

## Take-home Midterm Exam

### Requirement Data streaming and Realtime analytics System Requirement

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

##### i. Source 1 : PageView (stream datagen) (topic1)

###### 1\_pageviews

**General settings**

name	1_pageviews
partitions	5
min.insync.replicas	1
cleanup.policy	delete
retention.ms	604800000
max.message.bytes	1048588
retention.bytes	-1

**Messages**

- {"viewtime":216441,"userid":"User\_8","pageid":"Page\_62"}  
Partition: 3 Offset: 4252 Timestamp: 1731704783679
- {"viewtime":216421,"userid":"User\_3","pageid":"Page\_75"}  
Partition: 4 Offset: 4394 Timestamp: 1731704783200
- {"viewtime":216411,"userid":"User\_2","pageid":"Page\_65"}  
Partition: 3 Offset: 4251 Timestamp: 1731704782872

**Schema**

```

1 {
2   "viewtime": 217031,
3   "userid": "User_8",
4   "pageid": "Page_61"
5 }
  
```

What: เป็นการสร้าง pageviews มาจาก การ stream datagen ข้อมูลเข้ามาย่างต่อเนื่อง

Why: ใช้สำหรับจำลองพฤติกรรมของผู้ใช้งานบนเว็บไซต์ เช่น จำนวนการคลิก หน้าที่เข้าเยี่ยมชม ฯลฯ ซึ่งเป็นข้อมูลสำคัญสำหรับการวิเคราะห์พฤติกรรมผู้ใช้ในแบบเรียลไทม์

How: Input: เครื่องมือจำลองข้อมูล (datagen) จะส่งข้อมูลการเยี่ยมชมหน้าเว็บในรูปแบบสตรีมมายัง Kafka

Process: ข้อมูลถูกจัดเก็บใน Kafka และสามารถถูกนำ上来ใช้ในขั้นตอนการวิเคราะห์ต่อไป

Output: ข้อมูล pageview ที่พร้อมใช้งานใน real-time analytics

##### ii. Source 2 : Users\_ (stream datagen) (topic2)

###### 2\_users

**General settings**

name	2_users
partitions	5
min.insync.replicas	1
cleanup.policy	delete
retention.ms	604800000
max.message.bytes	1048588
retention.bytes	-1

**Messages**

- {"registertime":1491202144377,"userid":"User\_4","regionid":"Region\_9","gender":"OTHER"}  
Partition: 3 Offset: 9616 Timestamp: 1731704991883
- {"registertime":1492630070747,"userid":"User\_5","regionid":"Region\_1","gender":"MALE"}  
Partition: 2 Offset: 2414 Timestamp: 1731704990951
- {"registertime":1493306344909,"userid":"User\_6","regionid":"Region\_1","gender":"OTHER"}  
Partition: 0 Offset: 9810 Timestamp: 1731704990845

**Schema**

```

1 {
2   "registertime": 1497589055851,
3   "userid": "User_6",
4   "regionid": "Region_4",
5   "gender": "OTHER"
6 }
  
```

What: เป็นหัวข้อสำหรับจัดเก็บข้อมูลผู้ใช้งาน เช่น เพศ และข้อมูลโปรดิฟ์ล์ ที่สร้างจากเครื่องมือ datagen

Why: ข้อมูลผู้ใช้งานสำคัญสำหรับการวิเคราะห์ในเรื่อง demographic หรือการทำ personal recommendation

How: Input: ข้อมูลผู้ใช้งานที่สร้างขึ้นแบบสังเคราะห์

Process: เก็บและกระจายข้อมูลใน Kafka topics

Output: ข้อมูลผู้ใช้งานสำหรับการใช้งานใน ksqlDB และระบบอื่น

### iii. Source 3 : Your design (relational database) (topic3)

#### 3\_orders

The screenshot shows the Kafka topic configuration page for '3\_orders'. At the top, there are tabs for Configuration, Messages, and Schema. The Configuration tab is selected. Below it, the General settings section shows the following configuration:

- name: 3\_orders
- partitions: 5
- min.insync.replicas: 1
- cleanup.policy: delete
- retention.ms: 604800000
- max.message.bytes: 1048588
- retention.bytes: -1

Below the configuration, there are three sample messages shown in JSON format:

```

1 {
2   "schema": {
3     "type": "struct",
4     "fields": [
5       {
6         "type": "string",
7         "optional": false,
8         "field": "order_id"
9       },
10      {
11        "type": "string",
12        "optional": false,
13        "field": "users_id"
14      },
15      {
16        "type": "string",
17        "optional": false,
18        "field": "order_date"
19      },
20      {
21        "type": "float64",
22        "optional": false,
23        "field": "total_amount"
24      },
25      {
26        "type": "string",
27        "optional": false,
28        "field": "status"
29      },
30      {
31        "type": "string",
32        "optional": false,
33        "field": "product_name"
34      }
35    ],
36    "type": "string",
37    "optional": false,
38    "field": "product_name"
39  },
40  "payload": {
41    "order_id": "9e21e8bd-5c0d-4a95-8242-0c4c10f0ba98",
42    "users_id": "User_2",
43    "order_date": "2024-11-15T21:10:23.625956",
44    "total_amount": 95.35,
45    "status": "Pending",
46    "product_name": "dinto"
47  }
}

