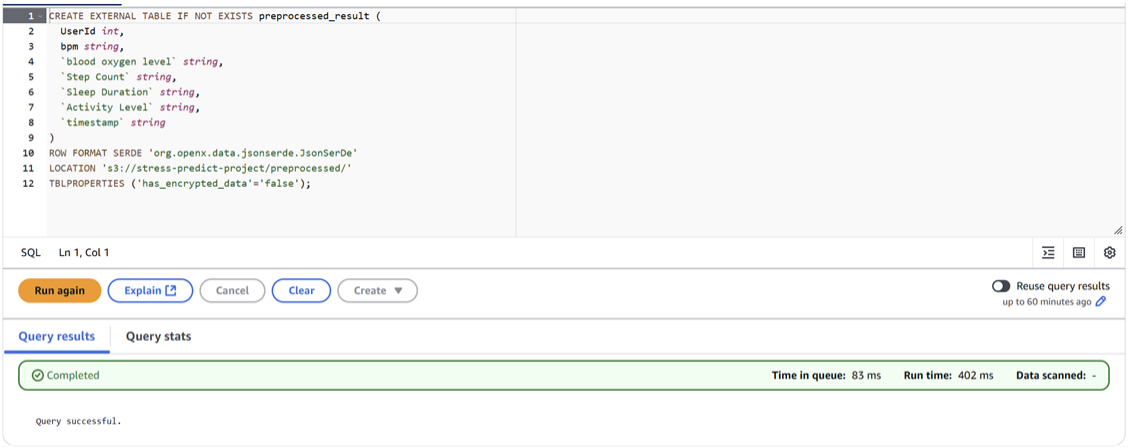
เมื่อสร้างในส่วน processing data เสร็จเรียบร้อย เมื่อเปิดใช้งาน schedule จะทำการเรียกใช้งาน GlueJob ตามที่เวลาได้ตั้งไว้อัตโนมัติ จากนั้นเมื่อได้ไฟล์ที่ผ่านกระบวนการ ETL เรียบแล้ว ต่อไปจะทำการตรวจเช็ครูปแบบข้อมูลเบื้องต้นว่ายังมีข้อมูลที่เป็น null หรือ ตัวหนังสือและข้อมูลถูกจัดเก็บครบถ้วนหรือไม่ ก่อนนำไปใช้งานในส่วนของ prediction ด้วยการ Query ข้อมูลผ่าน AWS Athena

1. เริ่มต้นด้วยการสร้าง Database ที่ AWS Glue โปรเจคนี้กำหนดชื่อ Database ไว้ว่า preprocessed_ml



รูปที่ 12 แสดงการสร้าง **AWS Gluejob**

2. เมื่อสร้าง database เรียบร้อยแล้วต่อไปสร้าง Table เพื่อ Query ข้อมูล ที่ AWS Athena เลือกใช้ SQL ด้านล่างนี้ในการสร้าง Table ชื่อ preprocessed_ml ขึ้นมา



รูปที่ 13 แสดงการสร้างตาราง **prepossessed_ml**

3. ดูข้อมูลเบื้องต้นว่ามีการแปลงคอลัม Activity Level ให้เป็นค่าตัวเลขแล้วหรือยังด้วยคำสั่ง SQL SELECT * FROM "preprocessed_ml"."preprocessed_result_ml" limit 100;
ข้อมูลที่ปรากฏขึ้นจะเห็นมีการเพิ่มคอลัม activity level num
ขึ้นมีซึ่งเกิดจากการแปลงข้อมูลให้เป็นตัวเลขตามรูปด้านล่าง

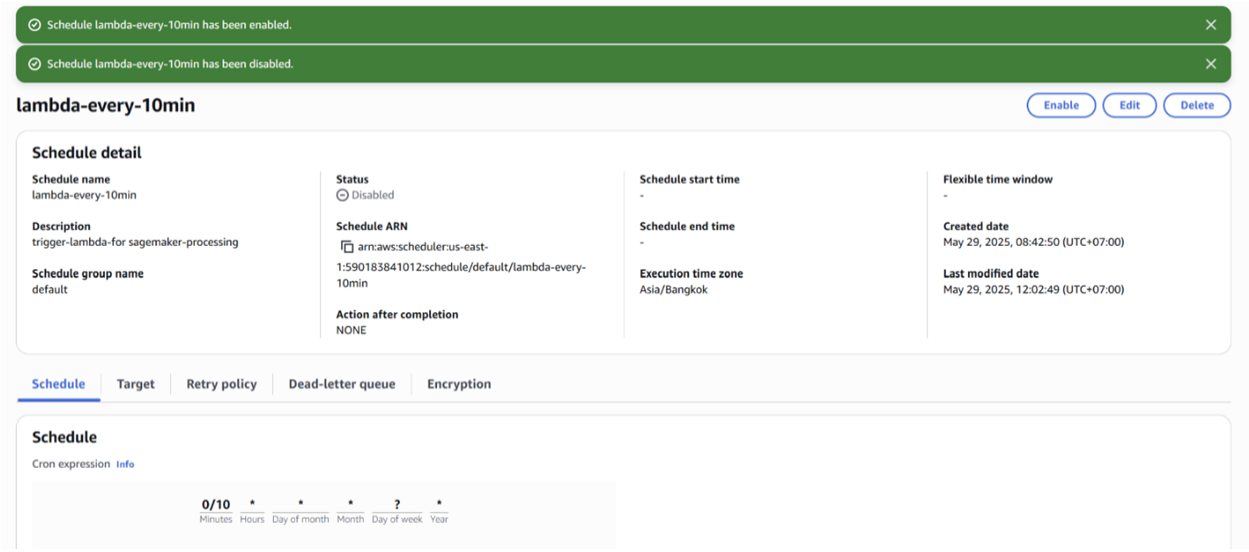
#	userid	bpm	blood oxygen level	step count	sleep duration	activity level	timestamp	activity_level_num
1	1508	70.754	90.317	7130.969	3.709	Moderate	2025-05-12T02:50:00Z	2
2	1577	77.87	97.292	7529.296	9.046	Moderate	2025-05-12T02:50:00Z	2
3	1518	79.217	95.575	7628.211	3.657	Moderate	2025-05-12T02:50:00Z	2
4	1520	65.234	95.53	6522.129	5.637	Highly_Active	2025-05-12T02:50:00Z	3
5	1530	73.937	99.778	6908.648	5.207	Moderate	2025-05-12T02:50:00Z	2
6	1586	63.106	92.258	7330.585	6.256	Low	2025-05-12T02:50:00Z	1
7	1556	63.318	98.809	6467.431	8.652	Low	2025-05-12T02:50:00Z	1
8	1546	67.606	96.383	6600.521	7.345	Moderate	2025-05-12T02:50:00Z	2
9	1578	76.149	94.297	7689.232	8.865	Moderate	2025-05-12T02:50:00Z	2
10	1587	61.053	98.321	7044.754	4.408	Highly_Active	2025-05-12T02:50:00Z	3
11	1585	68.179	96.43	7639.963	5.814	Highly_Active	2025-05-12T02:50:00Z	3
12	1541	66.616	92.109	6952.252	9.738	Low	2025-05-12T02:50:00Z	1
13	1500	61.56	91.495	6276.744	8.114	Moderate	2025-05-12T02:50:00Z	2

