

รายงาน

เว็บไซต์เพื่อการจัดการข้อมูลสำหรับร้านค้าชาบู 3 พี่น้อง

จัดทำโดย

กวีพัฒน์ ภควัตสิริกุล 6342002026

ชนกันต์ กนกมณีรัตน์ 6342015226

ปฏิญญา จารุอริยานนท์ 6342057626

ณัฐณิชา​ เมืองชู 6442025626

นัทธมนต์ ตินตะชาติ 6442046826​

ศุภาพิชญ์ ตั้งอำนวยสมบัติ 6442107426

เสนอ

รศ.ดร.วรสิทธิ์ ชูชัยวัฒนา

รายงานนี้เป็นส่วนหนึ่งของรายวิชา 2603491 การโปรแกรมสำหรับฐานข้อมูล

คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย

ภาคเรียนที่ 1 ปีการศึกษา 2566

**คำนำ**

        รายงานนี้เป็นส่วนหนึ่งของรายวิชา 2603491 การโปรแกรมฐานข้อมูลจัดทำขึ้นเพื่อแก้ไขปัญหาการจัดการการบริหารและการดำเนินการภายในร้านวิเคราะห์ข้อมูลการขายและการจัดการวัตถุดิบ โดยทางคณะผู้จัดทำการสร้างและพัฒนา Web Application ให้ตอบโจทย์กับการจัดการธุรกิจ Shabu 3 Peenong   
(ชาบู 3 พี่น้อง) โดยใช้ความรู้เกี่ยวกับระบบฐานข้อมูลและการสร้าง Web Application รวมถึงการใช้ Django มาช่วยในการเชื่อมต่อระบบฐานข้อมูลกับ Web Application ที่สร้างขึ้น

        ทางคณะผู้จัดทำขอขอบพระคุณอาจารย์ รศ.ดร.วรสิทธิ์ ชูชัยวัฒนา ผู้ที่ได้กรุณาชี้แนะและให้ความช่วยเหลือแนวทางการศึกษา คณะผู้จัดทำหวังว่ารายงานฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจการจัดการฐานข้อมูลในร้านอาหาร หรือธุรกิจชาบู และที่สำคัญคือจะช่วยแก้ไขปัญหาขอร้าน Shabu 3 Peenong ได้ หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด คณะผู้จัดทำขอน้อมรับไว้และขออภัยมา ณ ที่นี้

คณะผู้จัดทำ

**สารบัญ**

[1.บทนำ 1](#_Toc153925759)

[หัวข้อที่ศึกษา 1](#_Toc153925760)

[ที่มาและความสำคัญของปัญหา 1](#_Toc153925761)

[ปัญหาที่ต้องการแก้ไข 1](#_Toc153925762)

[แนวทางการแก้ไข 2](#_Toc153925763)

[ประโยชน์ที่คาดว่าจะได้รับ 2](#_Toc153925764)

[2. การออกแบบฐานข้อมูล 3](#_Toc153925765)

[เครื่องมือที่ใช้ 3](#_Toc153925766)

[การออกแบบ Entity Relationship Diagram 3](#_Toc153925767)

[คำอธิบายตัวแปร 5](#_Toc153925768)

[Programmable Object 11](#_Toc153925769)

[3. การออกแบบ  Web Application 22](#_Toc153925770)

[ผู้ใช้งาน (Users) 22](#_Toc153925771)

[ระบบและขอบเขตการทำงานของ Web Application 22](#_Toc153925772)

[ขอบเขตที่นอกเหนือจากการดำเนินงาน 23](#_Toc153925773)

[4. คู่มือการใช้งานเว็บไซต์ 24](#_Toc153925774)

[5. การสรุปผลและข้อเสนอแนะ 42](#_Toc153925775)

[การสรุปผล 42](#_Toc153925776)

[ข้อเสนอแนะ 42](#_Toc153925777)

[บรรณานุกรม 43](#_Toc153925778)

[ภาคผนวก 44](#_Toc153925779)

**URL GitHub:** <https://github.com/patinya2001/Database_Programming.git>

# 

# **1.บทนำ**

## **หัวข้อที่ศึกษา**

ระบบการวิเคราะห์และรายงานผลประกอบการร้านชาบู 3 พี่น้องผ่านเว็บไซต์เพื่อการจัดการ

## **ที่มาและความสำคัญของปัญหา**

ในปัจจุบันการแข่งขันในตลาดธุรกิจชาบูบุฟเฟต์ในไทยมีการเติบโตเป็นอย่างมาก โดยเฉพาะอย่างยิ่งในการค้นหาธุรกิจบุฟเฟ่ต์ในประเทศไทยเทียบตั้งแต่พ.ศ. 2547 จนถึง ปี 2566 ที่มากขึ้นถึง 1 แสนครั้ง ต่อ เดือน (สุรสิทธิ์ สัจจะเดว์, 2565) นอกจากนี้การสืบค้นหา Brand Keywords ประเภทบุฟเฟ่ต์ชาบูยังมาถึงร้อยละ 15.4 ของการสืบค้นทั้งหมด การบริหารจัดการต้นทุนและผลประกอบการขายเพื่อการแข่งกับกับคู่แข่งจึงเป็นส่วนที่มีความสำคัญเป็นอย่างยิ่งสำหรับผู้ประกอบการ พบว่าทางร้านชาบู 3 พี่น้องจัดการข้อมูลผลการดำเนินงานทุกเดือนจากการดึงข้อมูลระบบ POS บน loyverse และนำมาทำรายงานบน Microsoft Excel ทุกสิ้นเดือน เป็นผลให้ผู้ประกอบการร้านชาบู 3 พี่น้องต้องเสียเวลาและทรัพยากรในการจัดการรูปดังกล่าวเป็นจำนวนมาก ร้านจึงมีความจำเป็นต้องจัดการรายงานสรุปผลการดำเนินงานให้สะดวกรวดเร็วมากขึ้นด้วยการจัดการระบบฐานข้อมูล จากการดำเนินงานจะพบว่าในเดือนล่าสุดธุรกิจมีการขาดทุนทำให้ผู้ประกอบการต้องการทราบข้อมูลการจัดการและบริหารต้นทุนของการดำเนินงานทั้งหมด 3 สาขา อีกทั้งการจัดการในระบบ POS บน loyverse สามารถติดตามจำนวนยอดขายได้ในช่วงสิ้นเดือน แต่หากทางร้านชาบู 3 พี่น้องต้องการข้อมูลสรุปยอดขายเพื่อนำไปจัดทำการส่งเสริมทางการตลาดจะทำให้ต้องรอข้อมูลจนถึงสิ้นเดือนส่งผลให้ธุรกิจเสียโอกาสที่จะสร้างยอดขาย ด้วยปัญหาดังกล่าว ทางคณะผู้จัดจึงได้ทำการสร้างระบบฐานข้อมูลและ Web Application เพื่อใช้ในการวิเคราะห์ข้อมูลยอดขายและต้นทุนจากการดำเนินงาน รวมถึงการทำรายงานผลประกอบการจากการวิเคราะห์ข้อมูลให้สะดวกและรวดเร็วมากขึ้น

## **ปัญหาที่ต้องการแก้ไข**

1. ลดขั้นตอนในการวิเคราะห์สรุปผลการดำเนินงาน
2. ต้องการวิเคราะห์ข้อมูลต้นทุนจากการดำเนินงาน
3. ต้องการวิเคราะห์สินค้าขายดีในแต่ละเดือน
4. อัพเดตข้อมูลที่เกิดขึ้น เช่น ค่าใช้จ่ายรายวัน การเข้าทำงานของพนักงาน
5. ตรวจสอบความถูกต้องของฐานข้อมูล ว่ามีความสอดคล้องกันหรือไม่ เช่น ข้อมูลใบเสร็จกับจำนวนผลรวมของการขาย

## **แนวทางการแก้ไข**

พัฒนา Web Application ที่เชื่อมต่อกับฐานข้อมูลเชิงความสัมพันธ์

## **ประโยชน์ที่คาดว่าจะได้รับ**

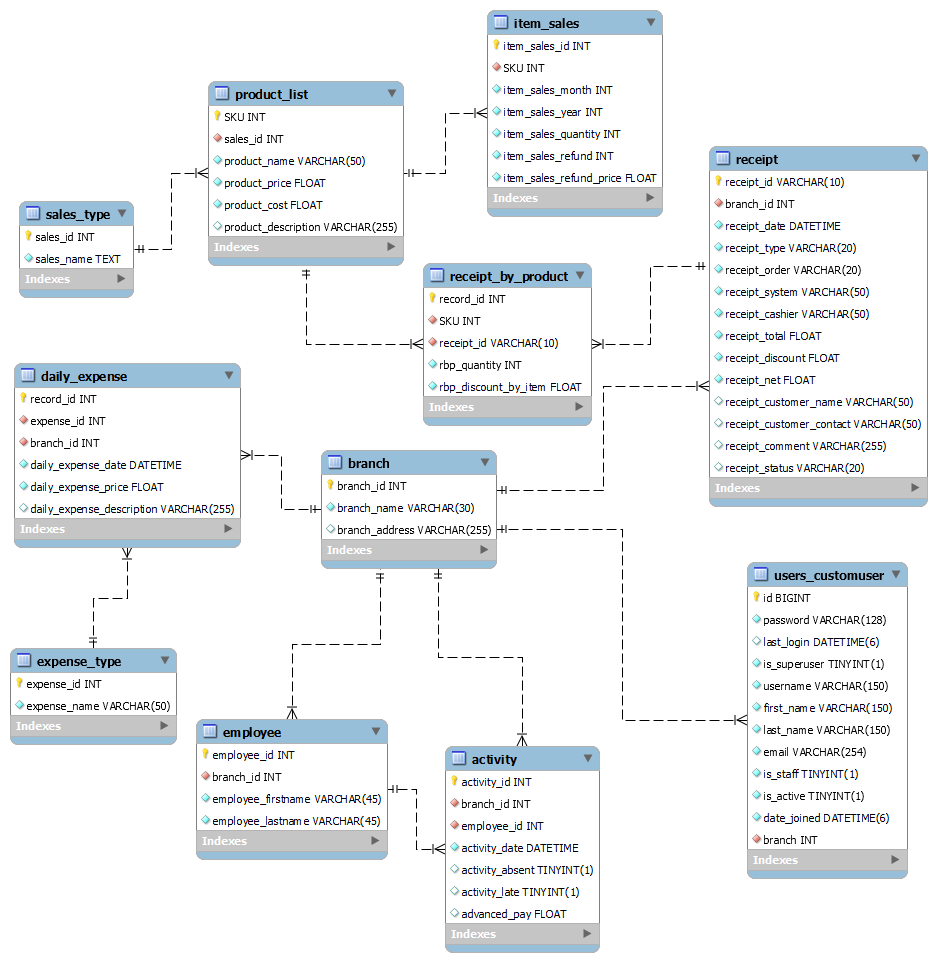
1. ช่วยสรุปผลการดำเนินงานให้สะดวกและรวดเร็วขึ้น
2. ช่วยจัดทำการวิเคราะห์ข้อมูลต้นทุนจากการดำเนินงาน
3. ช่วยจัดทำการวิเคราะห์สินค้าขายดีในแต่ละเดือนเพื่อนำไปต่อยอดในด้านการส่งเสริมการขายต่อไป
4. สามารถสร้างการสรุปและแสดงข้อมูลออกมาให้อยู่ในรูปของแผนภาพ กราฟ
5. สามารถจัดการสิทธิ์ในการเข้าถึงเพื่อดูและแก้ไขระบบฐานข้อมูลที่แสดงผลบนเว็บไซต์ให้การจัดเก็บข้อมูลมีความปลอดภัยต่อธุรกิจมากขึ้น

# **2. การออกแบบฐานข้อมูล**

## **เครื่องมือที่ใช้**

MySQL WorkBench

## **การออกแบบ Entity Relationship Diagram**



จากศึกษาข้อมูลผลการดำเนินจาก Microsoft Excel และข้อมูลบนระบบ POS ร้านชาบู 3 พี่น้อง คณะผู้จัดทำจึงได้ออกแบบระบบฐานข้อมูลซึ่งประกอบด้วย 11 ตาราง ได้แก่

