# ข้อที่ 1: การปรับปรุงประสิทธิภาพฐานข้อมูล

1. ยกตัวอย่าง SELECT * FROM users WHERE city = “bangkok” db จะใช้ full scan ซึ่งถ้าข้อมูลใน table users มีจำนวนมากอาจจะใช้เวลานาน วิธีแก้คือสร้าง composite index โดยใช้ command CREATE INDEX city_idx ON users(city) และรัน query อีกรอบ  SELECT * FROM users WHERE city = “bangkok” เพียงเท่านี้ db จะใช้ pointer ที่เราสร้างจาก command และลดเวลาในการดึง query

2. ใช้ LIMIT และ OFFSET ใน case ที่เราไม่ต้องการดึงทุก row เพื่อเพิ่มประสิธิภาพในการ run query ต่อครั้ง

3. การใช้ join แทน nested query เช่น SELECT * FROM orders WHERE customer_id IN ( SELECT id FROM users WHERE city = 'bangkok'' ); แทนด้วย SELECT * FROM orders INNER JOIN customers ON orders.customer_id = customers.id WHERE customers.city = 'New York';

4. การออกแบบฐานข้อมูลให้มีความรวดเร็ว ถ้าเรามี column เยอะเกินไปควรแยก table ยกตัวอย่าง

| OrderID | ProductName | Quantity | Price | CustomerName |
|---|---|---|---|---|
| 1001 | ProductA | 2 | 10 | John Doe |
| 1001 | ProductB | 1 | 15 | John Doe |

แยก table ออกมาเป็น 3 table orders  orderItems  และ customers

| OrderID | ProductID | Quantity | Price |
|---|---|---|---|
| 1001 | 3001 | 2 | 10 |
| 1001 | 3002 | 1 | 15 |

| CustomerID | CustomerName |
|---|---|
| 2001 | John Doe |

| OrderID | CustomerID | OrderDate |
|---|---|---|
| 1001 | 2001 | 2023-11-22 |

5. ในส่วนที่ query ที่ถูกใช้บ่อยและดึงข้อมูลมาหลาย rows ต่อครั้ง caching เป็นวิธีที่เหมาะสมเช่นกันและมีทั้งข้อดีและข้อเสีย โดย service ที่ใช้มี redis, dynamoDB, mongoDB วิธีการ save cache ใช้ query เป็น key และ value เป็น ผลของ query เก็บไว้เป็น json format

-ข้อดี
ลด server load โดยการดึงข้อมูลจาก caching service แทน

การดึงข้อมูลจาก caching service มีความรวดเร็วกว่า

ลดค่าใช้จ่ายโดยการลด server load

-ข้อเสีย
ข้อมูลไม่ได้ถูก update แบบ real time เนื่องจาก caching จะเซ็ต expiration limit อย่างน้อย 20 นาที

Complexity สุมมุติข้อมูลที่เก็บใน cache ถูกลบแล้ว caching service ก้ยังคงเก็บมันใว้ ถ้าเวลาลบน้อยกว่าที่กำหนด ซึ่งอาจจะทำให้ backend ต้องเพิ่ม function ในการเช็ค caching ไปด้วย