```

What: หัวข้อที่ออกแบบเพื่อเชื่อมโยงข้อมูลจากฐานข้อมูลเชิงรั้งพันธ์ที่ออกแบบเอง เป็นข้อมูลคำสั่งซื้อของ  
เครื่องสำอางที่จะมีการบันทึกถึง เลขคำสั่งซื้อ รหัสผู้สั่งซื้อ วันที่ทำการสั่งซื้อ ราคा สถานะจัดส่ง ชื่อสินค้าที่ทำการ  
สั่งซื้อ

Why: การนำฐานข้อมูลแบบ relational มาเข้ามาร่วมให้ระบบรองรับข้อมูลโครงสร้างแบบซับข้อมากขึ้น

How: Input: ข้อมูลที่ดึงจากฐานข้อมูล relational ผ่าน Kafka Connect

Process: แปลงและส่งข้อมูลเข้าสู่ Kafka topics

Output: ข้อมูลจาก relational database ที่พร้อมประมวลผล

## b. Kafka system [2]

What: ระบบจัดการ data streaming ที่รองรับการส่งข้อมูล (messaging) แบบกระจาย (distributed) และ มีส่วนประกอบ:

5 Partitions

3 Brokers

8 Topics

Schema Register

Kafka Connect

Why: Kafka ช่วยส่งข้อมูลจากแหล่งข้อมูลไปยังระบบอื่นอย่างมีประสิทธิภาพ รองรับความเร็วและการขยายตัวของข้อมูล

### i. 5 partitions

partitions 5

Partition คือหน่วยย่อยของข้อมูลในหัวข้อ (topic) แต่ละหัวข้อสามารถถูกแบ่งออกเป็นหลาย partitions แต่ละ partition จะถูกเก็บแยกกันใน broker และมี unique offset เพื่อรับลำดับของข้อมูลใน partition นั้น ๆ

### ii. 3 brokers

```
broker:  
  image: confluentinc/cp-kafka:7.7.1  
  hostname: broker  
  container_name: broker  
  ports:  
    - "29092:29092"  
    - "9092:9092"  
    - "9101:9101"
```

What: Broker คือส่วนสำคัญใน Kafka cluster ทำหน้าที่จัดเก็บ partitions ของหัวข้อ (topic) และประมวลผลการส่งข้อมูลระหว่าง producer และ consumer

Why: Kafka ช่วยส่งข้อมูลจากแหล่งข้อมูลไปยังระบบอื่นอย่างมีประสิทธิภาพ รองรับความเร็วและการขยายตัวของข้อมูล

### iii. 8 Topic

### iv. Schema Register

### v. Kafka connect

#### Topics

Topic name	Partitions
topic1	5
topic2	5
topic3	5
topic4	5
Topic4_trans	5
Topic5_agg	5
Topic6_TUMBLING	5
Topic7_HOPPING	5

## C. KsqlDB operation

What: เครื่องมือจัดการข้อมูลใน Kafka ผ่านคำสั่ง SQL ใช้สำหรับ clean, transform, aggregate และประมวลผลหน้าต่าง (window) ได้แก่ Tumbling, Hopping, และ Session

Why: ช่วยให้ผู้ใช้งานปรับปรุงข้อมูลและวิเคราะห์ข้อมูลแบบเรียลไทม์

How: Input: ข้อมูลที่มาจาก Kafka topics

Process: ใช้คำสั่ง SQL จัดการ เช่น: Clean หรือ transform ข้อมูล (topic4)

Aggregation เช่น join และ group by (topic5)

การใช้หน้าต่างสำหรับเวลาต่างๆ (topic6, topic7, topic8)

Output: ข้อมูลที่ประมวลผลแล้วส่งต่อไปยัง Apache Pinot

### ksqldb1



## i. Clean or transform data (topic4)

```
ksql> CREATE STREAM pageviews_with_minutes
>   WITH (KAFKA_TOPIC='T4_trans', PARTITIONS=5, VALUE_FORMAT='JSON') AS
>SELECT
>   (viewtime / 60000) AS viewtime_minutes,
>   userid,
>   pageid
>   FROM PAGEVIEWS_STREAM
>   EMIT CHANGES;

Message
-----
Created query with ID CSAS_PAGEVIEWS_WITH_MINUTES_19
```

```
ksql> select * from pageviews_with_minutes;
+-----+-----+-----+
|VIEWTIME_MINUTES|USERID|PAGEID|
+-----+-----+-----+
|3|User_7|Page_42
|3|User_1|Page_59
|3|User_3|Page_71
|3|User_5|Page_25
|3|User_5|Page_67
```