รูปที่ 14 แสดงข้อมูล **Activity Level**

Event Bridge Trigger + Lambda

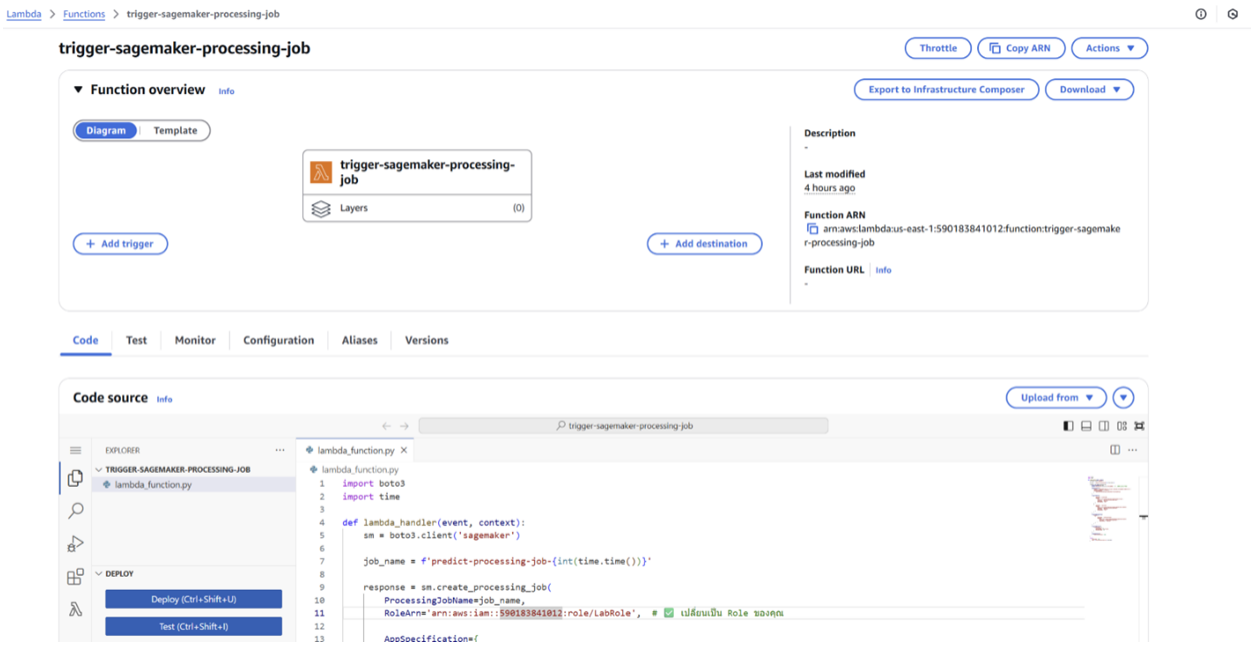
วัตถุประสงค์ของขั้นตอนนี้

ในกระบวนการประมวลผลข้อมูลระดับ production นั้น ความสามารถในการทำงานโดยอัตโนมัติ (automation) มีความสำคัญอย่างยิ่ง โดยเฉพาะในระบบที่ต้องประมวลผลข้อมูล streaming หรือข้อมูลใหม่ที่ถูก ingest อย่างสม่ำเสมอ



รูปที่ 15 แสดงความถี่ของ **Lambda**

เพื่อให้ pipeline สามารถทำงาน โดยไม่ต้องพึ่งพาการ run แบบ manual ทุกครั้งที่มีข้อมูลใหม่ ระบบจึงได้ออกแบบ automation layer ด้วยการใช้ Amazon EventBridge ร่วมกับ AWS Lambda เพื่อเป็นตัวกระตุ้นให้เกิดการประมวลผล downstream อย่างอัตโนมัติ



รูปที่ 16 แสดงการ **Trigger ของ Sage marker**

AWS Sagemarker

การสร้าง SageMaker Processing Job เพื่อประมวลผลข้อมูลสำหรับการทำนาย (Prediction)

