**Source Code ที่เกี่ยวข้อง**

**โยนไฟล์จากเครื่องเราเข้า Ubuntu**

* **Docker Compose**

scp -i kunakorn.pem docker-compose.yml [ubuntu@ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:/home/ubuntu](mailto:ubuntu@ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:/home/ubuntu)

* **Source 3 : Your design (relational database) (topic3)**

scp -i kunakorn.pem source3\_data.py [ubuntu@ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:/home/ubuntu](mailto:ubuntu@ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:/home/ubuntu)

* **Streamlit Dashboard (จริงๆอยู่ในขั้นตอนสุดท้าย แต่ผมเอามาสรุปทีเดียวครับ)**

scp -i kunakorn.pem dashboard.py ubuntu@ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:/home/ubuntu

**เข้าใช้งาน Ubuntu Server**

ssh -i kunakorn.pem [ubuntu@ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com](mailto:ubuntu@ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com)

**Install Docker บน Ubuntu**

sudo apt-get update

sudo apt-get install -y ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io

**Run Docker-Compose จากไฟล์ docker-compose.yml ที่โยนเข้าไป**

sudo docker-compose up -d

**install kafka บน ubuntu**

# Update the apt package list

sudo apt-get update

# Install Java (Kafka requires Java)

sudo apt-get install openjdk-11-jdk -y

# Download and install Kafka

wget https://archive.apache.org/dist/kafka/2.8.0/kafka\_2.13-2.8.0.tgz

tar -xvzf kafka\_2.13-2.8.0.tgz

cd kafka\_2.13-2.8.0

# Install Python3 and pip

sudo apt-get install python3 python3-pip -y

# Kafka Python client library

pip3 install kafka-python

#Run Code source3

python3 /home/ubuntu/source3\_data.py

**b. Kafka system สร้าง Topic (ตาม Requirement , สำหรับ Topic9-11 ผมวิเคราะห์ข้อมูลเพิ่มเติมครับ)**

sudo docker exec -it broker1 bash

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_1 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_2 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_3 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_4 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_5 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_6 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_7 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_8 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_9 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_10 --partitions 5 --replication-factor 3

kafka-topics --bootstrap-server broker1:29092 --create --topic topic\_11 --partitions 5 --replication-factor 3

**a. Data source 3 sources [1]**

**สร้าง Source1/Topic1 Datagen-pageview**

curl -X POST -H "Content-Type: application/json" \

-d '{

"name": "datagen-pageviews",

"config": {

"name": "datagen-pageviews",

"connector.class": "io.confluent.kafka.connect.datagen.DatagenConnector",

"key.converter": "org.apache.kafka.connect.storage.StringConverter",

"value.converter": "org.apache.kafka.connect.json.JsonConverter",

"value.converter.schemas.enable": "false",

"kafka.topic": "topic\_1",

"max.interval": "1000",

"quickstart": "pageviews",

"interval.type": "random",

"interval.range.min": "1",

"interval.range.max": "1000"

}

}' <http://localhost:8083/connectors>

**สร้าง Source2/Topic2 Datagen-users**

curl -X POST -H "Content-Type: application/json" \

-d '{

"name": "datagen-users",

"config": {

"name": "datagen-users",

"connector.class": "io.confluent.kafka.connect.datagen.DatagenConnector",

"key.converter": "org.apache.kafka.connect.storage.StringConverter",

"value.converter": "org.apache.kafka.connect.json.JsonConverter",

"value.converter.schemas.enable": "false",

"kafka.topic": "topic\_2",

"max.interval": "1000",

"quickstart": "users",

"interval.type": "random",

"interval.range.min": "1",

"interval.range.max": "1000"

}

}' <http://localhost:8083/connectors>

**สร้าง Source3/Topic3 จาก python file source3\_data จากไฟล์ที่โยนเข้าไปตั้งแต่แรก**

python3 /home/ubuntu/source3\_data.py

**c. ksqlDB operation [3]**

**#เข้าใช้งาน ksqldb ผ่าน ubuntu**

**sudo docker exec -it ksqldb-cli ksql http://ksqldb-server:8088**

**สร้าง Topic1 (stream)**

CREATE STREAM topic1\_stream (

userid VARCHAR,

pageid VARCHAR,

viewtime BIGINT

) WITH (

KAFKA\_TOPIC='topic\_1',

VALUE\_FORMAT='JSON'