1. **daily\_expense**

เก็บค่าใช้จ่ายรายวันที่เกิดขึ้น

1. **expense\_type**

เก็บข้อมูลประเภทค่าใช้จ่ายที่เกิดขึ้น

1. **employee**

เก็บข้อมูลพนักงานและสาขาที่ทำงาน

1. **branch**

เก็บข้อมูลชื่อสาขาและที่ตั้งสาขา

1. **activity**

เก็บข้อมูลการบันทึกการเข้าทำงานของพนักงานแยกตามสาขาและการเบิกจ่ายค่าแรงล่วงหน้า

1. **users\_customer**

เก็บข้อมูลเกี่ยวกับผู้ใช้งานเว็ปไซต์ของร้าน

1. **receipt**

เก็บข้อมูลลูกค้า รายการสินค้าที่สั่ง และยอดขาย

1. **receipt\_by\_product**

เก็บข้อมูลใบเสร็จและจำนวนสินค้าที่ขายได้ในใบเสร็จ

1. **item\_sales**

เก็บรายการขายสินค้า

1. **product\_list**

เก็บข้อมูลสินค้าที่ขายในร้าน

1. **sales\_type**

เก็บข้อมูลประเภทการขาย

## **คำอธิบายตัวแปร**

|  |  |  |  |
| --- | --- | --- | --- |
| daily\_expense | | | |
| No. | Column | Data Type | Description |
| 1 | record\_id | PRIMARY KEY | id ของ record |
| 2 | expense\_id | INT | รหัสค่าใช้จ่าย |
| 3 | branch\_id | INT | รหัสสาขา |
| 4 | daily\_expense\_date | DATETIME | วันที่เกิดค่าใช้จ่าย |
| 5 | daily\_expense\_price | FLOAT | ราคาค่าใช้จ่าย |
| 6 | daily\_expense\_description | VARCHAR | รายละเอียดของค่าใช้จ่ายที่เกิดขึ้น |

|  |  |  |  |
| --- | --- | --- | --- |
| expense\_type | | | |
| No. | Column | Data Type | Description |
| 1 | expense\_id | PRIMARY KEY | รหัสของค่าใช้จ่าย |
| 2 | expense\_name | VARCHAR | ชื่อของค่าใช้จ่าย |

|  |  |  |  |
| --- | --- | --- | --- |
| employee | | | |
| No. | Column | Data Type | Description |
| 1 | employee\_id | PRIMARY KEY | รหัสของพนักงาน |
| 2 | branch\_id | INT | รหัสสาขา |
| 3 | employee\_firstname | VARCHAR | ชื่อของพนักงาน |
| 4 | employee\_lastname | VARCHAR | นามสกุลของพนักงาน |

|  |  |  |  |
| --- | --- | --- | --- |
| branch | | | |
| No. | Column | Data Type | Description |
| 1 | branch\_id | PRIMARY KEY | รหัสสาขา |
| 2 | branch\_name | VARCHAR | ชื่อของสาขา |
| 3 | branch\_address | VARCHAR | รายละเอียดที่อยู่ของสาขา |

|  |  |  |  |
| --- | --- | --- | --- |
| activity | | | |
| No. | Column | Data Type | Description |
| 1 | activity\_id | INT | id การบันทึกการเข้าทำงานของพนักงาน |
| 2 | branch\_id | INT | รหัสสาขา |
| 3 | employee\_id | INT | รหัสพนักงาน |
| 4 | activity\_date | DATETIME | วันที่บันทึกการเข้าทำงานของพนักงาน |
| 5 | activity\_absent | TINYINT | บันทึกวันเข้าทำงาน  0 = ไม่มาทำงาน  1 = มาทำงาน |
| 6 | activity\_late | TINYINT | บันทึกวันเข้าทำงานสาย  0 = ไม่สาย  1 = สาย |
| 7 | acvanced\_pay | FLOAT | ค่าแรกเบิกจ่ายล่วงหน้า |

|  |  |  |  |
| --- | --- | --- | --- |
| users\_customuser | | | |
| No. | Column | Data Type | Description |
| 1 | id | PRIMARY KEY | รหัสผู้ใช้งาน |
| 2 | password | VARCHAR | รหัสผ่านสำหรับการล็อกอิน |
| 3 | last\_login | DATETIME | วันที่เข้าใช้งานครั้งสุดท้าย |
| 4 | is\_superuser | TINYINT(1) | เป็น Admin หรือ Account ที่ไม่มีสิทธิพิเศษหรือไม่  0 = ไม่มี  1 = มี |
| 5 | is\_staff | TINYINT(1) | เป็นพนักงานหรือไม่  0 = ไม่เป็นพนักงาน  1 = เป็นพนักงาน |
| 6 | is\_active | TINYINT(1) | เป็น account ที่ยังคงมีการใช้งาน  0 = ไม่ใช้งาน  1 = ยังคงมีการใช้งาน |
| 7 | date\_joined | DATETIME | วันที่เข้าใช้งานระบบครั้งแรก |
| 8 | username | VARCHAR | รหัสชื่อสำหรับการเข้าใช้งาน |
| 9 | first\_name | VARCHAR | ชื่อจริง |
| 10 | last\_name | VARCHAR | นามสกุล |
| 11 | email | VARCHAR | อีเมล |
| 12 | branch\_id | INT | รหัสสาขา |

|  |  |  |  |
| --- | --- | --- | --- |
| receipt | | | |
| No. | Column | Data Type | Description |
| 1 | receipt\_id | VARCHAR | รหัสใบเสร็จ |
| 2 | branch\_id | INT | รหัสสาขา |
| 3 | receipt\_date | DATETIME | วันที่ออกใบเสร็จ |
| 4 | receipt\_type | VARCHAR | ประเภทใบเสร็จ |
| 5 | receipt\_order | VARCHAR | รายการสินค้าที่สั่ง |
| 6 | receipt\_system | VARCHAR | ระบบขายหน้าร้าน |
| 7 | receipt\_cashier | VARCHAR | พนักงานที่ออกใบเสร็จ |
| 8 | receipt\_total | FLOAT | ยอดรวม |
| 9 | receipt\_discount | FLOAT | ส่วนลด |
| 10 | receipt\_net | FLOAT | ยอดสุทธิ |
| 11 | receipt\_customer\_name | VARCHAR | ชื่อลูกค้า |
| 12 | receipt\_customer\_contact | VARCHAR | ข้อมูลติดต่อลูกค้า |
| 13 | receipt\_comment | VARCHAR | ความคิดเห็น |
| 14 | receipt\_status | VARCHAR | สถานะของใบเสร็จ |

|  |  |  |  |
| --- | --- | --- | --- |
| receipt\_by\_product | | | |
| No. | Column | Data Type | Description |
| 1 | record\_id | INT | รหัสใบเสร็จตามสินค้า |
| 2 | SKU | INT | รหัสสินค้า |
| 3 | receipt\_id | VARCHAR | รหัสใบเสร็จ |
| 4 | rbp\_quantity | INT | จำนวนที่ขายได้ |
| 5 | rbp\_discount\_by\_item | FLOAT | ราคาส่วนลดของแต่ละสินค้า |

|  |  |  |  |
| --- | --- | --- | --- |
| item\_sales | | | |
| No. | Column | Data Type | Description |
| 1 | item\_sales\_id | INT | รหัสการบันทึกยอดขายสินค้า |
| 2 | SKU | INT | รหัสสินค้า |
| 3 | item\_sales\_month | INT | เดือนที่สินค้าขายได้ |
| 4 | item\_sales\_year | INT | ปีที่สินค้าขายได้ |
| 5 | item\_sales\_qunatity | INT | ยอดรวมสินค้าที่ขายได้ |
| 6 | item\_sales\_refund | INT | สินค้าที่รับคืน |
| 7 | item\_sales\_refund\_  price | FLOAT | มูลค่าการคืนเงิน |

|  |  |  |  |
| --- | --- | --- | --- |
| product\_list | | | |
| No. | Column | Data Type | Description |
| 1 | SKU | INT | รหัสสินค้า |
| 2 | sales\_id | INT | รหัสประเภทการขาย |
| 3 | product\_name | VARCHAR | ชื่อสินค้า |
| 4 | product\_price | FLOAT | ราคาสินค้า |
| 5 | product\_cost | FLOAT | ต้นทุนสินค้า |
| 6 | product\_description | VARCHAR | คำอธิบายสินค้า |

|  |  |  |  |
| --- | --- | --- | --- |
| sales\_type | | | |
| No. | Column | Data Type | Description |
| 1 | sales\_id | INT | รหัสประเภทการขาย |
| 2 | sales\_name | TEXT | ชื่อของประเภทการขาย |

## **Programmable Object**

**Stored Procedure**

1. **InsertProduct**

Description: ใช้ในการ insert ข้อมูลสินค้าเข้าตาราง product\_list

Input: รหัสสินค้า (SKU)

รหัสประเภทการขาย (sales\_id)

ชื่อประเภทการขาย (sales)

ชื่อสินค้า (product\_name)

ราคาสินค้า (product\_price)

ต้นทุนสินค้า (product\_cost)

คำอธิบายสินค้า (product\_descrption)

Output: -

Process:  รับข้อมูล SKU, sales\_id, sales, product\_name, product\_price, product\_cost และ product\_description จากนั้นเช็คว่า product\_description มีค่าหรือไม่ หากไม่มีจะกำหนดเป็น null และเช็คว่า sales มีค่าหรือไม่ หากมีค่าจะเลือก sales\_id ที่มีประเภทการขายตรงกับ sales จากตาราง sales\_type ไปใส่ใน sales\_name\_id โดยใช้ฟังก์ชัน COALESCE เช็คหาก sales\_id ที่รับมามีค่าเป็น null จะเลือกใช้ sales\_name\_id แทน และนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง product\_list

1. **InsertExpense**

Description: ใช้ในการ insert ข้อมูลค่าใช้จ่ายเข้าตาราง daily\_expense

Input: รหัสค่าใช้จ่าย (expense\_id)

รหัสสาขา (branch\_id)

วันที่เกิดค่าใช้จ่าย (daily\_expense\_date)

ราคาค่าใช้จ่าย (daily\_expense\_price)

รายละเอียดของค่าใช้จ่ายที่เกิดขึ้น (daily\_expense\_description)

Output: -

Process: รับข้อมูล expense\_id, branch\_id, daily\_expense\_date, daily\_expense\_price และ daily\_expense\_description จากนั้นเช็คว่า daily\_expense\_description มีค่าหรือไม่ หากไม่มีจะกำหนดเป็น null และนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง daily\_expense

1. **InsertActivity**

Description: ใช้ในการ insert ข้อมูลเข้าตาราง activity

Input: เป็น admin หรือไม่ (is\_superuser)

รหัสสาขา (branch\_id)

รหัสพนักงาน (employee\_id)

วันที่บันทึกการเข้าทำงานของพนักงาน (activity\_date)

บันทึกวันเข้ามาทำงาน (activity\_absent)

บันทึกวันเข้าทำงานสาย (activity\_late)

Output: -

Process: รับข้อมูล is\_superuser, branch\_id, employee\_id, activity\_date, activity\_absent และ activity\_late จากนั้นเช็คว่าใช่ admin หรือไม่ หากใช่จะนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง activity หากไม่ใช่จะไม่สามารถนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง activity ได้

1. **InsertItemSales**

Description: ใช้ในการ insert ข้อมูลเข้าตาราง item\_sales

Input: รหัสสินค้า (SKU)

เดือนที่ขายสินค้าได้ (item\_sales\_month)

ปีที่ขายสินค้าได้ (item\_sales\_year)

สินค้าที่ขายได้ (item\_sales\_quantity)

สินค้าที่รับคืน (item\_sales\_refund)

มูลค่าการคืนเงิน (item\_sales\_refund\_price)

Output: -

Process: รับข้อมูล SKU, item\_sales\_month, item\_sales\_year, item\_sales\_quantity, item\_sales\_refund และ item\_sales\_refund\_price จากนั้นนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง item\_sales

1. **InsertReceipt**

Description: ใช้ในการ insert ข้อมูลเข้าตาราง receipt

Input: รหัสใบเสร็จ (receipt\_id)

ชื่อสาขา (branch)

วันที่ออกใบเสร็จ (receipt\_date)

ประเภทใบเสร็จ (receipt\_type)

รายการสินค้าที่สั่ง (receipt\_order)

ระบบขายหน้าร้าน (receipt\_system)

พนักงานที่ออกใบเสร็จ (receipt\_cashier)

ยอดรวม (receipt\_total)

ส่วนลด (receipt\_discount)

ยอดรวมสุทธิ (receipt\_net)

ชื่อลูกค้า (receipt\_customer\_name)

ข้อมูลติดต่อลูกค้า (receipt\_customer\_contact)

ความคิดเห็น (receipt\_comment)

สถานะของใบเสร็จ (receipt\_status)