What: เป็นคำสั่งที่ใช้สร้าง stream pageviews\_with\_minutes ที่มีการกำหนดให้ข้อมูลใน stream ถูกจัดเก็บในหัวข้อ (topic) T4\_trans ใน Kafka และมีการคำนวณ viewtime\_minutes (เวลาในหน่วยนาที) โดยหารค่า viewtime (หน่วยมิลลิวินาที) ด้วย 60,000

Why: การแปลง viewtime เป็น viewtime\_minutes ทำให้การวิเคราะห์ข้อมูลในหน่วยนาทีง่ายขึ้น เช่น การนับจำนวนครั้งที่ผู้ใช้เยี่ยมชมเพจในแต่ละนาที

How: โดยการรับ input ตั้งทางมาจาก pageviews\_stream มาทำการแปลงค่าเวลาจากมิลลิวินาทีเป็นนาที

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

```
ksql> CREATE TABLE USER_PAGEVIEWS_SUMMARY
>   WITH (KAFKA_TOPIC='T5_agg', PARTITIONS=5, VALUE_FORMAT='JSON') AS
>SELECT
>   u.USERID AS U_USERID,
>   MIN(p.REGISTERTIME) AS REGISTER_DATE,
>   MAX(p.VIEWTIME_MINUTES) AS LAST_VIEW_DATE,
>   COUNT(p.PAGEID) AS TOTAL_PAGEVIEWS
>FROM USERS_STREAM u
>INNER JOIN PAGEVIEWS_with_Minutes p
>  WHERE p.WITHIN 1 HOUR ON u.USERID = p.USERID
>GROUP BY u.USERID;
Message
-----
Created query with ID CTAS_USER_PAGEVIEWS_SUMMARY_21
```

```
WARNING: DEPRECATION NOTICE: Stream-stream joins statements without a GRACE PERIOD will not be accepted in a future ksqlDB version.
Please use the GRACE PERIOD clause as specified in https://docs.ksqldb.io/en/latest/developer-guide/ksqldb-reference/select-push-query/
ksql> select * from USER_PAGEVIEWS_SUMMARY;
+-----+-----+-----+-----+
|U_USERID|REGISTER_DATE|LAST_VIEW_DATE|TOTAL_PAGEVIEWS|
+-----+-----+-----+-----+
|User_2|1488256054914|3|180
|User_3|1499028557594|3|130
|User_6|1497250000000|3|40
|User_8|148841273422|3|91
|User_5|1488416931703|3|135
|User_1|1489506360146|3|156
|User_4|1502458589628|3|8
|User_7|1487813526958|3|168
```

What: บันทึกข้อมูลสรุปของผู้ใช้แต่ละคนในช่วงเวลา 1 ชั่วโมง: วันที่ลงทะเบียน วันที่คูเพจล่าสุด จำนวนครั้งที่เข้าชมเพจ และเมื่อข้อมูลที่สร้างใน table มาจาก การ join และ group by ข้อมูลจากสอง streams คือ: USERS\_STREAM (ข้อมูลผู้ใช้งาน) กับ PAGEVIEWS\_with\_Minutes (ข้อมูลการเยี่ยมชมเพจที่แปลงเวลาเป็นนาทีแล้ว)

Why: รวมข้อมูลจากหลายแหล่ง (Join): การ join ข้อมูลผู้ใช้ (USERS\_STREAM) กับการเยี่ยมชมหน้าเว็บ (PAGEVIEWS\_with\_Minutes) ช่วยสร้างข้อมูลเชิงลึก เช่น ผู้ใช้คนใดเยี่ยมชมเว็บครั้งล่าสุดเมื่อไหร่

สรุปข้อมูล (Aggregation): การสรุปค่าทั้งหมด เช่น จำนวนการเยี่ยมชม (COUNT), เวลาล่าสุด (MAX), และวันที่ลงทะเบียน (MIN) ช่วยให้

How: รับ input จาก USERS\_STREAM และ PAGEVIEWS\_with\_Minutes ทำการ join data โดยมีการกำหนดช่วงเวลา 1 ชั่วโมง พั่วมทั้ง มีการคำนวณ aggregation : min หาค่าวันที่ลงทะเบียนที่ต่ำสุด(เร็วที่สุด) , max หาวันที่เยี่ยมชมล่าสุด , count นับจำนวนครั้งทั้งหมดที่ผู้ใช้เยี่ยมชมหน้าเว็บ จากนั้นทำการ group by เพื่อทำการแยกตามผู้ใช้แต่ละคน