);

**#ดูข้อมูล**

SELECT \* FROM topic1\_stream EMIT CHANGES;

+--------------------------------------+--------------------------------------+--------------------------------------+

|USERID |PAGEID |VIEWTIME |

+--------------------------------------+--------------------------------------+--------------------------------------+

|User\_5 |Page\_25 |90121 |

|User\_7 |Page\_91 |90131 |

|User\_1 |Page\_99 |90141 |

|User\_9 |Page\_32 |90151 |

|User\_3 |Page\_27 |90161 |

**สร้าง Topic 2 (stream)**

CREATE TABLE topic2\_stream (

userid VARCHAR PRIMARY KEY,

regionid VARCHAR,

gender VARCHAR,

registertime BIGINT

) WITH (

KAFKA\_TOPIC = 'topic\_2',

VALUE\_FORMAT = 'JSON'

);

**#ดูข้อมูล**

SELECT \* from topic2\_stream EMIT CHANGES;

+----------------------------+----------------------------+----------------------------+----------------------------+

|USERID |REGIONID |GENDER |REGISTERTIME |

+----------------------------+----------------------------+----------------------------+----------------------------+

|User\_5 |Region\_1 |FEMALE |1494777042498 |

|User\_1 |Region\_3 |FEMALE |1494573136138 |

|User\_5 |Region\_8 |OTHER |1514955962454 |

|User\_9 |Region\_6 |MALE |1489908326601 |

**สร้าง Topic3 (Table)**

CREATE TABLE topic3\_table (

Region\_id INT PRIMARY KEY,

Region\_name VARCHAR,

Population BIGINT,

Area\_size INT

) WITH (

KAFKA\_TOPIC='topic\_3',

VALUE\_FORMAT='JSON'

);

**#ตรวจสอบข้อมูลใน topic\_3 ออกจากหน้า ksqldb ก่อน แล้วรันโค้ดด้านล่าง**

sudo docker exec -it broker1 /bin/bash

kafka-console-consumer --bootstrap-server broker1:29092 --topic topic\_3 --from-beginning

**ถ้าข้อมูลไม่ขึ้น Run python อันนี้ใหม่ โดยออกจากหน้า ksqldb ก่อน  
#สร้าง Source3/Topic3 จาก python file source3\_data ที่โยนเข้าไปตั้งแต่แรก**

python3 /home/ubuntu/source3\_data.py

**i. Clean or transform data (topic4)**

**สร้าง Topic4 (Stream)**

CREATE STREAM users\_formatted WITH (

KAFKA\_TOPIC='topic\_4',

VALUE\_FORMAT='JSON'

) AS

SELECT

userid AS UserId,

regionid AS RegionId,

gender AS Gender,

TIMESTAMPTOSTRING(registertime, 'yyyy-MM-dd HH:mm:ss') AS RegisterTimeFormatted

FROM

topic2\_stream

EMIT CHANGES;

**#ดูข้อมูล**

SELECT \* FROM users\_formatted EMIT CHANGES;

+----------------------------+----------------------------+----------------------------+----------------------------+

|USERID |REGIONID |GENDER |REGISTERTIMEFORMATTED |

+----------------------------+----------------------------+----------------------------+----------------------------+

|User\_1 |Region\_3 |FEMALE |2017-05-12 07:12:16 |

|User\_5 |Region\_1 |FEMALE |2017-05-14 15:50:42 |

|User\_5 |Region\_8 |OTHER |2018-01-03 05:06:02 |

|User\_9 |Region\_6 |MALE |2017-03-19 07:25:26 |

**ii. Aggregation (join + group by) (topic5)**

**สร้าง Topic5 (Stream)**

CREATE STREAM Consolidate\_Stream WITH (

KAFKA\_TOPIC = 'topic\_5',

VALUE\_FORMAT = 'JSON'

) AS

SELECT

p.userid AS UserId,

p.pageid AS PageId,

u.RegionId AS RegionId,

u.Gender AS Gender,

r.Region\_id AS RegionIdInTable,

r.Region\_name AS RegionName,

r.Population AS Population,

r.Area\_size AS AreaSize,

p.viewtime AS ViewTime,

(p.viewtime \* 100.0 / r.Population) AS ViewPercentage

FROM topic1\_stream p

LEFT JOIN users\_formatted u ON p.userid = u.UserId

LEFT JOIN topic3\_table r ON CAST(SUBSTRING(u.RegionId, 8) AS INT) = r.Region\_id