Output: -

Process: รับข้อมูล receipt\_id, branch, receipt\_date, receipt\_type, receipt\_order, receipt\_system, receipt\_cashier, receipt\_total, receipt\_discount, receipt\_net, receipt\_customer\_name, receipt\_customer\_contact, receipt\_comment และ receipt\_status จากนั้นเช็ค branch เพื่อกำหนดรหัสสาขา (branch\_id) ให้ตรงตามชื่อสาขาที่รับมาและทำการ insert ข้อมูลนำเข้าเหล่านี้เข้าตาราง item\_sales

1. **InsertReceiptByProduct**

Description: ใช้ในการ insert ข้อมูลเข้าตาราง receipt\_by\_product

Input: รหัสสินค้า (SKU)

รหัสใบเสร็จ (receipt\_id)

จำนวนที่ขายได้ (rbp\_quantity)

ราคาส่วนลดของแต่ละสินค้า (rbp\_discount\_by\_item)

Output: -

Process: รับข้อมูล SKU, receipt\_id, rbp\_quantity และ rbp\_discount\_by\_item จากนั้นนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง receipt\_by\_item

**View**

1. **home\_daily**

Description: ใช้ในการแสดงข้อมูลยอดขายในแต่ละวัน

Input: -

Output: วันที่

รายได้รวมสุทธิ

Process: เลือกข้อมูลในคอลัมน์ receipt\_date และ receipt\_total จากตาราง receipt จากนั้นใช้ฟังก์ชัน CAST เพื่อดึงวันออกมาจาก receipt\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่าวันที่ และใช้ฟังก์ชัน SUM กับคอลัมน์ receipt\_total เพื่อคำนวณหารายได้รวมแล้วตั้งชื่อคอลัมน์ใหม่ว่ารายได้รวมสุทธิ โดย GROUP BY ตามวันที่ และ ORDER BY ตามวันที่

1. **summary\_view**

Description: ใช้ในการคำนวณหารายได้รวม รายได้เฉลี่ยต่อเดือน ค่าใช้จ่ายรวม และค่าใช้จ่ายเฉลี่ยต่อเดือน

Input: -

Output: เดือนปี (month\_year)

รายได้รวม (total\_income)

รายได้เฉลี่ยต่อเดือน (avg\_income)

ค่าใช้จ่ายรวม (total\_expense)

ค่าใช้จ่ายเฉลี่ยต่อเดือน (avg\_expense)

Process: เลือกข้อมูลในคอลัมน์ month\_year,  total\_income, avg\_income, total\_expense และ avg\_expense จากตาราง receipt UNION กับตาราง daily\_expense โดยใช้ฟังก์ชัน SUM กับคอลัมน์ receipt\_net และ daily\_expense\_price เพื่อคำนวณหารายได้รวมและค่าใช้จ่ายรวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_income และ total\_expense, ใช้ฟังก์ชัน AVG กับคอลัมน์ receipt\_net และ daily\_expense\_price เพื่อคำนวณหารายได้และค่าใช้จ่ายเฉลี่ยต่อเดือนแล้วตั้งชื่อคอลัมน์ใหม่ว่า avg\_income และ avg\_expnse โดย GROUP BY month\_year

1. **total\_income\_by\_type**

Description: ใช้ในการคำนวณหารายได้รวมแยกตามประเภทของใบเสร็จ

Input: -

Output: เดือนปี (month\_year)

ประเภทรายได้ (receipt\_type)

รายได้รวม (total\_income)

Process: เลือกข้อมูลในคอลัมน์ month\_year, receipt\_type และ total\_income จากตาราง receipt โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ receipt\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และใช้ฟังก์ชัน SUM กับคอลัมน์ receipt\_net เพื่อคำนวณหารายได้รวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_income โดย GROUP BY month\_year และ receipt\_type

1. **total\_income\_by\_branch**

Description: ใช้ในการคำนวณหารายได้รวมแยกตามสาขา

Input: -

Output: เดือนปี (month\_year)

ชื่อสาขา (branch\_name)

รายได้รวม (total\_income)

Process: เลือกข้อมูลในคอลัมน์ month\_year, branch\_name และ total\_income จากตาราง receipt INNER JOIN กับตาราง branch โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ receipt\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และใช้ฟังก์ชัน SUM กับคอลัมน์ receipt\_net เพื่อคำนวณหารายได้รวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_income โดย GROUP BY month\_year และ branch name

1. **total\_expense\_by\_type**

Description: ใช้ในการคำนวณหาค่าใช้จ่ายรวมแยกตามประเภทของค่าใช้จ่าย

Input: -

Output: เดือนปี (month\_year)

ประเภทค่าใช้จ่าย (expense\_name)

ค่าใช้จ่ายรวม (total\_expense)

Process: เลือกข้อมูลในคอลัมน์ month\_year, expense\_name และ total\_expense จากตาราง activity UNION กับตาราง daily\_expense INNER JOIN กับตาราง expense\_type โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ activity\_date และ daily\_expense\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และใช้ฟังก์ชัน SUM กับคอลัมน์ advanced\_pay และ daily\_expense\_price  เพื่อคำนวณหาค่าใช้จ่ายรวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_advanced\_pay และ total\_daily\_expense\_price จากนั้นจึงนำ total\_advanced\_pay และ total\_daily\_expense\_price มาคำนวณหา total\_expense ต่อไป โดย GROUP BY month\_year และ expense\_name

1. **total\_expense\_by\_branch**

Description: ใช้ในการคำนวณหาค่าใช้จ่ายรวมแยกตามสาขา

Input: -

Output: เดือนปี (month\_year)

ชื่อสาขา (branch\_name)

ค่าใช้จ่ายรวม (total\_expense)

Process: เลือกข้อมูลในคอลัมน์ month\_year, branch\_name และ total\_expense จากตาราง activity UNION กับตาราง daily\_expense และนำมา INNER JOIN กับตาราง branch โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ activity\_date และ daily\_expense\_date  แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และใช้ฟังก์ชัน SUM กับคอลัมน์ advanced\_pay และ daily\_expense\_price  เพื่อคำนวณหาค่าใช้จ่ายรวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_advanced\_pay และ total\_daily\_expense\_price เพื่อคำนวณหา total\_expense ต่อไป โดย GROUP BY month\_year และ branch

1. **top\_ten\_sales**

Description: ใช้ในการคำนวณหาสินค้าที่ขายได้สูงสุด 10 อันดับแรก

Input: -

Output: เดือนปี (month\_year)

ชื่อสินค้า (product\_name)

สินค้าที่ขายได้รวม (total\_sales)

Process: เลือกข้อมูลในคอลัมน์ month\_year, product\_name และ total\_sales จากตาราง item\_sales INNER JOIN กับตาราง product\_list โดยใช้ฟังก์ชัน CONCAT ในการดึงเดือนปีจากคอลัมน์ item\_sales\_year และ item\_sales\_month แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และใช้ฟังก์ชัน SUM กับคอลัมน์ item\_sales\_quantity เพื่อคำนวณหาจำนวนที่ขายได้รวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_sales โดย GROUP BY month\_year และ product\_name แล้ว ORDER BY total\_sales เลือกมาเฉพาะสินค้าที่ขายได้สูงสุด 10 อันดับแรก

1. **total\_income\_before\_after\_discount**

Description: ใช้ในการคำนวณหารายได้ก่อนและหลังหักส่วนลด

Input: -

Output: เดือนปี (month\_year)

รายได้ก่อนหักส่วนลดรวม (total\_income)

รายได้หลังหักส่วนลดรวม (total\_net\_income)

Process: เลือกข้อมูลในคอลัมน์ month\_year, total\_income และ total\_net\_income จากตาราง receipt โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ receipt\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year จากนั้นใช้ฟังก์ชัน SUM กับคอลัมน์ receipt\_total และ receipt\_net เพื่อคำนวณหารายได้ก่อนหักส่วนลดรวมและรายได้หลังหักส่วนลดรวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_income และ total\_net\_income โดย GROUP BY month\_year

1. **total\_income\_by\_day**

Description: ใช้ในการคำนวณหารายได้รวมในแต่ละวัน

Input: -

Output: เดือนปี (month\_year)

วันที่ออกใบเสร็จ (receipt\_date)

รายได้รวม (total\_income)

Process: เลือกข้อมูลในคอลัมน์ month\_year, receipt\_date และ total\_income จากตาราง receipt โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ receipt\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และใช้ฟังก์ชัน SUM กับคอลัมน์ receipt\_net เพื่อคำนวณหารายได้รวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_income โดย GROUP BY month\_year และ receiot\_date ORDER BY receipt\_date

1. **total\_expense\_and\_count\_by\_month**

Description: ใช้ในการคำนวณหาค่าใช้จ่ายรวมและจำนวนรายการค่าใช้จ่ายรวมในแต่ละเดือน

Input: -

Output: เดือนปี (month\_year)

ค่าใช้จ่ายรวม (total\_expense)

จำนวนรายการค่าใช้จ่ายรวม (count)

Process: เลือกข้อมูลในคอลัมน์ month\_year, total\_expense และ count จากตาราง daily\_expense โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ daily\_expense\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year, ใช้ฟังก์ชัน SUM กับคอลัมน์ daily\_expense\_price เพื่อคำนวณหาค่าใช้จ่ายรวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_expense และใช้ฟังก์ชัน COUNT กับคอลัมน์ record\_id เพื่อคำนวณหาจำนวนรายการค่าใช้จ่ายรวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า count โดย GROUP BY month\_year

1. **total\_expense\_by\_day**

Description: ใช้ในการคำนวณหาค่าใช้จ่ายรวมรายวัน

Input: -

Output: เดือนปี (month\_year)

วันที่เกิดค่าใช้จ่าย (daily\_expense\_date)

ค่าใช้จ่ายรวม (total\_expense)

Process: เลือกข้อมูลในคอลัมน์ month\_year, daily\_expense\_date และ total\_expense จากตาราง daily\_expense โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ daily\_expense\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และ ใช้ฟังก์ชัน SUM กับคอลัมน์ daily\_expense\_price เพื่อคำนวณหาค่าใช้จ่ายรวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total\_expense โดย GROUP BY month\_year และ daily\_expense\_date จากนั้น ORDER BY daily\_expense\_date

1. **total\_emp\_late\_absent**

Description: ใช้ในการคำนวณหาจำนวนครั้งที่พนักงานมาสายและไม่มาทำงาน

Input: -

Output: เดือนปี (month\_year)

จำนวนครั้งที่พนักงานมาสาย (count\_late)

จำนวนครั้งที่พนักงานไม่มาทำงาน (count\_absent)

Process: เลือกข้อมูลในคอลัมน์ month\_year, count\_late และ count\_absent จากตาราง activity โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ activity\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และใช้ฟังก์ชัน SUM กับคอลัมน์ activity\_late และ activity\_absent เพื่อคำนวณหาจำนวนครั้งที่พนักงานมาสายและไม่มาทำงานแล้วตั้งชื่อคอลัมน์ใหม่ว่า count\_late และ count\_absent โดย GROUP BY month\_year

1. **total\_emp\_late\_absent\_by\_branch**

Description: ใช้ในการคำนวณหาจำนวนครั้งที่พนักงานมาสายและไม่มาทำงานในแต่ละสาขา

Input: -

Output: เดือนปี (month\_year)

ชื่อสาขา (branch\_name)

จำนวนครั้งที่พนักงานมาสาย (count\_late)

จำนวนครั้งที่พนักงานไม่มาทำงาน (count\_absent)

Process: เลือกข้อมูลในคอลัมน์ month\_year, branch\_name, count\_late และ count\_absent จากตาราง activity INNER JOIN กับตาราง branch โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ activity\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า month\_year และใช้ฟังก์ชัน SUM กับคอลัมน์ activity\_late และ activity\_absent เพื่อคำนวณหาจำนวนครั้งที่พนักงานมาสายและไม่มาทำงานแล้วตั้งชื่อคอลัมน์ใหม่ว่า count\_late และ count\_absent โดย GROUP BY month\_year และ branch\_name

1. **total\_emp\_late\_absent\_day**

Description: ใช้ในการคำนวณหาจำนวนครั้งที่พนักงานมาสายและไม่มาทำงานในแต่ละวัน

Input: -

Output: เดือนปี (month\_year)

วันที่ (by\_date)

จำนวนครั้งที่พนักงานมาสาย (count\_late)

จำนวนครั้งที่พนักงานไม่มาทำงาน (count\_absent)