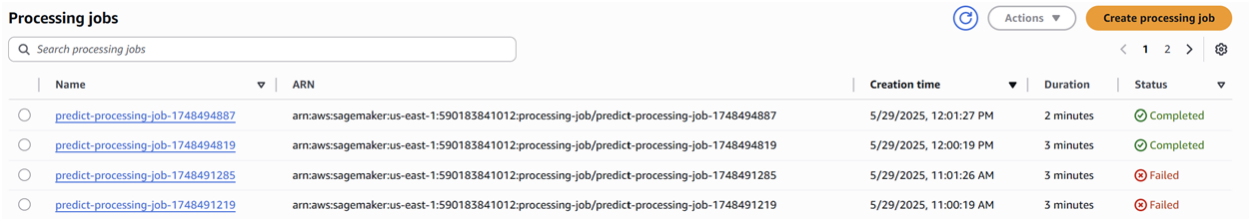
วัตถุประสงค์

- การสร้าง **SageMaker Processing Job** ในโครงการนี้ มีวัตถุประสงค์หลักเพื่อ:

- ประมวลผลข้อมูลที่ได้จาก **Glue Job** และรวมเป็นไฟล์ .csv ที่พร้อมสำหรับนำเข้าสู่โมเดล Machine Learning

- ทำ **inference** (การทำนาย) ด้วยโมเดล XGBoost จาก endpoint ที่ deploy ไว้ผ่าน **SageMaker Endpoint โดยการสร้าง script python เพื่อเรียกใช้งานเวลาได้รับ trigger จาก lambda.**

- บันทึกผลลัพธ์ที่ทำนายได้ เช่น ระดับความเครียด (Stress Level) ไว้ในรูปแบบ .csv สำหรับนำไปใช้งาน downstream (เช่น Visualize, Alert, Dashboard)



รูปที่ 17 แสดง **Prediction** *ไฟล์*

ข้อมูลการทำนายที่ได้

Activity Level	Sleep Duration	Step Count	UserId	blood oxygen level	bpm	timestamp	Activity_Level_Num	predicted_stress_level	activity_level_desc
Low	8.201	6498.01	1590	98.204	68.523	14/05/2025	1	2	Low Active
Low	1.298	7217.917	1584	93.213	70.723	14/05/2025	1	3	Low Active
Highly_Active	3.673	6895.089	1587	96.398	76.68	14/05/2025	3	2	Low Active
Highly_Active	2.932	7794.795	1562	92.174	64.476	14/05/2025	3	3	Somewhat Active
Highly_Active	2.364	6015.448	1575	93.796	73.728	14/05/2025	3	3	Low Active
Low	7.504	7754.494	1503	92.912	79.976	14/05/2025	1	3	Somewhat Active
Moderate	3.132	7258.425	1591	93.731	63.747	14/05/2025	2	3	Low Active
Highly_Active	7.95	7495.207	1501	91.498	78.906	14/05/2025	3	3	Low Active
Moderate	5.864	7660.623	1543	90.498	79.714	14/05/2025	2	3	Somewhat Active
Highly_Active	1.887	6790.285	1509	96.775	75.077	14/05/2025	3	2	Low Active
Moderate	1.944	7617.552	1533	90.4	75.842	14/05/2025	2	3	Somewhat Active
Moderate	4.249	6320.523	1517	91.238	67.059	14/05/2025	2	3	Low Active
Low	1.834	6739.385	1591	95.022	61.298	14/05/2025	1	3	Low Active
Low	5.314	6406.994	1539	94.569	64.282	14/05/2025	1	3	Low Active

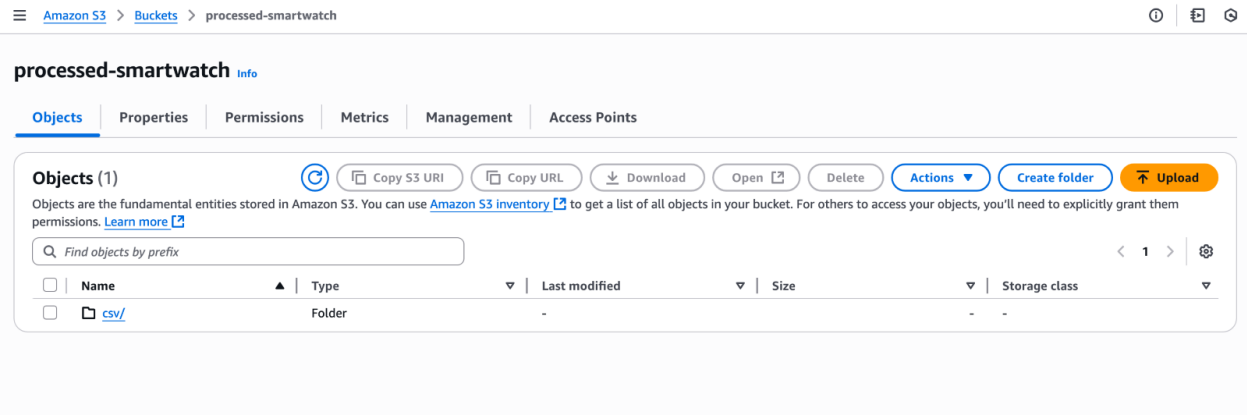
รูปที่ 18 แสดงการทำนายระดับความเครียด

การเชื่อมต่อข้อมูลจาก Amazon S3 สู่การแสดงผลผ่าน AWS QuickSight

ผลลัพธ์ที่ได้จาก SageMaker ถูกนำไปแสดงผลผ่าน Amazon QuickSight ซึ่งเชื่อมต่อกับข้อมูลที่จัดเก็บใน S3 ผ่าน Amazon Athena โดยมีการสร้างตารางแบบ External Table บน Athena เพื่อดึงข้อมูลจากไฟล์ .csv ที่ได้จากขั้นตอนก่อนหน้า โดยเลือกใช้รูปแบบการเชื่อมต่อแบบ Import to SPICE เพื่อเพิ่มประสิทธิภาพในการวิเคราะห์ข้อมูลแบบ interactive ดังรายละเอียดต่อไปนี้