EMIT CHANGES;

**#ดูข้อมูล**

SELECT \* FROM Consolidate\_Stream EMIT CHANGES;

A screen shot of a computer screen

Description automatically generated

**iii. Windows   
1. Tumbling (topic6)**

**สร้าง Topic6 (Table)**

CREATE TABLE PageViews\_Tumbling WITH (

KAFKA\_TOPIC = 'topic\_6',

VALUE\_FORMAT = 'JSON',

PARTITIONS = 5,

REPLICAS = 3

) AS

SELECT

RegionId,

LATEST\_BY\_OFFSET(RegionName) AS RegionName,

COUNT(\*) AS ViewCount,

WINDOWSTART AS StartWindow,

WINDOWEND AS EndWindow

FROM Consolidate\_Stream

WINDOW TUMBLING (SIZE 1 MINUTE)

GROUP BY RegionId

EMIT CHANGES;

**#ดูข้อมูล**

SELECT \* FROM PageViews\_Tumbling EMIT CHANGES;

A screen shot of a computer screen

Description automatically generated

**2. Hopping (topic7)**

**สร้าง Topic7 (Table)**

CREATE TABLE PageViews\_Hopping WITH (

KAFKA\_TOPIC = 'topic\_7',

KEY\_FORMAT = 'JSON',

VALUE\_FORMAT = 'JSON'

) AS

SELECT

RegionId,

RegionName,

COUNT(\*) AS PageViewCount,

WINDOWSTART AS StartPeriod,

WINDOWEND AS EndPeriod

FROM Consolidate\_Stream

WINDOW HOPPING (SIZE 5 SECONDS, ADVANCE BY 2 SECONDS)

GROUP BY RegionId, RegionName

EMIT CHANGES;

**#ดูข้อมูล**

SELECT \* FROM PageViews\_Hopping EMIT CHANGES;

A screen shot of a computer screen

Description automatically generated

**3. Session (topic8)**

**สร้าง Topic 8 (Table)**

CREATE TABLE Session\_Window\_Analysis WITH (

KAFKA\_TOPIC = 'topic\_8',

VALUE\_FORMAT = 'JSON',

PARTITIONS = 5,

REPLICAS = 3

) AS

SELECT

RegionId,

LATEST\_BY\_OFFSET(Gender) AS Gender,

COUNT(\*) AS PageVisitCount,

(WINDOWEND - WINDOWSTART) / 1000 AS SessionLengthSeconds,

WINDOWSTART AS SessionStart,

WINDOWEND AS SessionEnd

FROM Consolidate\_Stream

WINDOW SESSION (5 SECONDS)

GROUP BY RegionId

EMIT CHANGES;

**#ดูข้อมูล**

SELECT \* FROM Session\_Window\_Analysis EMIT CHANGES;

A screen shot of a computer screen

Description automatically generated

**#Topic10 (เพิ่มเติม)**

CREATE TABLE Session\_Window\_Analysis WITH (

KAFKA\_TOPIC = 'topic\_10',

VALUE\_FORMAT = 'JSON',

PARTITIONS = 5,

REPLICAS = 3

) AS

SELECT

RegionId,

LATEST\_BY\_OFFSET(Gender) AS Gender,

COUNT(\*) AS PageVisitCount,

(WINDOWEND - WINDOWSTART) / 1000 AS SessionLengthSeconds,

WINDOWSTART AS SessionStart,

WINDOWEND AS SessionEnd

FROM Consolidate\_Stream

WINDOW SESSION (5 SECONDS)

GROUP BY RegionId

EMIT CHANGES;

**#Topic11(เพิ่มเติม)**

CREATE TABLE Region\_Page\_Summary WITH (

KAFKA\_TOPIC = 'topic\_11',

VALUE\_FORMAT = 'JSON',

PARTITIONS = 5,

REPLICAS = 3

) AS

SELECT

RegionId,

LATEST\_BY\_OFFSET(RegionName) AS RegionName,

COUNT(\*) AS TotalPageVisits,

ROUND(AVG(ViewTime), 2) AS AverageViewTime

FROM Consolidate\_Stream

GROUP BY RegionId

EMIT CHANGES;