### iii. Windows

#### 1. Tumbling(topic6)

```
ksql> CREATE TABLE pageviews_count
> WITH (KAFKA_TOPIC='T6_TUMBLING', PARTITIONS=5, VALUE_FORMAT='JSON') AS
>SELECT
>     USERID,
>     COUNT(PAGEID) AS pageview_count,
>     WINDOWSTART AS window_start,
>     WINDOWEND AS window_end
> FROM PAGEVIEWS_WITH_MINUTES
> WINDOW TUMBLING (SIZE 30 MINUTES)
> GROUP BY USERID
> EMIT CHANGES;

Message
Created query with ID CTAS_PAGEVIEWS_COUNT_23

ksql> select * from pageviews_count;
+-----+-----+-----+-----+-----+
|USERID |WINDOWSTART |WINDOWEND |PAGEVIEW_COUNT |WINDOW_START |WINDOW_END |
+-----+-----+-----+-----+-----+
|User_2 |1731699000000 |1731700800000 |4 |1731699000000 |1731700800000 |
|User_3 |1731699000000 |1731700800000 |4 |1731699000000 |1731700800000 |
|User_6 |1731699000000 |1731700800000 |4 |1731699000000 |1731700800000 |
|User_8 |1731699000000 |1731700800000 |4 |1731699000000 |1731700800000 |
|User_5 |1731699000000 |1731700800000 |5 |1731699000000 |1731700800000 |
|User_1 |1731699000000 |1731700800000 |5 |1731699000000 |1731700800000 |
|User_4 |1731699000000 |1731700800000 |7 |1731699000000 |1731700800000 |
|User_7 |1731699000000 |1731700800000 |6 |1731699000000 |1731700800000 |
|User_9 |1731699000000 |1731700800000 |4 |1731699000000 |1731700800000 |
+-----+-----+-----+-----+-----+
```

What: เป็นการคำนวณจำนวนการเข้าชม (pageviews) ของผู้ใช้งานในแต่ละช่วงเวลา (30 นาที)

Why: วิเคราะห์ข้อมูลแบบเวลาจำกัด การแบ่งข้อมูลออกเป็นหน้าต่างเวลาแบบ 30 นาทีช่วยให้สามารถติดตามพฤติกรรมของผู้ใช้ในช่วงเวลาที่กำหนดได้อย่างชัดเจน เช่น การดูว่าผู้ใช้ active ช่วงเวลาใด

#### 2. Hopping(topic7)

```
ksql> CREATE TABLE user_activity
> WITH (KAFKA_TOPIC='T7_HOPPING', PARTITIONS=5, VALUE_FORMAT='JSON') AS
>SELECT
>     USERID AS key,
>     AS_VALUE(USERID) AS id,
>     COUNT(*) AS user_count,
>     SUM(CASE WHEN REGIONID IS NOT NULL THEN 1 ELSE 0 END) AS region_count,
>     SUM(CASE WHEN GENDER IS NOT NULL THEN 1 ELSE 0 END) AS gender_count
> FROM USERS_STREAM
> WINDOW HOPPING (SIZE 10 MINUTES, ADVANCE BY 5 MINUTES)
> GROUP BY USERID;

Message
Created query with ID CTAS_USER_ACTIVITY_25

ksql> select * from user_activity;
+-----+-----+-----+-----+-----+-----+-----+
|KEY   |WINDOWSTART |WINDOWEND |ID    |USER_COUNT |REGION_COUNT |GENDER_COUNT |
+-----+-----+-----+-----+-----+-----+
|User_2 |1731699000000 |1731699600000 |User_2 |7 |7 |7 |
|User_2 |1731699380000 |1731699980000 |User_2 |7 |7 |7 |
|User_3 |1731699000000 |1731699600000 |User_3 |7 |7 |7 |
|User_3 |1731699380000 |1731699980000 |User_3 |7 |7 |7 |
|User_6 |1731699000000 |1731699600000 |User_6 |5 |5 |5 |
|User_6 |1731699380000 |1731699980000 |User_6 |5 |5 |5 |
|User_8 |1731699000000 |1731699600000 |User_8 |5 |5 |5 |
|User_8 |1731699380000 |1731699980000 |User_8 |5 |5 |5 |
|User_5 |1731699000000 |1731699600000 |User_5 |3 |3 |3 |
|User_5 |1731699380000 |1731699980000 |User_5 |3 |3 |3 |
|User_6 |1731699000000 |1731699600000 |User_6 |5 |5 |5 |
|User_1 |1731699000000 |1731699600000 |User_1 |7 |7 |7 |
|User_1 |1731699380000 |1731699980000 |User_1 |7 |7 |7 |
|User_4 |1731699000000 |1731699600000 |User_4 |4 |4 |4 |
+-----+-----+-----+-----+-----+-----+
```