1. การจัดเก็บข้อมูลบน Amazon S3

ข้อมูลทั้งหมดถูกจัดเก็บในรูปแบบไฟล์ CSV ภายใต้ bucket processed-smartwatch /csv/ ซึ่งประกอบด้วยข้อมูลด้านสุขภาพ เช่น อัตราการเต้นของหัวใจ (bpm), ระยะเวลาการนอนหลับ, จำนวนก้าวเดิน, ระดับกิจกรรม, และระดับความเครียดที่คาดการณ์ไว้จากโมเดล



รูปที่ 19 แสดง **S3** *ในการเก็บข้อมูลหลังจาก* **Prediction** *แล้ว*

2. การสร้างฐานข้อมูลและตารางใน Athena

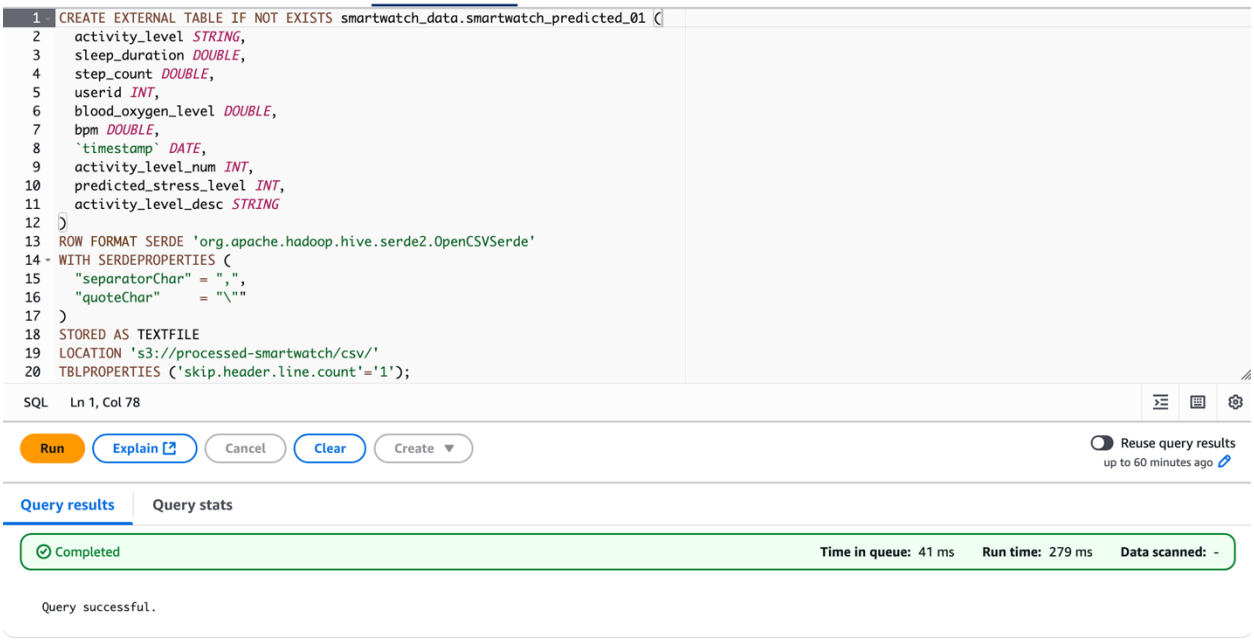
เพื่อให้ AWS QuickSight สามารถเข้าถึงและแสดงผลข้อมูลที่จัดเก็บอยู่ใน S3 ได้ จึงมีการสร้างฐานข้อมูลบน Amazon Athena ด้วยคำสั่ง SQL ดังนี้

```
CREATE DATABASE IF NOT EXISTS smartwatch_data;
```

จากนั้นได้มีการสร้างตารางภายนอก (External Table) เพื่อเชื่อมต่อกับไฟล์ใน S3 โดยไม่ใช้ Glue Crawler โดยมีคำสั่ง SQL ดังต่อไปนี้

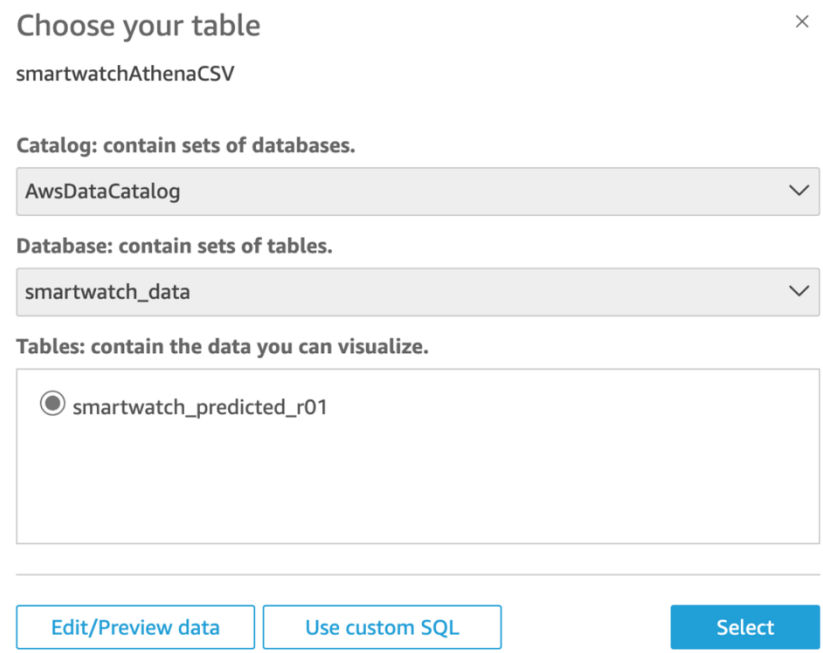
```
CREATE EXTERNAL TABLE IF NOT EXISTS smartwatch_data.smartwatch_predicted_r01 (
  activity_level STRING,
```