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**d. Apache Pinot [5,6]**

**สร้าง Schema และ Table สำหรับ Consolidate\_Stream (ข้อมูลจาก topic5)**

**#สร้าง Schema**

SCHEMA\_JSON='{

"schemaName": "Consolidate",

"enableColumnBasedNullHandling": false,

"dimensionFieldSpecs": [

{ "name": "USERID", "dataType": "STRING", "notNull": false },

{ "name": "REGIONID", "dataType": "STRING", "notNull": false },

{ "name": "GENDER", "dataType": "STRING", "notNull": false },

{ "name": "REGIONNAME", "dataType": "STRING", "notNull": false },

{ "name": "AREASIZE", "dataType": "INT", "notNull": false }

],

"metricFieldSpecs": [

{ "name": "VIEWPERCENTAGE", "dataType": "FLOAT", "notNull": false }

],

"dateTimeFieldSpecs": [

{ "name": "VIEWTIME", "dataType": "LONG", "notNull": false, "format": "1:MILLISECONDS:EPOCH", "granularity": "1:MILLISECONDS" }

]

}'

curl -X POST -H "Content-Type: application/json" -d "$SCHEMA\_JSON" http://localhost:9000/schemas

**#สร้าง table**

TABLE\_JSON='{

"tableName": "Consolidate\_REALTIME",

"tableType": "REALTIME",

"segmentsConfig": {

"schemaName": "Consolidate",

"replication": "1",

"timeColumnName": "VIEWTIME",

"replicasPerPartition": "1"

},

"tenants": {

"broker": "DefaultTenant",

"server": "DefaultTenant"

},

"tableIndexConfig": {

"streamConfigs": {

"streamType": "kafka",

"stream.kafka.topic.name": "topic\_5",

"stream.kafka.broker.list": "broker1:29092",

"stream.kafka.consumer.type": "lowlevel",

"stream.kafka.consumer.prop.auto.offset.reset": "smallest",

"stream.kafka.consumer.factory.class.name": "org.apache.pinot.plugin.stream.kafka20.KafkaConsumerFactory",

"stream.kafka.decoder.class.name": "org.apache.pinot.plugin.stream.kafka.KafkaJSONMessageDecoder"

}

},

"metadata": {}

}'

curl -X POST -H "Content-Type: application/json" -d "$TABLE\_JSON" http://localhost:9000/tables

**สร้าง Schema และ Table สำหรับ PageViews\_Tumbling (ข้อมูลจาก topic\_6)**

**# สร้าง Schema**

SCHEMA\_JSON='{

"schemaName": "PageViews\_Tumbling",

"enableColumnBasedNullHandling": false,

"dimensionFieldSpecs": [

{ "name": "REGIONNAME", "dataType": "STRING", "notNull": false }

],

"metricFieldSpecs": [

{ "name": "VIEWCOUNT", "dataType": "INT", "notNull": false }

],

"dateTimeFieldSpecs": [

{ "name": "STARTWINDOW", "dataType": "LONG", "notNull": false, "format": "1:MILLISECONDS:EPOCH", "granularity": "1:MILLISECONDS" },

{ "name": "ENDWINDOW", "dataType": "LONG", "notNull": false, "format": "1:MILLISECONDS:EPOCH", "granularity": "1:MILLISECONDS" }

]

}'

curl -X POST -H "Content-Type: application/json" -d "$SCHEMA\_JSON" http://localhost:9000/schemas

**# สร้าง Table**

TABLE\_JSON='{

"tableName": "PageViews\_Tumbling\_REALTIME",

"tableType": "REALTIME",

"segmentsConfig": {

"schemaName": "PageViews\_Tumbling",

"replication": "1",

"timeColumnName": "STARTWINDOW",

"replicasPerPartition": "1"

},

"tenants": {

"broker": "DefaultTenant",

"server": "DefaultTenant"

},

"tableIndexConfig": {

"streamConfigs": {

"streamType": "kafka",

"stream.kafka.topic.name": "topic\_6",

"stream.kafka.broker.list": "broker1:29092",

"stream.kafka.consumer.type": "lowlevel",

"stream.kafka.consumer.prop.auto.offset.reset": "smallest",

"stream.kafka.consumer.factory.class.name": "org.apache.pinot.plugin.stream.kafka20.KafkaConsumerFactory",