Process: เลือกข้อมูลในคอลัมน์ month\_year, by\_date, count\_late และ count\_absent จากตาราง activity โดยใช้ฟังก์ชัน DATE\_FORMAT ในการดึงเดือนปีจากคอลัมน์ activity\_date แล้วตั้งชื่อใหม่ว่า month\_year, ใช้ฟังก์ชัน DATE ในการดึงวันที่จากคอลัมน์ activity\_date แล้วตั้งชื่อคอลัมน์ใหม่ว่า by\_date และใช้ฟังก์ชัน SUM กับคอลัมน์ activity\_late และ activity\_absent เพื่อคำนวณหาจำนวนครั้งที่พนักงานมาสายและไม่มาทำงานแล้วตั้งชื่อคอลัมน์ใหม่ว่า count\_late และ count\_absent โดย GROUP BY month\_year และ by\_date

**Trigger**

1. **before\_insert\_product\_list**

Description: ใช้ในการเช็คข้อมูลสินค้าก่อน insert ข้อมูลเข้าตาราง product\_list

Input: รหัสสินค้า (SKU)

รหัสประเภทการขาย (sales\_id)

ชื่อสินค้า (product\_name)

ราคาสินค้า (product\_price)

ต้นทุนสินค้า (product\_cost)

คำอธิบายสินค้า (product\_description)

Output: ข้อความ (MESSAGE\_TEXT)

Process: นำข้อมูลรหัส SKU มาเช็คว่ามีรหัสสินค้านี้อยู่ในระบบหรือไม่ หากไม่มีรหัสสินค้าในระบบจะเช็คว่า product\_price มากกว่า 0 หรือไม่ หากมากกว่า 0 จะนำข้อมูลสินค้า insert เข้าตาราง product\_list หากน้อยกว่าหรือเท่ากับ 0 จะไม่นำข้อมูลสินค้า insert เข้าตาราง product\_list และแจ้งเตือนรหัส SKU ราคาสินค้า และต้นทุนสินค้าต้องมีค่ามากกว่าหรือเท่ากับ 0 และหากมีรหัส SKU ในระบบจะไม่นำข้อมูลสินค้า insert เข้าตาราง product\_list และแจ้งเตือนรหัส SKU หรือสินค้ามีอยู่ในระบบแล้ว

1. **before\_insert\_daily\_expense**

Description: ใช้ในการเช็คข้อมูลค่าใช้จ่ายก่อน insert ข้อมูลเข้าตาราง daily\_expense

Input: id ของ record (record\_id)

รหัสค่าใช้จ่าย (expense\_id)

รหัสสาขา (branch\_id)

วันที่เกิดค่าใช้จ่าย (daily\_expense\_date)

ราคาค่าใช้จ่าย (daily\_expense\_price)

รายละเอียดของค่าใช้จ่ายที่เกิดขึ้น (daily\_expense\_description)

Output: ข้อความ (MESSAGE\_TEXT)

Process: นำข้อมูล daily\_expense\_price มาเช็ค หาก daily\_expense\_price มากกว่า 0 จะนำข้อมูลค่าใช้จ่าย insert เข้าตาราง daily\_expnse หากน้อยกว่าหรือเท่ากับ 0 จะไม่นำข้อมูลค่าใช้จ่าย insert เข้าตาราง daily\_expense และแจ้งเตือนค่าใช้จ่ายต้องมีค่ามากกว่า 0 กลับไป

# **3. การออกแบบ  Web Application**

## **ผู้ใช้งาน (Users)**

1. ผู้ประกอบการ

นำเข้าข้อมูลจากระบบ POS บันทึกข้อมูล แก้ไขข้อมูล และวิเคราะห์ข้อมูลผลการดำเนินงานรายเดือนและดูต้นทุนและยอดขายจากการดำเนินงาน เพื่อเป็นประโยชน์ในการบริหารและจัดการธุรกิจ

1. ผู้จัดการสาขา หรือ ตัวแทนของแต่ละสาขา

บันทึกข้อมูลการจ่ายเงินของสาขา และบันทึกการเข้ามางานและมาสายของพนักงาน

## **ระบบและขอบเขตการทำงานของ Web Application**

1. **ระบบ Login-Logout**

ให้ผู้จัดการสาขาและเจ้าของร้าน Login เข้ามาในเว็บไซต์เพื่อแก้ไขและบันทึกการทำงานต่างๆในร้านชาบู 3 พี่น้อง เมื่อใช้งานเสร็จเรียบร้อย สามารถกด Logout จากระบบ

1. **การบันทึกรายการค่าใช้จ่ายในร้าน**

ระบบที่ผู้จัดการสาขาเข้ามาบันทึกรายการค่าใช้จ่ายแต่ละประเภทที่เกิดขึ้นกับสาขา เพื่อลดปัญหาที่เกิดขึ้นจากมนุษย์ (Human Error) และติดตามค่าใช้จ่ายที่เกิดขึ้น เว็บไซต์จะให้เลือกสิ่งที่ต้องการบันทึก ให้ไปยัง Choice Box ของรายการที่ต้องการบันทึก เลือก “ค่าใช้จ่าย” หน้าการบันทึกค่าใช้จ่ายจะแสดงผลออกมา จากนั้นให้ผู้เข้าใช้งานเลือกประเภทค่าใช้จ่ายและบันทึกตามข้อมูลที่สอดคล้องกับฐานข้อมูล

1. **การอัพโหลดข้อมูลจากระบบ Loyverse**

เจ้าของร้านค้านำเข้าข้อมูลจาก Loyverse จากนั้นระบบจะทำการบันทึกไฟล์ csv จาก Loyverse เข้ามายังฐานข้อมูล โดยจะแสดงผลไปยังหน้าเว็บไซต์ว่านำเข้าข้อมูลอะไรบ้างเพื่อให้ผู้ใช้งานสามารถตรวจสอบความถูกต้องของข้อมูลได้

1. **การบันทึกข้อมูลพนักงาน**

ผู้จัดการสาขาบันทึกข้อมูลการเข้าออกการทำงานของพนักงาน เว็บไซต์จะแสดงหน้าจอให้เลือกสาขา ชื่อพนักงาน และ Checkbox ว่าขาดงาน มาสายหรือไม่ จากนั้นกดบันทึก ระบบจะเชื่อมต่อกับฐานข้อมูลและบันทึกรายการที่เกิดขึ้น

1. **การสร้างสรุปกราฟผลการดำเนินงาน**

เจ้าของสาขาเลือกเมนูสรุปผลการดำเนินงาน ระบบจะแสดงผล Dashboard แสดงข้อมูลที่เจ้าของสาขาเลือกจากทั้งหมด 4 รายการ ได้แก่

* ภาพรวม
* รายได้
* ค่าใช้จ่าย
* พนักงาน

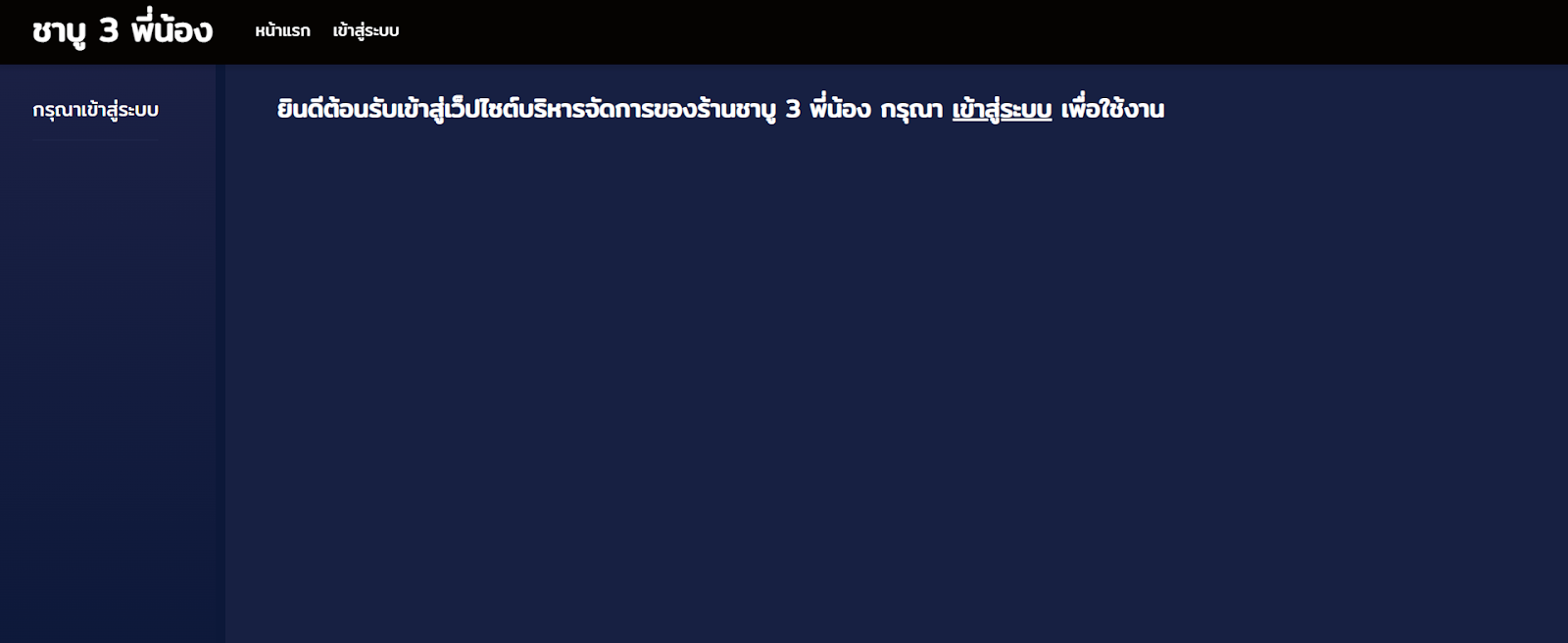
จากนั้นแสดงผลการดำเนินงานแยกตามเดือน โดยจะประยอดรวม ผลการดำเนินงานแยกตามสาขา และผลการดำเนินงานแยกตามวัน

## **ขอบเขตที่นอกเหนือจากการดำเนินงาน**

1. ความสวยงามของ Web Application
2. ความถูกต้องของข้อมูลและความสอดคล้องที่นำเข้าต่อฐานข้อมูล
3. การเพิ่มความปลอดภัยของฐานข้อมูลที่เข้าถึงข้อมูลของร้านชาบู 3 พี่น้อง เนื่องจากมีการใช้ username และ password เพื่อจำกัดความสามารถในหารเข้าถึงของข้อมูล ป้องกันข้อมูลรั่วไหลและข้อมูลผิดพลาด