What: เป็นการสรุปข้อมูลการกระทำของผู้ใช้งานในช่วงเวลาที่กำหนด โดยใช้การนับจำนวนการกระทำต่างๆ เช่น การมี REGIONID หรือ GENDER

Why: การใช้ Hopping Window ช่วยให้สามารถจับภาพข้อมูลที่ซ้อนทับในช่วงเวลาสั้น ๆ ได้ เช่น การนับจำนวนกิจกรรมของผู้ใช้ในช่วง 10 นาที แต่รีเฟรชทุก 5 นาที และ ด้วย Hopping Window คุณสามารถดูการเปลี่ยนแปลงของกิจกรรมในช่วงเวลาต่อเนื่อง

### 3. Session(topic8)

The screenshot shows a Kafka topic viewer interface. At the top, there is a code editor containing a CREATE TABLE statement for a table named 'pageviews\_session\_count'. The table is defined with columns for USERID, session\_start\_ts, session\_end\_ts, and pageview\_count, and includes a windowed session length column. Below the code editor are several input fields: 'auto.offset.reset' set to 'Latest', a dropdown menu, and buttons for 'Stop' and 'Run query'. The main area displays the results of the query, which is a JSON object representing a command response. The response includes a 'statementText' field with the CREATE TABLE statement, a 'status' field set to 'SUCCESS', and a 'message' field indicating the query was created. There are also fields for 'queryId', 'commandSequenceNumber', and 'warnings'. Below the results, there is a section titled '+ Produce a new message to this topic' and a list of two messages in the topic. The first message is expanded, showing its details: Partition: 3, Offset: 148243, and Timestamp: 1731729348127. The second message is collapsed.

```

1 | CREATE TABLE pageviews_session_count AS
2 |   SELECT USERID,
3 |     FORMAT_TIMESTAMP('yyyy-MM-dd HH:mm:ss', 'FROM_UNIXTIME(WINDOWSTART)'), 'yyyy-MM-dd HH:mm:ss', 'UTC') AS session_start_ts,
4 |     FORMAT_TIMESTAMP('yyyy-MM-dd HH:mm:ss', 'FROM_UNIXTIME(WINDOWEND)', 'yyyy-MM-dd HH:mm:ss', 'UTC') AS session_end_ts,
5 |     COUNT(*) AS pageview_count,
6 |     (WINDOWEND - WINDOWSTART) AS session_length_ms
7 |   FROM PAGEVIEW_EVENTS
8 |   WINDOW SESSION (5 MINUTES)
9 |   GROUP BY USERID
10|   EMIT CHANGES;

```

● Add query properties  
auto.offset.reset = Latest

+ Add another field  Stop  Run query

```

1 | {
2 |   "type": "createTopic",
3 |   "statementText": "CREATE TABLE PAGEVIEW_IS_SESSION_COUNT WITH (CLEANUP_POLICY='compact,delete', KAFKA_TOPIC='PAGEVIEW_IS_SESSION_COUNT', PARTITIONS=5, REPLICAS=1, RETENTION_MS=404800000) AS SELECT\n4 |     PAGEVIEW_IS_STREAM.USERID,\n5 |     FORMAT_TIMESTAMP('yyyy-MM-dd HH:mm:ss', 'FROM_UNIXTIME(WINDOWEND)'), 'yyyy-MM-dd HH:mm:ss', 'UTC') SESSION_START_TS,\n6 |     FORMAT_TIMESTAMP('yyyy-MM-dd HH:mm:ss', 'FROM_UNIXTIME(WINDOWEND)', 'yyyy-MM-dd HH:mm:ss', 'UTC') SESSION_END_TS,\n7 |     COUNT(*) PAGEVIEW_COUNT,\n8 |     (WINDOWEND - WINDOWSTART) SESSION_LENGTH_MS\n9 |   FROM PAGEVIEW_IS_STREAM\n10|   WINDOW SESSION (5 MINUTES)\n11|   GROUP BY PAGEVIEW_IS_STREAM.USERID\n12|   EMIT CHANGES",
13| }
14|

```

+ Produce a new message to this topic

▼ {"PAGEVIEW\_COUNT":275,"SESSION\_START":1731728045864,"SESSION\_END":1731729348127}  
Partition: 3    Offset: 148243    Timestamp: 1731729348127

▼ Partition: 3    Offset: 148242    Timestamp: 1731729340126

What: เป็นการคำนวณจำนวนการเยี่ยมชม (pageview\_count) ของผู้ใช้ที่เกิดขึ้นภายในระยะเวลา 5 นาทีต่อเนื่อง โดยแบ่งกลุ่มตาม USERID