"stream.kafka.decoder.class.name": "org.apache.pinot.plugin.stream.kafka.KafkaJSONMessageDecoder"

}

},

"metadata": {}

}'

curl -X POST -H "Content-Type: application/json" -d "$TABLE\_JSON" http://localhost:9000/tables

**สร้าง Schema และ Table สำหรับ Session\_Window\_Analysis (ข้อมูลจาก topic8)**

**#สร้าง Schema**

SCHEMA\_JSON='{

"schemaName": "Session\_Window\_Analysis",

"enableColumnBasedNullHandling": false,

"dimensionFieldSpecs": [

{ "name": "GENDER", "dataType": "STRING", "notNull": false },

{ "name": "PAGEVISITCOUNT", "dataType": "INT", "notNull": false }

],

"metricFieldSpecs": [

{ "name": "SESSIONLENGTHSECONDS", "dataType": "LONG", "notNull": false }

],

"dateTimeFieldSpecs": [

{ "name": "SESSIONSTART", "dataType": "LONG", "notNull": false, "format": "1:MILLISECONDS:EPOCH", "granularity": "1:MILLISECONDS" }

]

}'

curl -X POST -H "Content-Type: application/json" -d "$SCHEMA\_JSON" http://localhost:9000/schemas

**#สร้าง Table**

TABLE\_JSON='{

"tableName": "Session\_Window\_Analysis\_REALTIME",

"tableType": "REALTIME",

"segmentsConfig": {

"schemaName": "Session\_Window\_Analysis",

"replication": "1",

"timeColumnName": "SESSIONSTART",

"replicasPerPartition": "1"

},

"tenants": {

"broker": "DefaultTenant",

"server": "DefaultTenant"

},

"tableIndexConfig": {

"streamConfigs": {

"streamType": "kafka",

"stream.kafka.topic.name": "topic\_8",

"stream.kafka.broker.list": "broker1:29092",

"stream.kafka.consumer.type": "lowlevel",

"stream.kafka.consumer.prop.auto.offset.reset": "smallest",

"stream.kafka.consumer.factory.class.name": "org.apache.pinot.plugin.stream.kafka20.KafkaConsumerFactory",

"stream.kafka.decoder.class.name": "org.apache.pinot.plugin.stream.kafka.KafkaJSONMessageDecoder"

}

},

"metadata": {}

}'

curl -X POST -H "Content-Type: application/json" -d "$TABLE\_JSON" http://localhost:9000/tables

**สร้าง Schema และ Table สำหรับ Region\_Page\_Summary (ข้อมูลจาก topic11)**

**#สร้าง Schema**

SCHEMA\_JSON='{

"schemaName": "Region\_Page\_Summary",

"enableColumnBasedNullHandling": false,

"dimensionFieldSpecs": [

{ "name": "REGIONNAME", "dataType": "STRING", "notNull": false }

],

"metricFieldSpecs": [

{ "name": "TOTALPAGEVISITS", "dataType": "INT", "notNull": false },

{ "name": "AVERAGEVIEWTIME", "dataType": "DOUBLE", "notNull": false }

],

"dateTimeFieldSpecs": [

{ "name": "TIMESTAMP", "dataType": "LONG", "notNull": false, "format": "1:MILLISECONDS:EPOCH", "granularity": "1:MILLISECONDS" }

]

}'

curl -X POST -H "Content-Type: application/json" -d "$SCHEMA\_JSON" http://localhost:9000/schemas

**#สร้าง Table**

TABLE\_JSON='{

"tableName": "Region\_Page\_Summary\_REALTIME",

"tableType": "REALTIME",

"segmentsConfig": {

"schemaName": "Region\_Page\_Summary",

"replication": "1",

"timeColumnName": "TIMESTAMP",

"replicasPerPartition": "1"

},

"tenants": {

"broker": "DefaultTenant",

"server": "DefaultTenant"

},

"tableIndexConfig": {

"streamConfigs": {

"streamType": "kafka",

"stream.kafka.topic.name": "topic\_11",

"stream.kafka.broker.list": "broker1:29092",

"stream.kafka.consumer.type": "lowlevel",

"stream.kafka.consumer.prop.auto.offset.reset": "smallest",

"stream.kafka.consumer.factory.class.name": "org.apache.pinot.plugin.stream.kafka20.KafkaConsumerFactory",