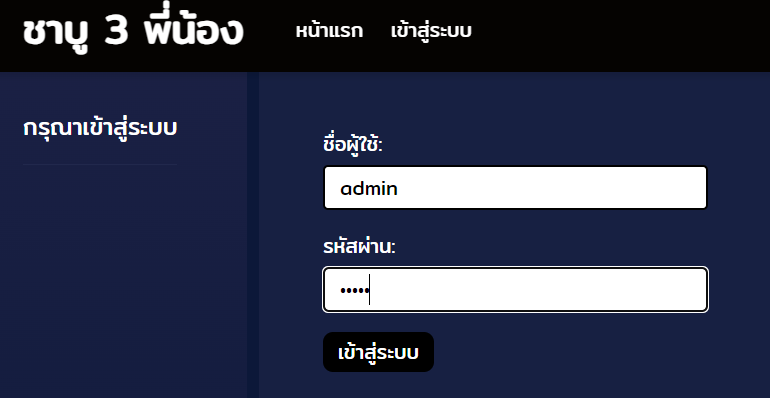
# **4. คู่มือการใช้งานเว็บไซต์**

1. **เข้าสู่เว็บไซต์ จาก python manage.py runserver**

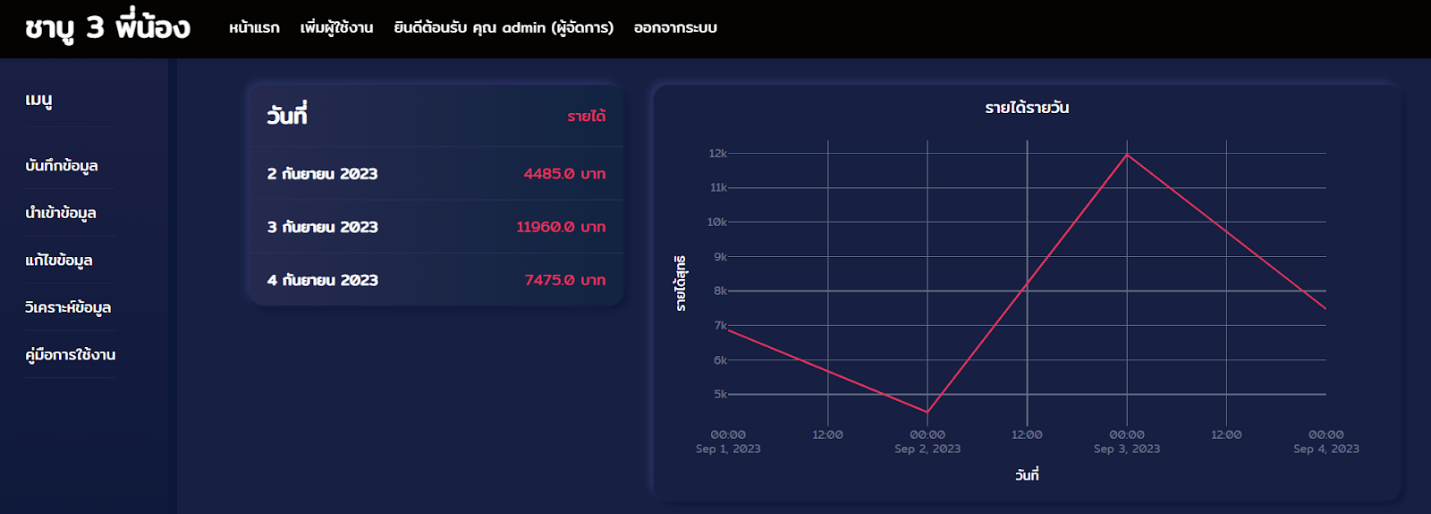


1. **การเข้าสู่ log-in**

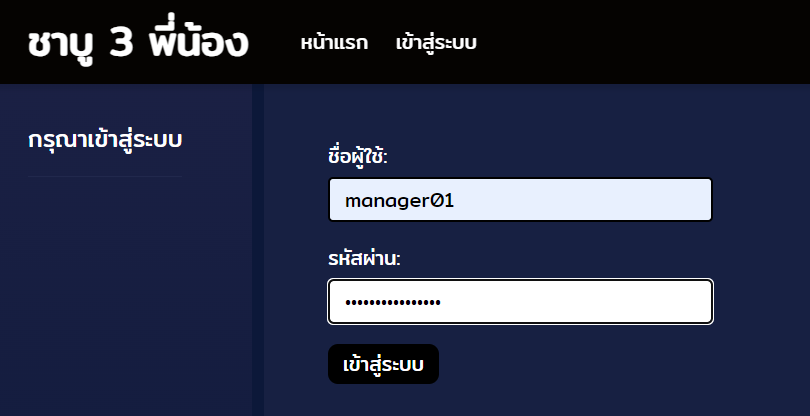
2.1 เข้าสู่ระบบด้วยผู้ใช้ admin และใส่รหัสผ่าน



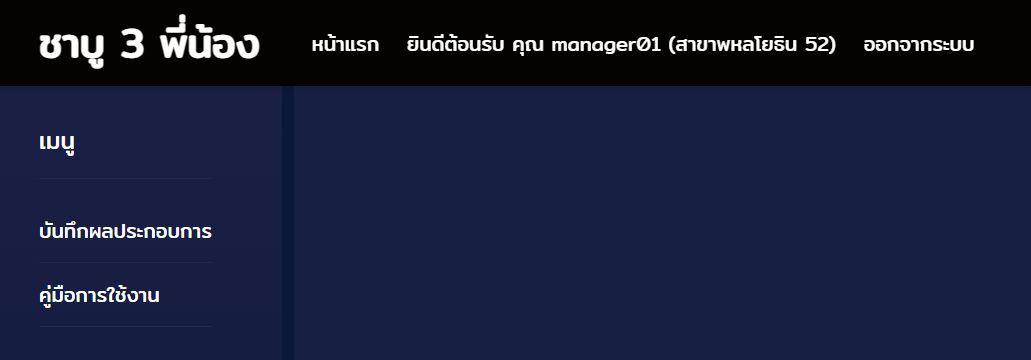
2.2 แสดงหน้าแรกของบัญชี admin



2.3  เข้าสู่ระบบด้วยผู้ใช้ manager ของแต่ละสาขา และใส่รหัสผ่าน

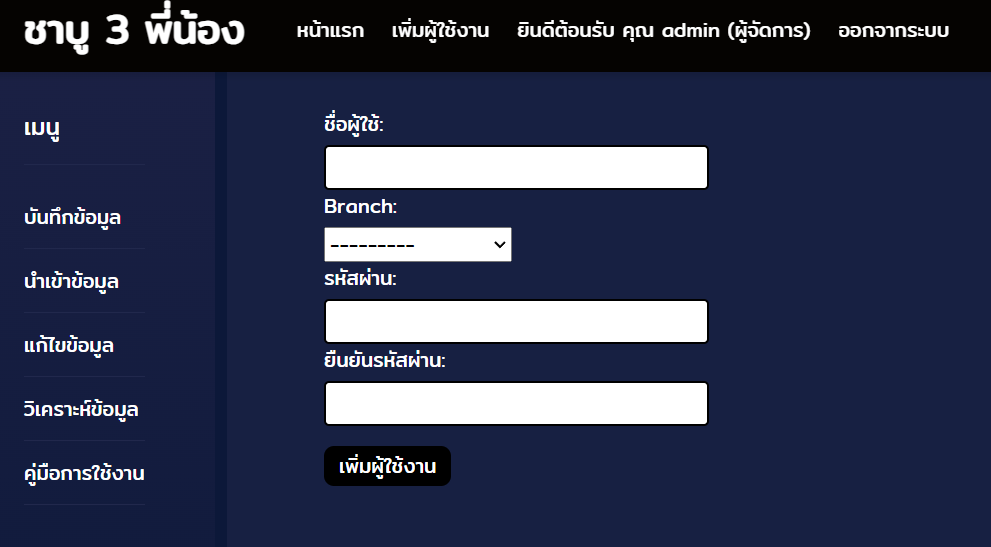


2.4 แสดงหน้าแรกของบัญชีของ Manager

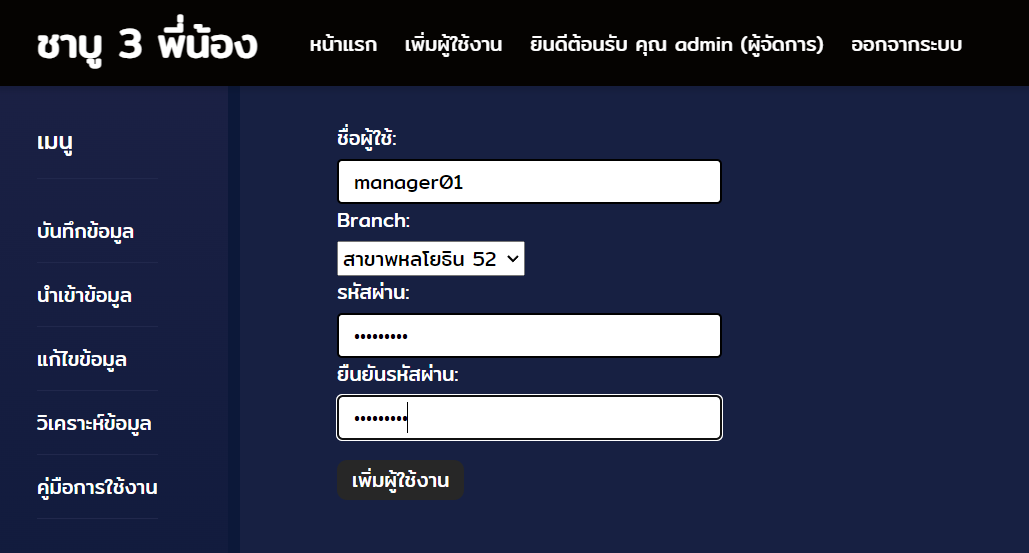


**3. การเพิ่มผู้ใช้งานของแต่ละสาขา**

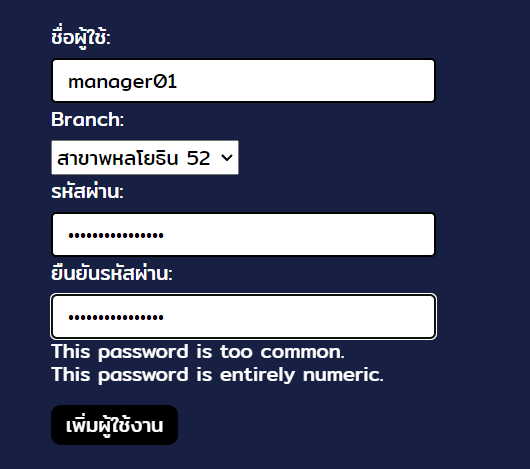
3.1 คลิ๊กไปที่เมนู “เพิ่มผู้ใช้งาน” จะสามารถเพิ่มผู้ใช้งานในแต่ละสาขาได้ ด้วยบัญชี admin



3.2 การเพิ่มผู้ใช้งาน จำเป็นต้องกรอก ชื่อผู้ใช้งาน สาขา และยืนยันรหัสผ่านสำหรับผู้ใช้งานที่เพิ่มใหม่