Why: การวิเคราะห์ข้อมูล session ช่วยให้เข้าใจพฤติกรรมการใช้งาน เช่น ผู้ใช้ใช้เวลาเฉลี่ยเท่าใดในแต่ละ session และ ช่วยลดข้อมูลดิบที่กระบวนการจัดกรวยโดยสรุปเป็น session พัฒนาระบุจำนวนหนึ้น เก็บที่เยี่ยมชม

How: คำสั่งนี้ช่วยสร้างข้อมูลที่มีการสรุปในเชิง session ซึ่งเป็นพื้นฐานสำคัญสำหรับการทำ real-time analytics โดยสามารถนำไปใช้ในระบบที่ต้องการติดตามพฤติกรรมผู้ใช้และสร้างการตอบสนองได้อย่างทันท่วงที

## D. Apache Pinot

What: ระบบจัดเก็บและวิเคราะห์ข้อมูลที่สามารถตอบคำถามเชิงวิเคราะห์ได้รวดเร็ว

Why: เป็นส่วนสำคัญสำหรับการแสดงผลข้อมูลเชิงลึก (insight) ใน real-time analytics

How: Input: ข้อมูลที่ผ่านการประมวลผลจาก ksqlDB

Process: รันคำสั่ง query คำสั่งสำหรับการวิเคราะห์ข้อมูล

Output: ผลลัพธ์ที่พร้อมแสดงบน dashboard

### i. At least 3 query.

#### 1. User Activity by Region: Query: Number of users in each region.

```
SELECT
    regionid,
    gender,
    COUNT(*) AS totalUsers
FROM 2_users
GROUP BY regionid, gender
ORDER BY totalUsers DESC;
```

The screenshot shows the Apache Pinot Query Console interface. The top navigation bar includes 'Home' and 'Query Console'. The cluster name is set to 'PinotCluster'. On the left sidebar, there are links for 'Cluster Manager', 'Query Console' (which is currently selected), 'Zookeeper Browser', and 'Swagger REST API'. The main area has three main sections: 'SQL EDITOR', 'QUERY RESPONSE STATS', and 'QUERY RESULT'.

**SQL EDITOR:**

```
1 SELECT
2   regionid,
3   gender,
4   COUNT(*) AS totalUsers
5 FROM 2_users
6 GROUP BY regionid, gender
7 ORDER BY totalUsers DESC;
```

**QUERY RESPONSE STATS:**

timeUsedMs	numDocsScanned	totalDocs	numServersQueried	numServersResponded	numSegments
3	64287	64287	1	1	5

**Buttons:** Tracing, Use Multi-Stage Engine, Timeout (Milliseconds), FORMAT (SQL), RUN QUERY, EXCEL, CSV, COPY, Show JSON format.

The screenshot shows the Pinot Query Console interface. On the left sidebar, there are links for Cluster Manager, Query Console (which is selected), Zookeeper Browser, and Swagger REST API. The main area has a 'TABLES' section with a search bar and a list of tables: 1\_pageviews, 2\_users, 3\_orders, Topic7\_HOPPING, and topic3. Below this is a summary table with columns: timeUsedMs, numDocsScanned, totalDocs, numServersQueried, numServersResponded, and numSegments. The values shown are 3, 64287, 64287, 1, 1, and 5 respectively. There are buttons for EXCEL, CSV, COPY, and a 'Show JSON format' checkbox. The right side shows a 'QUERY RESULT' table with columns: regionid, gender, and totalUsers. The data rows are: Region\_6 (FEMALE, 2458), Region\_9 (MALE, 2446), Region\_7 (FEMALE, 2445), Region\_8 (FEMALE, 2438), Region\_8 (MALE, 2419), Region\_5 (FEMALE, 2418), Region\_6 (MALE, 2415), and Region\_4 (FEMALE, 2414).

What: เป็นการสรุปข้อมูลเกี่ยวกับผู้ใช้งานโดยการนับจำนวนผู้ใช้งาน ตาม ภูมิภาค และ เพศ พัฒนาเพื่อการเรียนรู้ด้วยตัวเองจากการจำแนกผู้ใช้งานที่มากที่สุดไปหน้าอยู่ที่สุด

Why: เพื่อใช้ในการวิเคราะห์ข้อมูลประชากรว่าผู้ใช้งานส่วนใหญ่มาจากภูมิภาคไหน และเมืองใด และเป็นการนำข้อมูลมาใช้ในการวางแผนการตลาดเพื่อเจาะกลุ่มผู้ใช้ตัวอย่างเช่น ประเทศที่มีผู้ใช้งานเยอะมากที่สุด และยังทำให้เราสามารถวัดผลตามวัตถุประสงค์ที่ต้องการได้

## 2. Total Pageviews per User

```
SELECT
    userid,
    COUNT(*) AS totalPageviews
FROM 1_pageviews
GROUP BY userid
ORDER BY totalPageviews DESC;
```