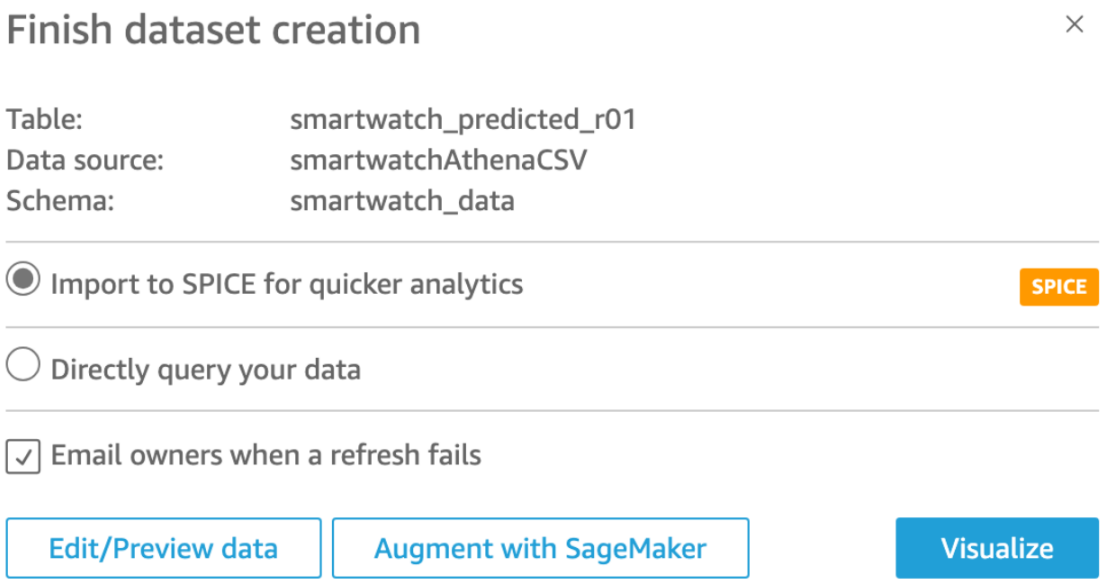
```
  sleep_duration DOUBLE,
  step_count DOUBLE,
  userid INT,
  blood_oxygen_level DOUBLE,
  bpm DOUBLE,
  `timestamp` DATE,
  activity_level_num INT,
  predicted_stress_level INT,
  activity_level_desc STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = ",",
  "quoteChar"     = "\""
)
STORED AS TEXTFILE
LOCATION 's3://processed-smartwatch /csv/'
TBLPROPERTIES ('skip.header.line.count'='1');
```



3. การนำเข้าข้อมูลสู่ QuickSight

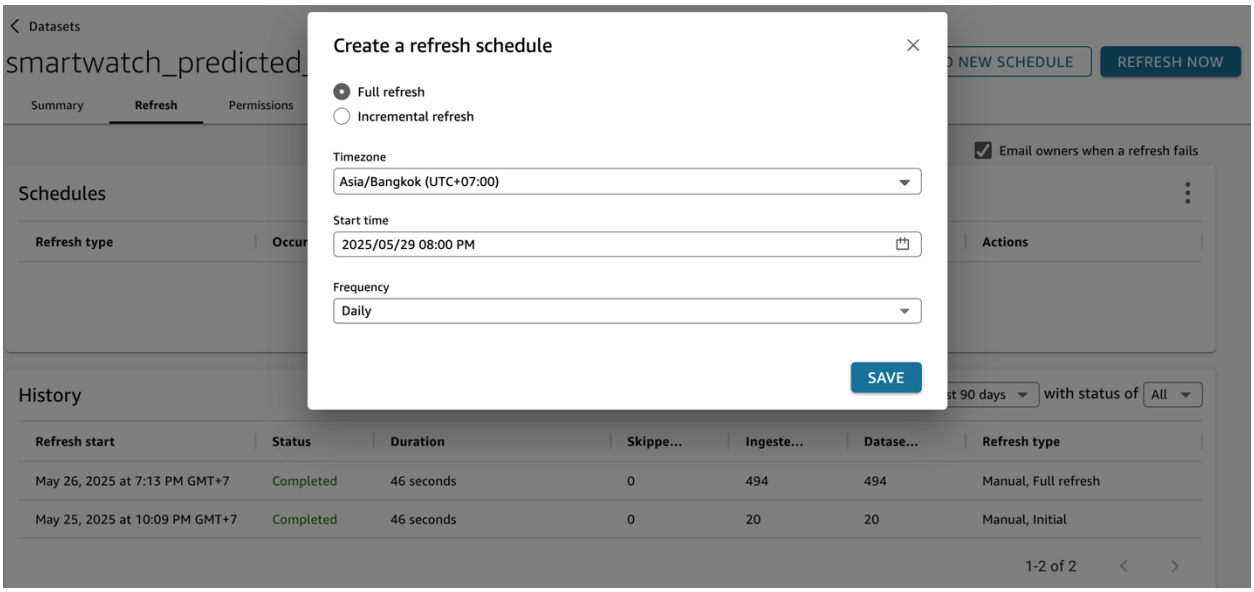
ในขั้นตอนนี้ ได้มีการสร้าง Data source ใน AWS QuickSight โดยเชื่อมต่อกับ Athena ผ่านฐานข้อมูล smartwatch_data และตาราง smartwatch_predicted_01 ซึ่งได้เลือกใช้วิธี Import to SPICE เพื่อให้สามารถตั้งเวลาอัปเดตข้อมูลแบบอัตโนมัติได้ในแต่ละวัน