3.3 หากรหัสผ่านที่ตั้งง่ายเกินไปและมีข้อความขึ้นเตือน

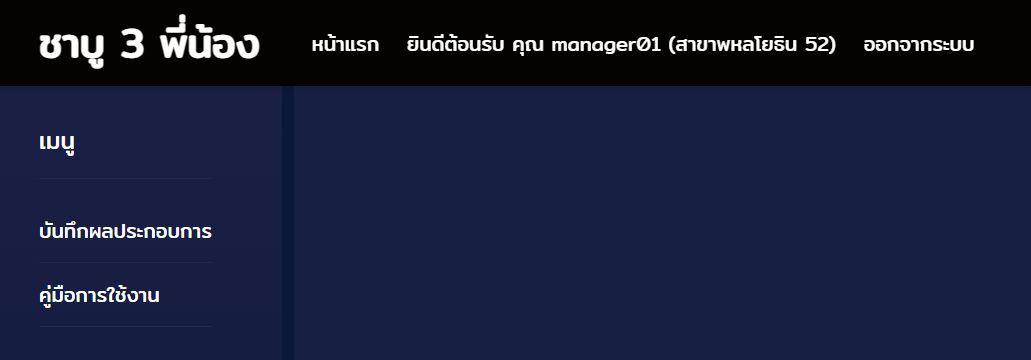


3.4 การบันทึกเสร็จเรียบร้อย

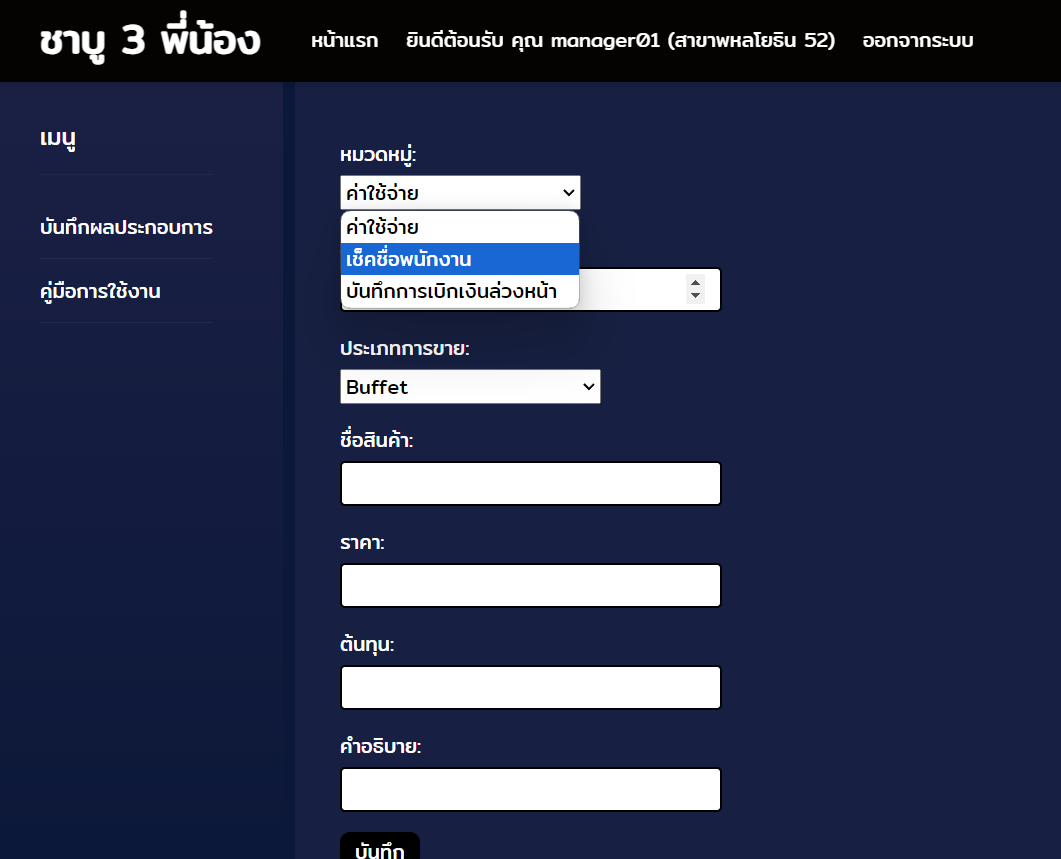


**4. ระบบการบันทึกข้อมูล**

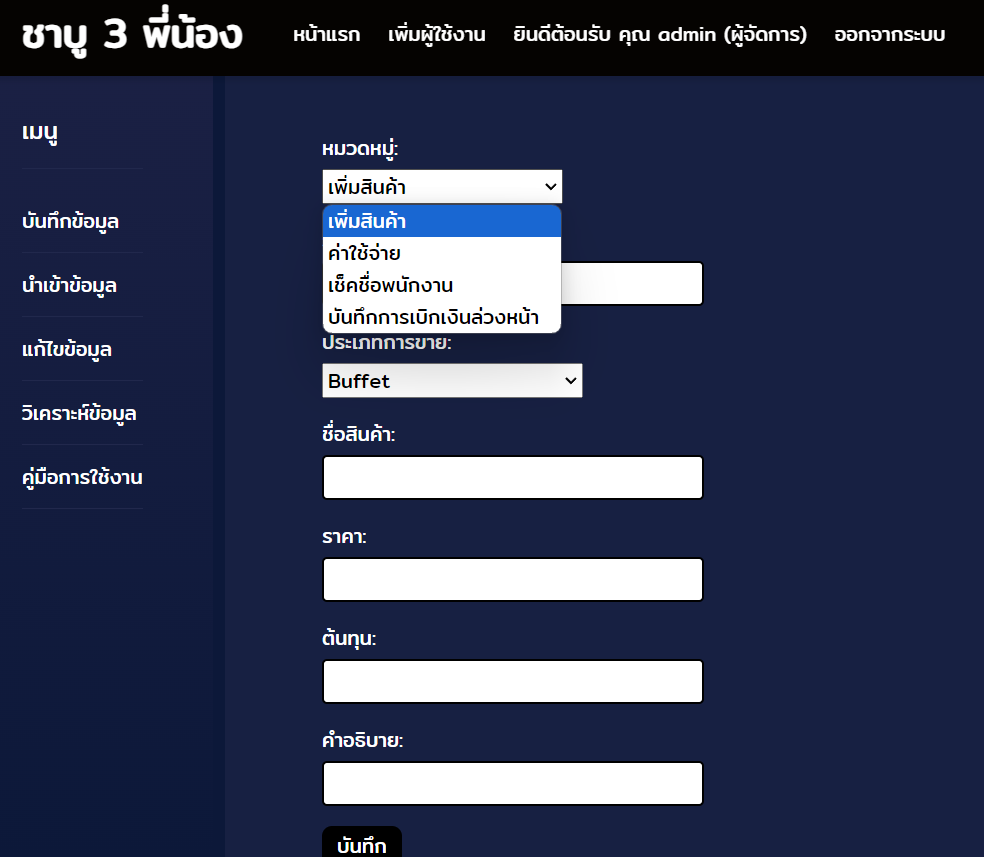
4.1 แสดงหน้าแรกที่บัญชี manager สามารถเข้าถึงข้อมูลได้



4.2 ระบบจะแสดงเฉพาะเมนูที่ Manager มีหน้าที่ ได้แก่บันทึกข้อมูล ค่าใช้จ่าย เช็คชื่อพนัก และ การเบิกเงินล่วงหน้าของพนักงาน



4.3 ในขณะที่ Admin สามารถเข้าถึงข้อมูลการเพิ่มสินค้า ค่าใช้จ่าย เช็คชื่อพนัก และ การเบิกเงินล่วงหน้าของพนักงาน อีกทั้งการแก้ไขและวิเคราะห์ข้อมูล



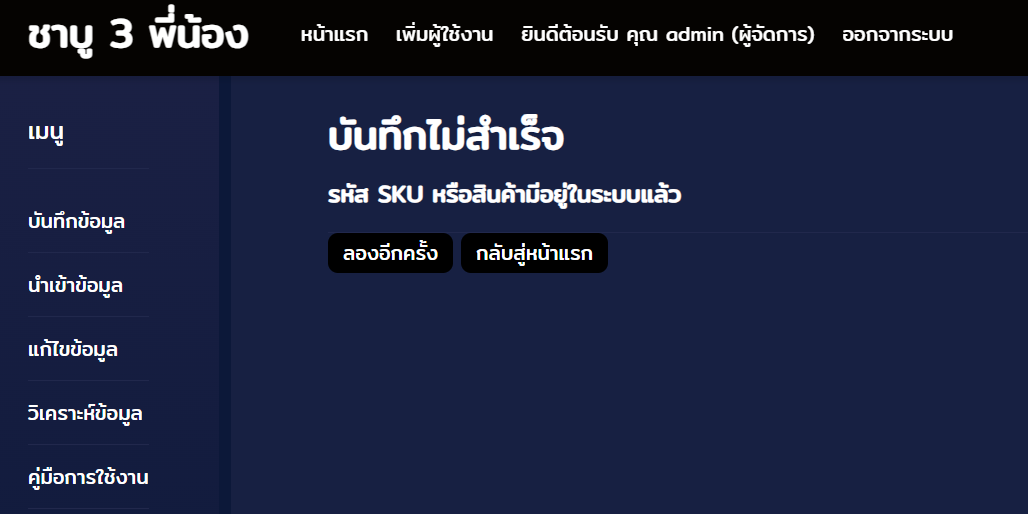
4.4 หากต้องการเพิ่มรายการสำหรับเมนูสินค้า ให้เลือกไปยังหมวดหมู่ “เพิ่มสินค้า” กรอกรหัส SKU  ราคา ต้นทุน คำอธิบาย



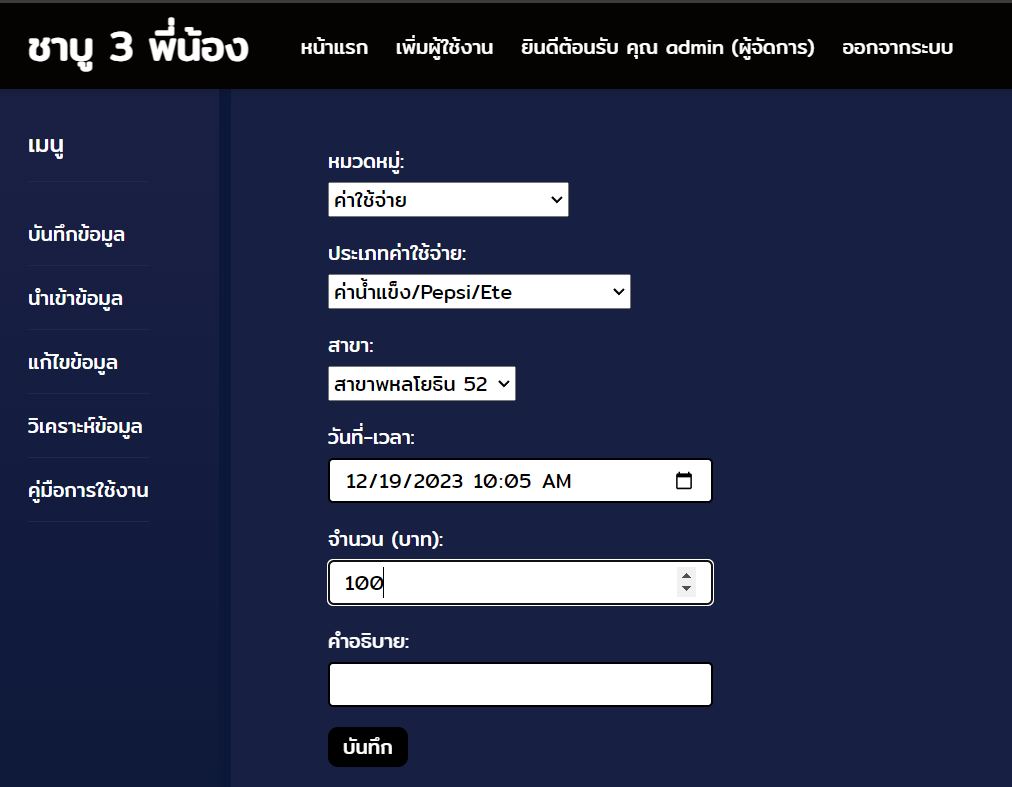
4.5 เมื่อใส่ข้อมูลเสร็จ ให้กดบันทึก หากรายการสินค้าดังกล่าวอยู่แล้วระบบจะทำการแจ้งเตือน ว่ามีสินค้าดังกล่าวอยู่แล้วและให้ผู้ใช้งานลองอีกครั้ง



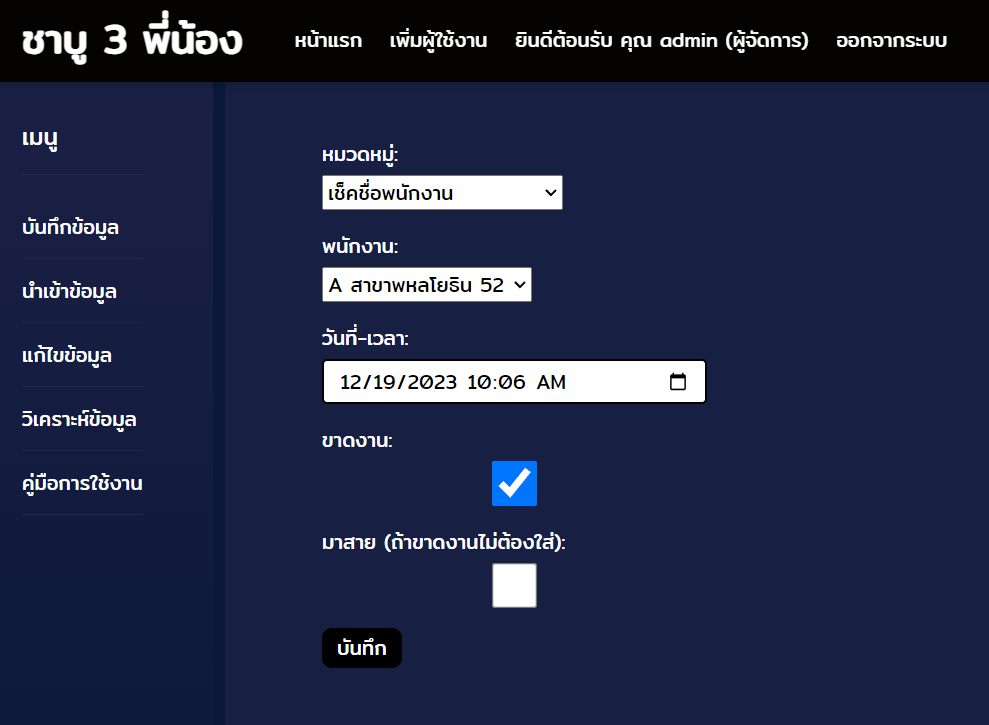
4.6 หากรายการสินค้าดังกล่าวอยู่แล้วระบบจะทำการแจ้งเตือน ว่ามีสินค้าดังกล่าวอยู่แล้วและให้ผู้ใช้งานลองอีกครั้ง



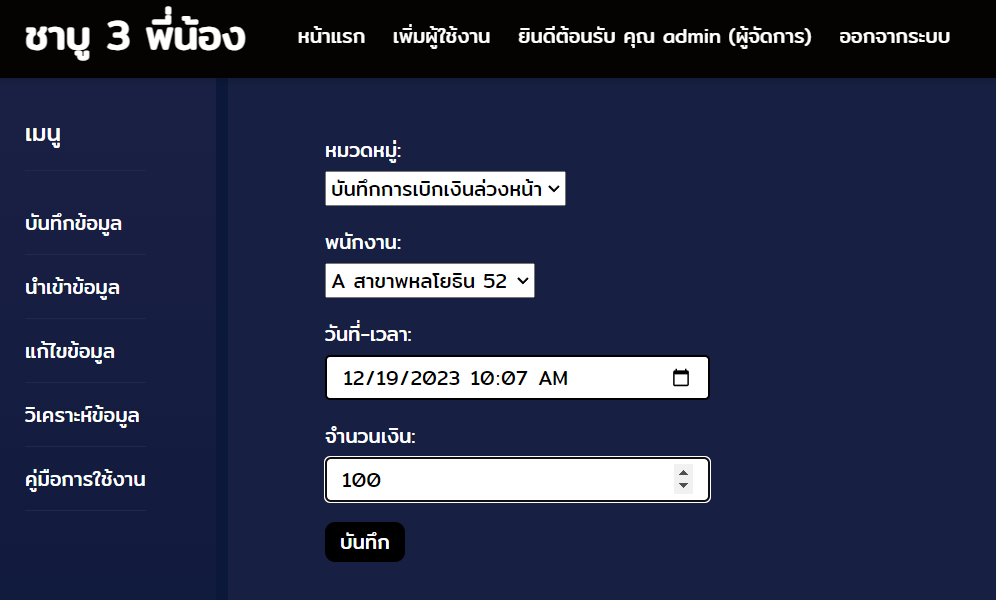
4.7 การบันทึกค่าใช้จ่าย ให้เลือกหมวดหมู่ “ค่าใช้จ่าย” เลือกประเภทค่าใช้จ่าย สาขา ใส่วันที่และเวลา ระบุจำนวน และกดบันทึก ในคำอธิบายสามารถเว้นว่างไว้ได้