The screenshot shows the Pinot Query Console interface. On the left sidebar, there are links for Cluster Manager, Query Console (selected), Zookeeper Browser, and Swagger REST API. The main area has a 'TABLES' section with a search bar and a list of tables: 1\_pageviews, 2\_users, 3\_orders, Topic7\_HOPPING, and topic3. In the center, the 'SQL EDITOR' pane contains the following SQL query:

```
1 SELECT
2     userid,
3     COUNT(*) AS totalPageviews
4 FROM 1_pageviews
5 GROUP BY userid
6 ORDER BY totalPageviews DESC;
```

Below the editor are checkboxes for Tracing, Use Multi-Stage Engine, and a timeout setting. There are also 'FORMAT SQL' and 'RUN QUERY' buttons. The 'QUERY RESPONSE STATS' table shows: timeUsedMs (4), numDocsScanned (65468), totalDocs (65468), numServersQueried (1), numServersResponded (1), and numSegments (5). At the bottom, there are buttons for EXCEL, CSV, COPY, and a 'Show JSON format' checkbox.

CLUSTER NAME  
PinotCluster

**TABLES**

1\_pageviews

2\_users

3\_orders

Topic7\_HOPPING

topic3

**QUERY RESULT**

userid	totalPageviews
User_3	7381
User_9	7303
User_6	7294
User_8	7292
User_5	7280
User_2	7257
User_1	7235
User_4	7219

What: เป็นการนับจำนวนการเข้าชมที่แต่ละผู้ใช้ทำ และ แสดงผลลัพธ์ตามจำนวนการเข้าชมจากมากไปน้อย

Why: ข้อมูลนี้สามารถนำมาใช้ในการวิเคราะห์พฤติกรรมของผู้ใช้งานหรือการตัดสินใจได้

### 3. Pageviews per User

```
SELECT pageid, COUNT(*) AS pageviews
FROM 1_pageviews
GROUP BY pageid
ORDER BY pageviews DESC;
```

CLUSTER NAME  
PinotCluster

**SQL EDITOR**

```
1 SELECT pageid, COUNT(*) AS pageviews
2 FROM 1_pageviews
3 GROUP BY pageid
4 ORDER BY pageviews DESC;
```

Tracing    Use Multi-Stage Engine   Timeout (Milliseconds)

**FORMAT** **SQL** **RUN QUERY**

**QUERY RESPONSE STATS**

timeUsedMs	numDocsScanned	totalDocs	numServersQueried	numServersResponded	numSegments
4	65661	65661	1	1	5

**EXCEL** **CSV** **COPY**

**Show JSON format**

**QUERY RESULT**

The screenshot shows the Pinot Query Console interface. On the left, there's a sidebar with links: Cluster Manager, Query Console (which is selected and highlighted in blue), Zookeeper Browser, and Swagger REST API. The main area has a header with 'CLUSTER NAME PinotCluster' and navigation buttons for '4', '65661', '65661', '1', '1', and '5'. Below this is a search bar and buttons for 'EXCEL', 'CSV', and 'COPY'. A checkbox labeled 'Show JSON format' is also present. The central part of the screen is titled 'QUERY RESULT' and contains a table with the following data:

pageid	pageviews
Page_34	808
Page_21	790
Page_36	784
Page_15	783
Page_72	773
Page_67	770
Page_13	767
Page_62	765

**What:** เป็นการนับจำนวนการเข้าชมที่แต่ละหน้าเว็บได้รับและทำการแสดงผลลัพธ์เรียงจากหน้าที่มีการเข้าชมมากที่สุดไปหน้าอยู่ที่สุด เพื่อเป็นการดูว่าเพจไหนนิยมในระบบที่มีการเข้าชมมากที่สุด และนับจำนวนการเข้าชมแต่ละเพจ

**Why:** เพื่อทำมาวิเคราะห์ความนิยมของเนื้อหาหรือเพจ และช่วยในการปรับปรุง หากเพจไหนที่มีการเข้าชมน้อย ก็จะได้มีการปรับปรุงเนื้อหา

#### 4. Regions with More Than 4 Users

```
SELECT regionid, COUNT(*) AS user_count
FROM 2_users
GROUP BY regionid
HAVING user_count > 4
ORDER BY user_count DESC;
```

The screenshot shows the Pinot Query Console interface. On the left, there's a sidebar with links to Cluster Manager, Query Console (which is selected), Zookeeper Browser, and Swagger REST API. The main area has tabs for TABLES and SQL EDITOR. In the SQL EDITOR tab, the previously provided SQL query is pasted. Below it, there are checkboxes for Tracing and Use Multi-Stage Engine, and a Timeout (Milliseconds) input field set to 5. There are also FORMAT (SQL) and RUN QUERY buttons. Under the QUERY RESPONSE STATS section, a table shows the following data:

timeUsedMs	numDocsScanned	totalDocs	numServersQueried	numServersResponded	numSegments
4	65644	65644	1	1	5

At the bottom, there are EXCEL, CSV, and COPY buttons, and a Show JSON format checkbox.