4. การตั้งเวลา Refresh ข้อมูล

เพื่อให้ข้อมูลในแดชบอร์ดมีความทันสมัยอยู่เสมอ จึงมีการตั้งเวลาอัปเดต SPICE Dataset ให้ทำงานอัตโนมัติทุกวันในช่วงเวลาที่กำหนด (เช่น 01:00 UTC) โดย QuickSight จะดึงข้อมูลใหม่จาก S3 ผ่าน Athena และโหลดเข้าสู่ SPICE โดยอัตโนมัติ



5. การแสดงผลข้อมูลด้วยแดชบอร์ด

ในการนำเสนอข้อมูลระดับความเครียดของผู้ใช้ผ่านแดชบอร์ดในระบบ Amazon QuickSight ได้มีการใช้รหัสสี (Color Coding) เพื่อสื่อสารระดับความเครียดอย่างชัดเจน และสอดคล้องกับการตีความในเชิงคลินิก โดยสีแต่ละระดับมีความหมายดังนี้

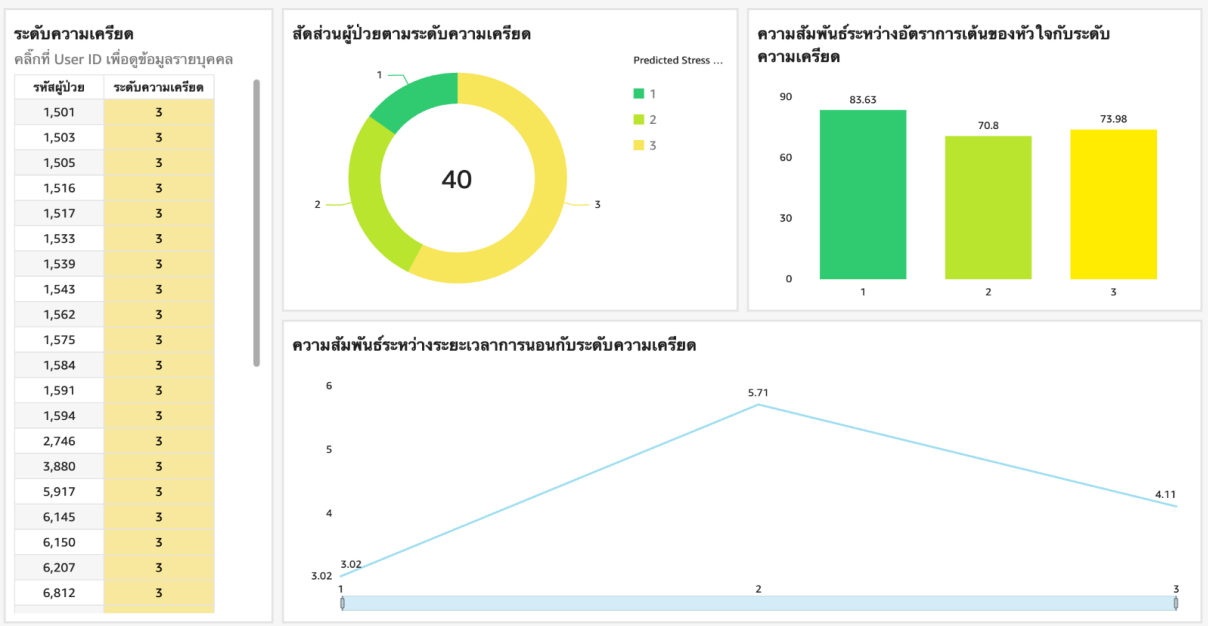
- **สีเขียวเข้ม** หมายถึง **ไม่มีความเครียด**
  กลุ่มนี้แสดงถึงผู้ใช้ที่ไม่มีอาการหรือภาวะความเครียดใด ๆ ปรากฏในการพยากรณ์
- **สีเขียวอ่อน** หมายถึง **ความเครียดระดับ 2 (ต่ำ)**
  สื่อถึงกลุ่มที่มีภาวะความเครียดในระดับน้อย หรืออยู่ในขอบเขตที่สามารถจัดการได้ด้วยตนเอง
- **สีเหลือง** หมายถึง **ความเครียดระดับ 3 (ปานกลาง)**
  กลุ่มนี้เริ่มมีภาวะความเครียดที่อาจส่งผลต่อพฤติกรรมหรือสภาพจิตใจ
  และควรได้รับการติดตามอย่างใกล้ชิด
- **สีส้ม** หมายถึง **ความเครียดระดับ 4 (สูง)**
  สะท้อนถึงระดับความเครียดที่มีความรุนแรง ซึ่งอาจเชื่อมโยงกับภาวะความเครียดเรื้อรัง
  หรือปัญหาสุขภาพจิตในระยะยาว
- **แดง** หมายถึง **ความเครียดระดับ 5 ขึ้นไป (วิกฤตหรือเสี่ยงสูงมาก)**
  กลุ่มนี้แสดงถึงผู้ใช้ที่อยู่ในระดับความเครียดที่รุนแรงที่สุด
  ซึ่งอาจมีผลกระทบทางร่างกายหรือจิตใจในระดับเฉียบพลัน เช่น ความวิตกกังวลรุนแรง
  ภาวะเครียดเรื้อรัง หรือสัญญาณของภาวะหมดไฟ (burnout)
  ผู้ใช้กลุ่มนี้ควรได้รับการประเมินอย่างละเอียด
  และดำเนินการแทรกแซงในระดับเร่งด่วนจากทีมผู้เชี่ยวชาญด้านสุขภาพจิต