"stream.kafka.decoder.class.name": "org.apache.pinot.plugin.stream.kafka.KafkaJSONMessageDecoder"

}

},

"metadata": {}

}'

curl -X POST -H "Content-Type: application/json" -d "$TABLE\_JSON" <http://localhost:9000/tables>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**e. Dashboard [7]**

**Dashboard ใช้ streamlit**

pip install streamlit pandas pinotdb plotly

~/.local/bin/streamlit run dashboard.py --server.port 8501 --server.enableCORS false

export PATH=$PATH:~/.local/bin

streamlit --version

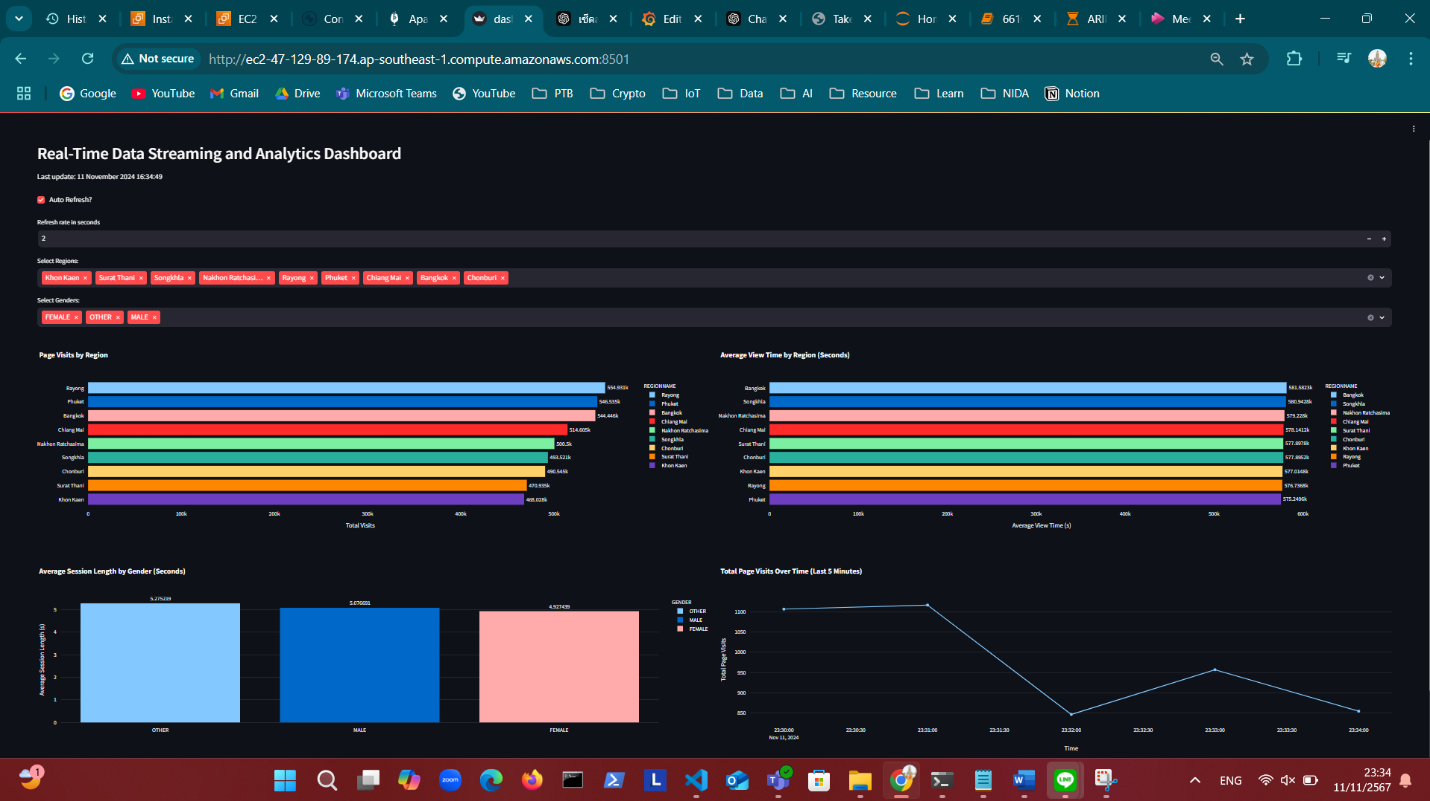
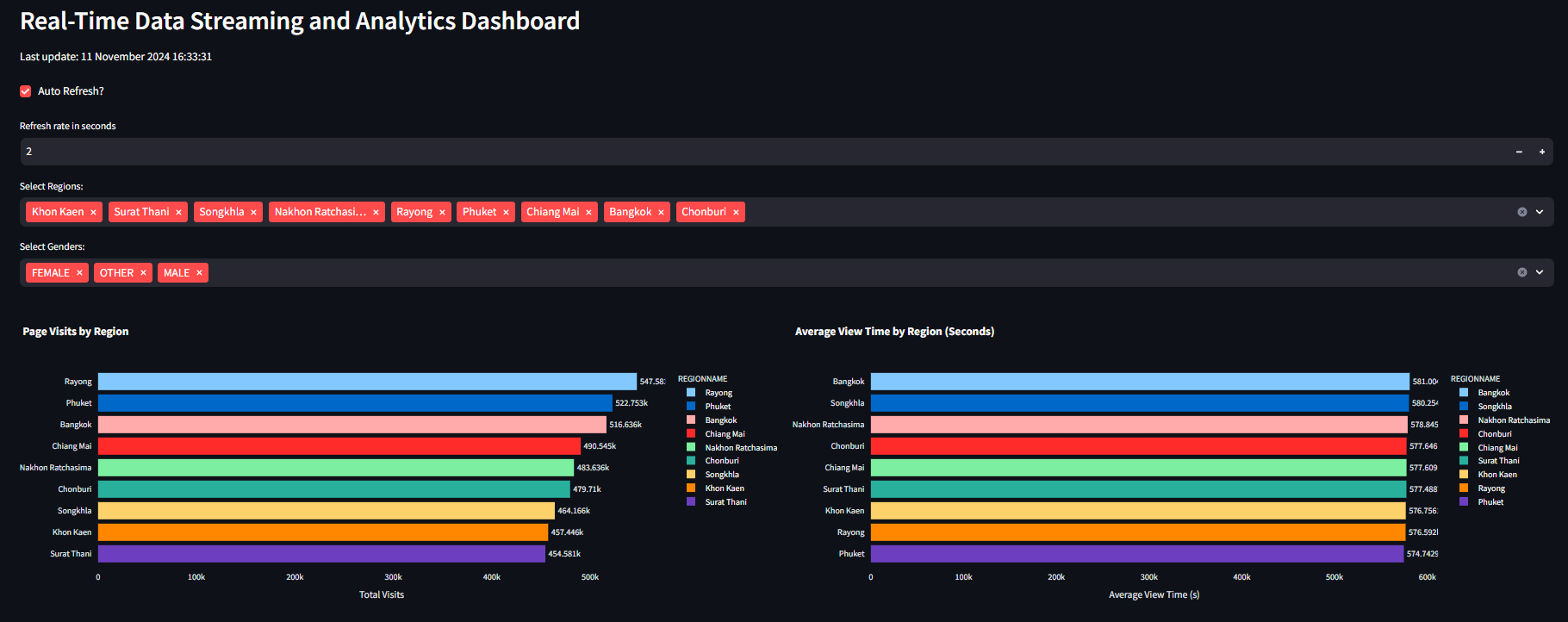
nohup streamlit run dashboard.py --server.port 8501 --server.enableCORS false > streamlit.log 2>&1 &

# Run Streamlit ที่เขียนใน python ไฟล์ (คำสั่งโยนไฟล์อยู่ด้านบนรวมกับโยนไฟล์อื่นๆครับ File : dashboard.py)

python3 -m streamlit run dashboard.py --server.port 8501 --server.enableCORS false

**ดู Dashboard ผ่าน URL นี้**  
<http://ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:8501>

**ตัวอย่างหน้าตา Dashboard**



A screenshot of a computer

Description automatically generated

**Port ทั้งหมด**

* **Confluent**

<http://ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:9021>

* **Apache Pinot**

<http://ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:9000>

* **Streamlit real-time dashboard**

<http://ec2-47-129-89-174.ap-southeast-1.compute.amazonaws.com:8501>

**#ลบ Datagen (Topic1,2)**curl -X DELETE <http://localhost:8083/connectors/datagen-pageviews>

curl -X DELETE <http://localhost:8083/connectors/datagen-users>