4.8 การเช็คชื่อพนักงาน ให้เลือกหมวดหมู่ “เช็คเชื่อพนักงาน” เลือกชื่อพนักงาน วันที่และเวลา ติ๊กการขาดงานหรือมาสาย (หากมี) และกดบันทึก



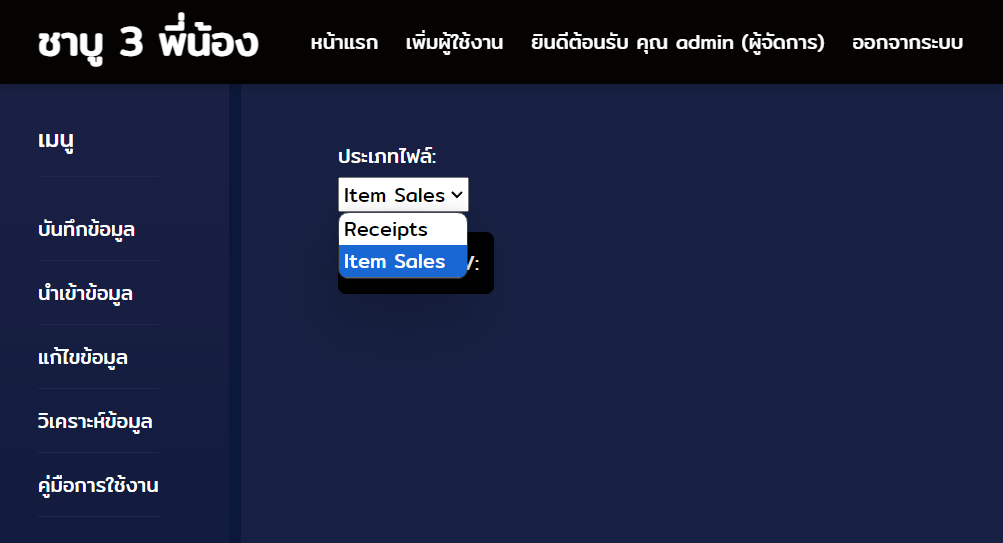
4.9  บันทึกข้อมูลเบิกเงินล่วงหน้าของพนักงาน หากมีค่าเบิกล่วงหน้าติดลบระบบจะทำการแจ้งเตือน



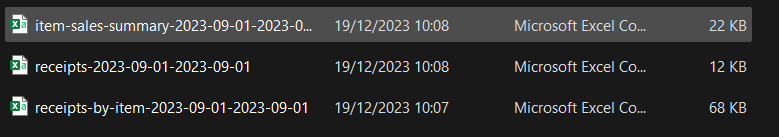


**5. การนำเข้าข้อมูล**

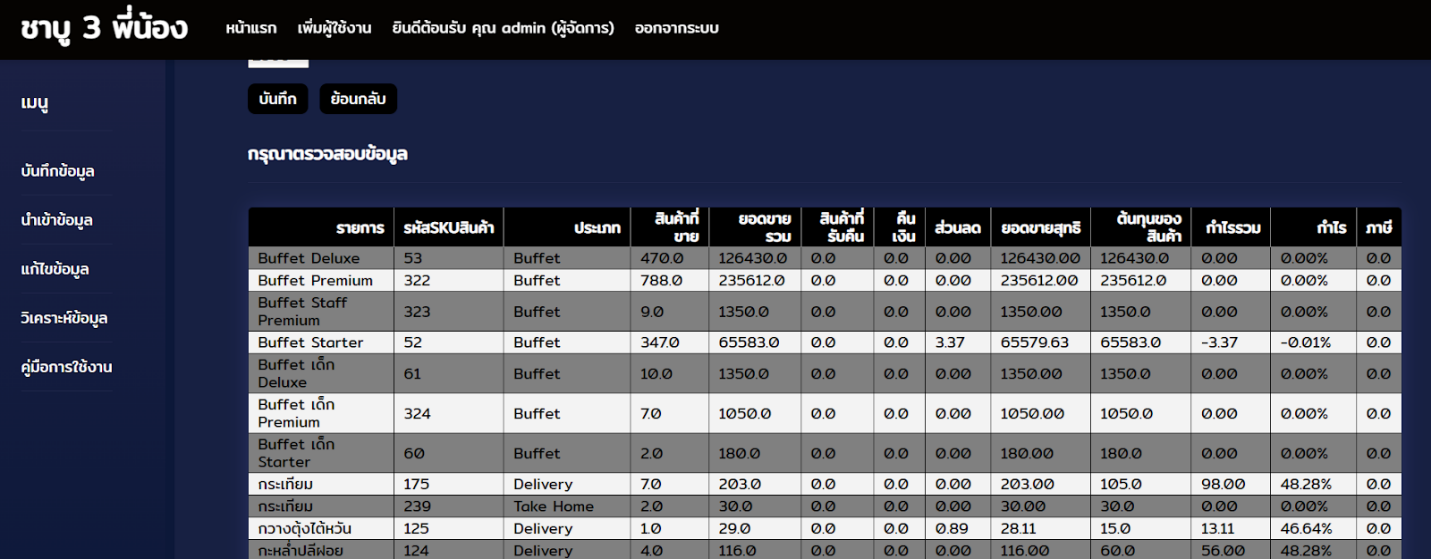
5.1 เลือกนำเข้าข้อมูล Receipt หรือ Item Sales



5.2 หากเลือก “Item Sales” ให้อัปโหลดไฟล์ item-sales



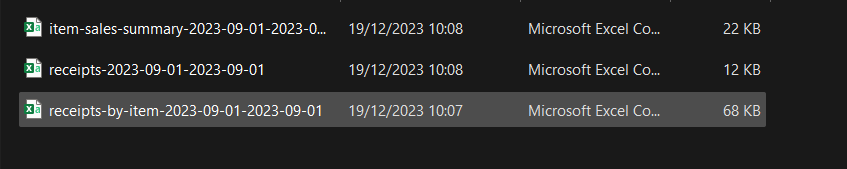
5.3 ตรวจสอบความถูกต้องก่อนบันทึก หากใน Item sales ระบบจะทำสอบถามว่าเราต้องการบันทึก



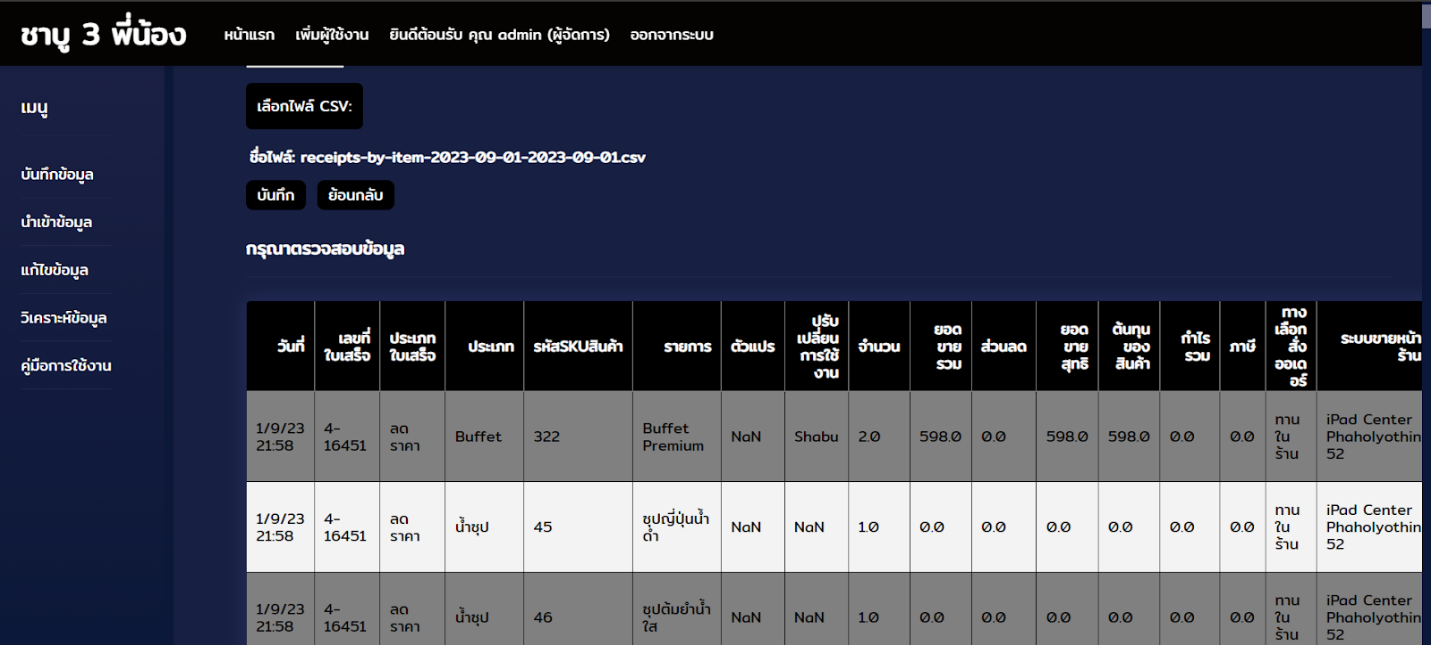


(หมายเหตุ ในตัวอย่างของระบบการ Trigger เป็นไฟล์ข้อมูลใหม่ที่มีการสร้างขึ้นเพื่อยกตัวอย่างเท่านั้น)