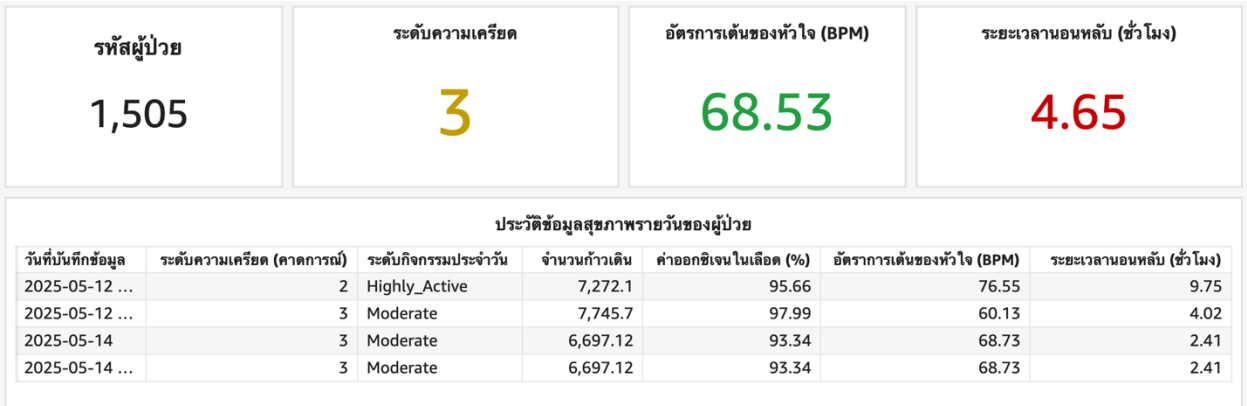
การใช้สีในลักษณะนี้มีวัตถุประสงค์เพื่อเพิ่มประสิทธิภาพในการสื่อสารข้อมูลเชิงภาพ ช่วยให้ผู้ใช้งานสามารถแยกแยะระดับของความเครียดจากผลการวิเคราะห์ของแบบจำลองได้อย่างชัดเจน รวดเร็ว และสนับสนุนการตัดสินใจเชิงสุขภาพในเชิงรุก

f. ผลลัพธ์ที่ได้ อภิปรายและสรุปผล พร้อมภาพแสดงตัวอย่าง Data Pipeline ที่ Validate และ Deploy แล้ว

จากการออกแบบและดำเนินการระบบ Data Pipeline พบว่า ระบบสามารถทำงานได้อย่างอัตโนมัติ ตั้งแต่ขั้นตอนการนำเข้าข้อมูลจาก Amazon S3 การจัดการข้อมูลผ่าน Athena ไปจนถึงการแสดงผลในรูปแบบแดชบอร์ดด้วย Amazon QuickSight ได้อย่างมีประสิทธิภาพ แดชบอร์ดที่สร้างขึ้นสามารถแสดงผลข้อมูลด้านสุขภาพของผู้ใช้รายวัน เช่น ระดับความเครียด

อัตราการเต้นของหัวใจ จำนวนก้าวเดิน และระยะเวลาการนอน ได้อย่างชัดเจน
โดยไม่ต้องจัดการข้อมูลด้วยตนเองทุกครั้งที่มีการอัปเดตข้อมูลใหม่ และการใช้รหัสสีในแดชบอร์ด (เช่น เขียว–เหลือง–แดง) ช่วยให้การสื่อสารข้อมูลสุขภาพ เช่น ระดับความเครียดของผู้ใช้
สามารถเข้าใจได้ง่ายและตรงประเด็นสำหรับผู้ใช้งานหรือบุคลากรด้านสุขภาพ





จากภาพแสดงตัวอย่างแดชบอร์ดใน Amazon QuickSight ที่ได้รับข้อมูลจากระบบ Data Pipeline ซึ่งผ่านการพยากรณ์ระดับความเครียดและวิเคราะห์ข้อมูลสุขภาพรายบุคคลแล้ว โดยสามารถแสดงผลแบบ Interactive Dashboard เพื่อให้ผู้ดูแลสุขภาพสามารถเข้าถึงข้อมูลได้ง่ายและรวดเร็ว

g. อภิปรายสิ่งที่ได้เรียนรู้และแนวทางในการพัฒนาต่อยอด

แนวทางการพัฒนาระบบ Data Pipeline บน AWSสำหรับการวิเคราะห์และพยากรณ์ระดับความเครียดจากข้อมูลสมาร์ตวอทช์มีข้อดีหลายด้าน

โดยเฉพาะในเรื่องความสามารถในการจัดการข้อมูลขนาดใหญ่ (Big Data) ได้อย่างมีประสิทธิภาพ และการแยกสถาปัตยกรรมออกเป็นขั้นตอนที่ชัดเจน เช่น การ ingestion, การทำความสะอาดข้อมูล (ETL), การวิเคราะห์ และการแสดงผลผ่านแดชบอร์ด ทำให้สามารถดูแล ปรับปรุง หรือขยายระบบได้ง่ายในอนาคต อีกทั้งการใช้บริการที่มีอยู่ใน AWS เช่น Glue, Lambda, SageMaker และ QuickSight ยังช่วยให้สามารถพัฒนาได้อย่างรวดเร็ว และประหยัดเวลาในการจัดการโครงสร้างพื้นฐานเอง