This screenshot shows the detailed results of the SQL query. The results table is titled "QUERY RESULT" and contains the following data:

regionid	user_count
Region_9	7410
Region_8	7376
Region_6	7325
Region_7	7296
Region_2	7267
Region_1	7248
Region_4	7244
Region_3	7239

**What:** เป็นคำสั่งที่ใช้นับจำนวนผู้ใช้ในแต่ละภูมิภาค และ ระบุภูมิภาคที่มีผู้ใช้มากกว่า 4 คน โดยจะเรียงลำดับจากภูมิภาคที่มีผู้ใช้มากที่สุดไปหนึ่งอย่างต่อไป

**Why:** เพื่อนำมาใช้ในการวิเคราะห์การกระจายของผู้ใช้ตามภูมิภาค และนำไปใช้ในการวิเคราะห์การตลาด ถ้าคุณทำการตลาดในหลายภูมิภาค ก็สามารถใช้คำสั่งนี้ในการหาภูมิภาคที่มีจำนวนผู้ใช้ที่มากได้

## E. Dashboard

What: ระบบแสดงข้อมูลผ่านแดชบอร์ด มีอย่างน้อย 4 panels

Why: ช่วยให้ผู้ใช้งานเข้าใจข้อมูลและ insight จากการวิเคราะห์ได้ง่าย

How: Input: ผลลัพธ์จาก Apache Pinot

Process: จัดแสดงข้อมูลในรูปแบบกราฟ, ตาราง

Output: Dashboard ที่ใช้งานได้จริงและช่วยวิเคราะห์ข้อมูล

### i. At least 4 panels



Link streamlit :

<https://phornpailincosmetic.streamlit.app>

Code streamlit:

```
import pandas as pd
import plotly.express as px
import streamlit as st
```

```

from pinotdb import connect

# Function to fetch data from Pinot
def fetch_data(query):
    conn = connect(host='13.250.117.179', port=8099, path='/query/sql', schema='http')
    curs = conn.cursor()
    curs.execute(query)
    tables = [row for row in curs.fetchall()]
    return tables

# Query for User Distribution by Gender per Region
query_gender_region = """
SELECT
    regionid,
    gender,
    COUNT(*) AS totalUsers
FROM 2_users
GROUP BY regionid, gender
ORDER BY totalUsers DESC;
"""

# Query for User Pageviews
query_pageviews_user = """
SELECT
    userid,
    COUNT(*) AS totalPageviews
FROM 1_pageviews
GROUP BY userid
ORDER BY totalPageviews DESC;
"""

# Query for Regions with More Than 4 Users
query_regions = """
SELECT regionid, COUNT(*) AS user_count
FROM 2_users
GROUP BY regionid
HAVING user_count > 4
ORDER BY user_count DESC;
"""

# Fetch data
gender_region_data = fetch_data(query_gender_region)
pageviews_user_data = fetch_data(query_pageviews_user)
regions_data = fetch_data(query_regions)

# Convert to DataFrame
df_gender_region = pd.DataFrame(gender_region_data, columns=['regionid', 'gender', 'totalUsers'])
df_pageviews_user = pd.DataFrame(pageviews_user_data, columns=['userid', 'totalPageviews'])
df_regions = pd.DataFrame(regions_data, columns=["regionid", "user_count"])

# Streamlit Layout: 2 Rows and 2 Columns
col1, col2 = st.columns(2)
col3, col4 = st.columns(2)

# First chart: Pie chart - User Distribution by Gender per Region

```

```

with col1:
    fig1 = px.pie(df_gender_region, names='gender', values='totalUsers',
                  title='User Distribution by Gender per Region', color='gender')
    st.plotly_chart(fig1)

# Second chart: Bar chart - Total Pageviews per User
with col2:
    fig2 = px.bar(df_pageviews_user, x="userid", y="totalPageviews", title="Pageviews per User",
                  labels={"userid": "User ID", "totalPageviews": "Total Pageviews"},
                  color="totalPageviews", color_continuous_scale="Blues")
    st.plotly_chart(fig2)

# Third chart: Bar chart - Regions with More Than 4 Users
with col3:
    fig3 = px.bar(df_regions, x="regionid", y="user_count", title="Regions with More Than 4 Users",
                  labels={"regionid": "Region ID", "user_count": "User Count"})
    st.plotly_chart(fig3)

# Display the DataFrame of regions data in the fourth column
with col4:
    st.write("### Regions Data")
    st.dataframe(df_regions)

```