อย่างไรก็ตาม ระบบยังมีข้อจำกัด เช่น ความซับซ้อนในการจัดการสิทธิ์การเข้าถึงข้อมูล และความรู้เชิงเทคนิคที่จำเป็นในการใช้งาน AWS อย่างถูกต้อง นอกจากนี้ ข้อมูลที่ใช้ในโครงการยังเป็นข้อมูลจำลองที่มีคุณภาพไม่สมบูรณ์นักทำให้ความแม่นยำของโมเดลอาจยังไม่สามารถนำไปใช้งานจริงได้ทันที หากจะนำไปใช้จริงจึงต้องมีการเชื่อมต่อกับข้อมูลสุขภาพจากอุปกรณ์จริงและปรับปรุงคุณภาพข้อมูลให้ดียิ่งขึ้น

แนวทางการพัฒนาต่อยอดที่เป็นไปได้ ได้แก่ การเพิ่มความสามารถให้ระบบรองรับการเชื่อมต่อกับข้อมูลแบบเรียลไทม์จาก Smartwatch จริง การพัฒนาโมเดล Machine Learning ให้มีความแม่นยำมากขึ้นผ่านการเพิ่มฟีเจอร์หรือการใช้โมเดลที่ซับซ้อนขึ้น ตลอดจนการพัฒนาแอปพลิเคชันเพื่อแสดงผลและให้คำแนะนำผู้ใช้งานได้แบบส่วนบุคคล รวมถึงการปรับระบบให้รองรับการใช้งานในระดับองค์กร เช่น โรงพยาบาล หรือระบบสุขภาพที่เน้นการดูแลเชิงป้องกัน

จากการพัฒนาระบบวิเคราะห์และพยากรณ์ระดับความเครียดจากข้อมูลสมาร์ตวอทช์บน AWS ผู้จัดทำได้เรียนรู้กระบวนการสร้าง Data Pipeline อย่างครบถ้วน ตั้งแต่การจัดการกับข้อมูลดิบ (Uncleaned Data) ไปจนถึงการแสดงผลแบบ Interactive Dashboard โดยใช้เครื่องมือต่าง ๆ บน AWS ที่มีความยืดหยุ่นและสามารถออกแบบให้ทำงานแบบอัตโนมัติ (Automation) ได้ เช่น S3, Firehose, Cloud9, Glue, Lambda, Athena, SageMaker และ QuickSight ทำให้เข้าใจถึงขั้นตอนที่จำเป็นในการจัดการข้อมูลขนาดใหญ่ (Big Data) ทั้งด้านการ ingestion, ETL, การฝึกโมเดล Machine Learning และการนำเสนอผล

สิ่งที่สำคัญที่ได้เรียนรู้คือ ความสำคัญของคุณภาพข้อมูลเบื้องต้น (Data Quality) ต่อผลลัพธ์ของโมเดล ML และการวิเคราะห์ เนื่องจากข้อมูลจาก Smartwatch ที่ใช้ในการฝึกโมเดลมีทั้งค่าที่หายไป ค่าผิดปกติ และค่าที่ไม่สอดคล้องกัน จึงต้องมีการทำ Data Cleansing อย่างเข้มงวด เช่น การกรองค่า outlier การแปลง activity level ให้ตรงตามมาตรฐาน WHO และการเติมค่า missing อย่างเหมาะสม ซึ่งล้วนส่งผลต่อความแม่นยำของโมเดลพยากรณ์

นอกจากนี้ ยังได้ฝึกออกแบบการเชื่อมต่อระหว่างบริการต่าง ๆ บน AWS ให้ทำงานร่วมกันอย่างต่อเนื่อง เช่น การตั้งเวลา Trigger ด้วย EventBridge เพื่อเรียกใช้งาน Glue และ SageMaker โดยอัตโนมัติ และการตั้งค่า QuickSight ให้ Refresh ข้อมูลใหม่ในทุกวัน เพื่อให้ระบบสามารถนำเสนอข้อมูลที่อัปเดตต่อเนื่อง

อย่างไรก็ตาม ระบบยังมีข้อจำกัด เช่น ความซับซ้อนในการจัดการสิทธิ์การเข้าถึงข้อมูล และความรู้เชิงเทคนิคที่จำเป็นในการใช้งาน AWS อย่างถูกต้อง นอกจากนี้ ข้อมูลที่ใช้ในโครงการยังเป็นข้อมูลจำลองที่มีคุณภาพไม่สมบูรณ์นักทำให้ความแม่นยำของโมเดลอาจยังไม่สามารถนำไปใช้งานจริงได้ทันที หากจะนำไปใช้จริงจึงต้องมีการเชื่อมต่อกับข้อมูลสุขภาพจากอุปกรณ์จริงและปรับปรุงคุณภาพข้อมูลให้ดียิ่งขึ้น

สำหรับแนวทางในการพัฒนาต่อยอดในอนาคต ได้แก่ การให้ระบบรองรับการรับข้อมูลจาก Smartwatch จริงแทนข้อมูลจำลอง เพื่อทดสอบการทำงานกับข้อมูลสดแบบ Real-Time ได้จริงมากขึ้น อีกทั้งยังสามารถพัฒนาโมเดล Machine Learning ให้มีความซับซ้อนมากขึ้น เช่น การใช้โมเดล Deep Learning หรือ Time Series Prediction ที่เหมาะสมกับข้อมูลสุขภาพรายวัน นอกจากนี้ยังสามารถนำระบบไปประยุกต์ใช้ในงานสุขภาพอื่น ๆ เช่น การติดตามภาวะเสี่ยงต่าง ๆ หรือการสร้างระบบแนะนำด้านสุขภาพเฉพาะบุคคล และอาจขยายผลไปสู่การใช้งานในระดับองค์กร เช่น โรงพยาบาล หรือบริษัทประกันสุขภาพ ที่ต้องการวิเคราะห์ความเสี่ยงของผู้ใช้ในมิติต่าง ๆ อย่างแม่นยำ