5.4  หากเลือก “Receipt” ให้อัปโหลดไฟล์ receipt

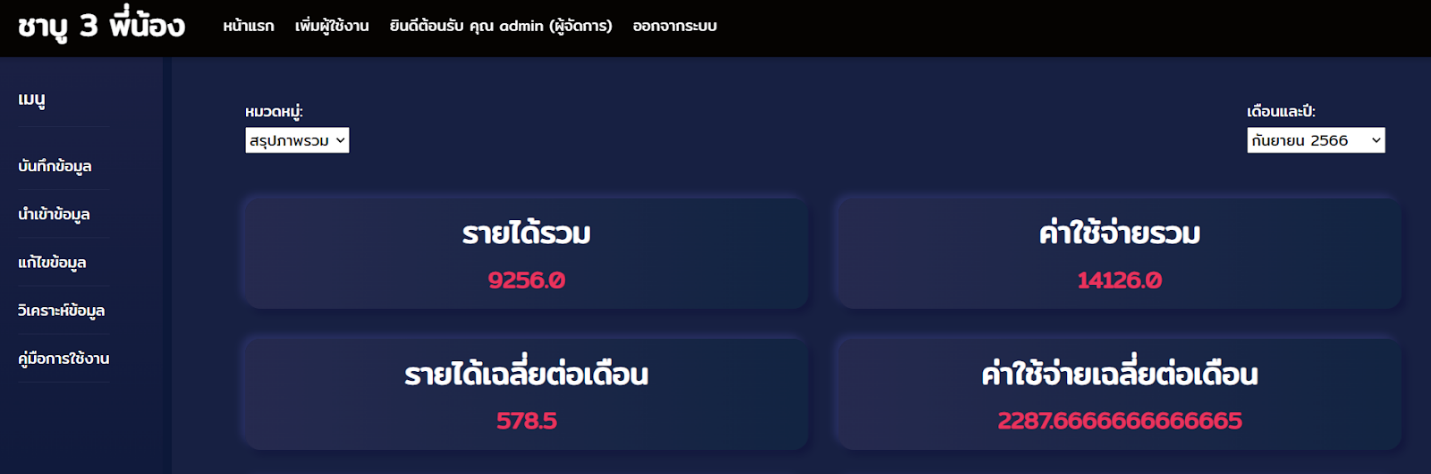


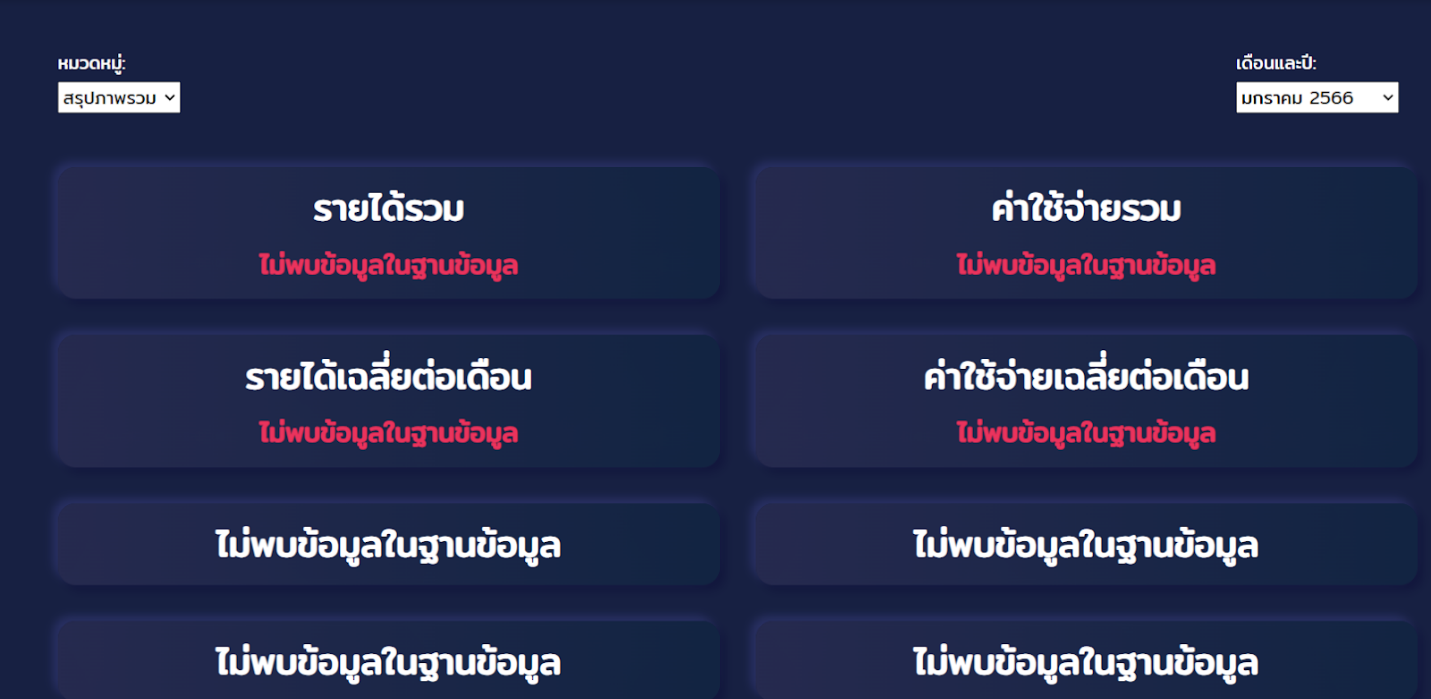
5.5 ตรวจสอบความถูกต้องก่อนบันทึก



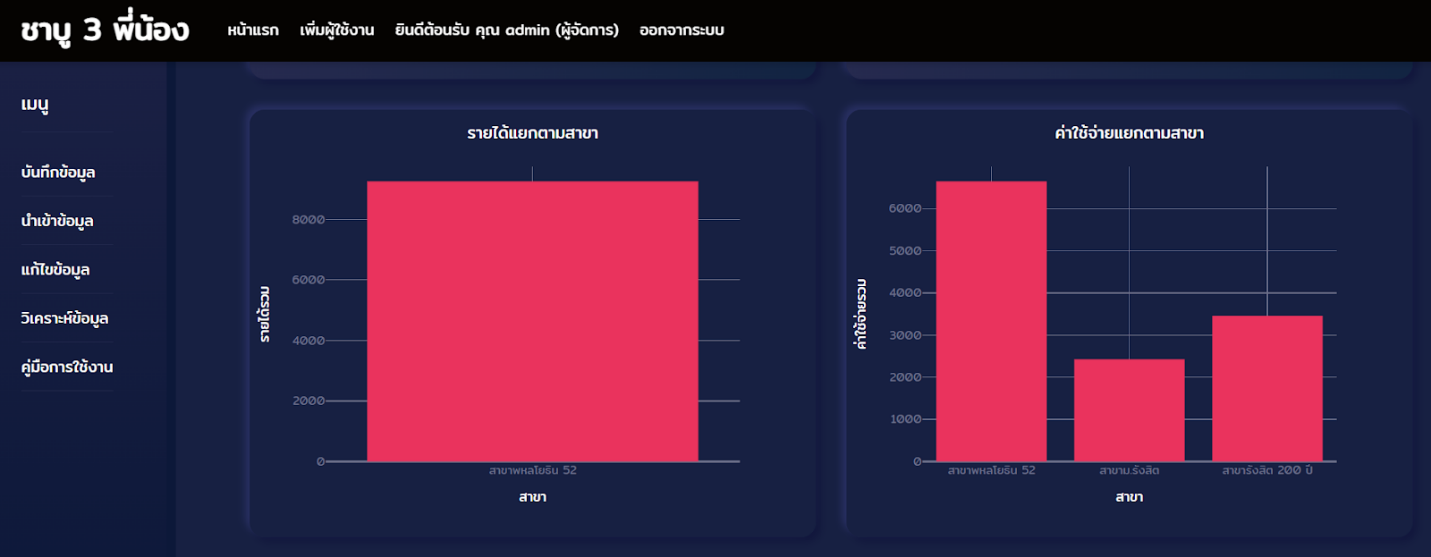
**6. การวิเคราะห์ข้อมูล**

6.1 เลือกหมวดหมู่ “สรุปภาพรวม” และเลือกเดือนที่ต้องการวิเคราะห์ข้อมูล หากเดือนที่เลือกยังไม่มีข้อมูลที่เกี่ยวข้อง ระบบจะแสดงว่า “ไม่พบข้อมูลในฐานข้อมูล”

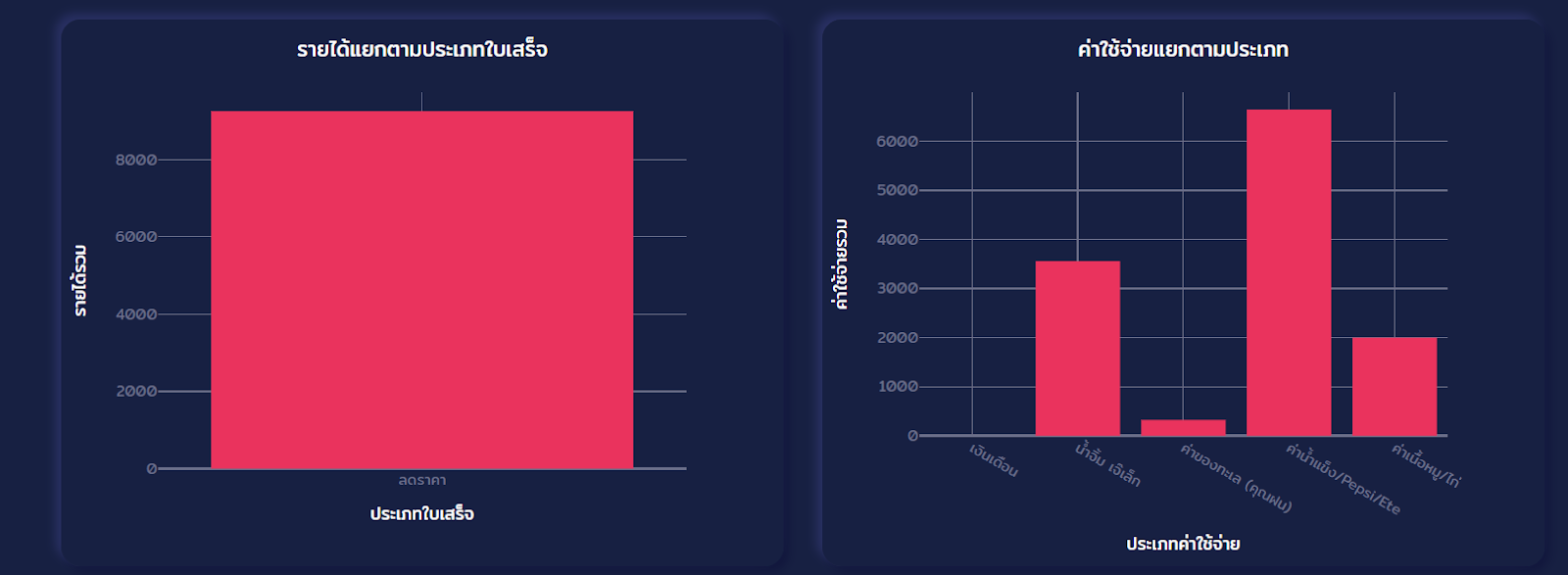




6.2 แสดงผลข้อมูลสรุปรวม กราฟ ของรายได้และค่าใช้จ่าย



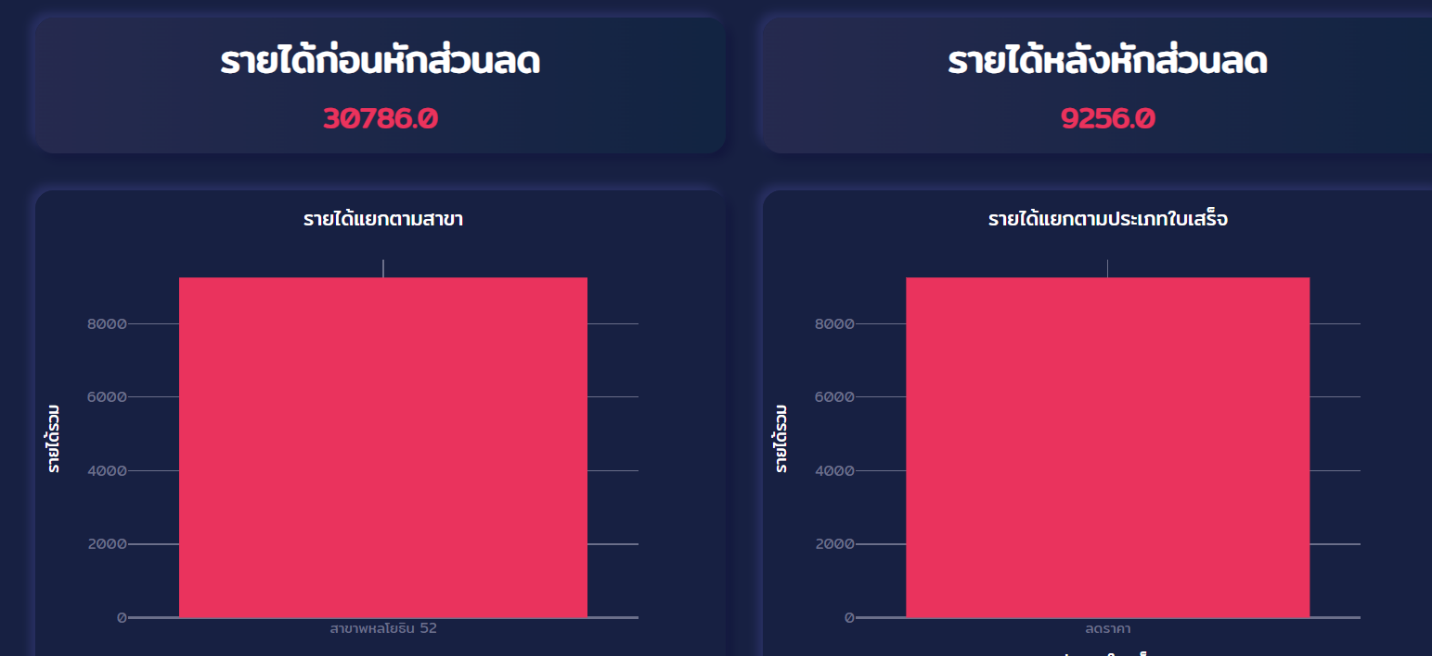
6.3 แสดงข้อมูลรายได้แยกตามประเภทใบเสร็จ (ใบเสร็จประเภท บุฟเฟต์, delivery) และค่าใช้จ่ายแยกประเภทตามหมวดหมู่



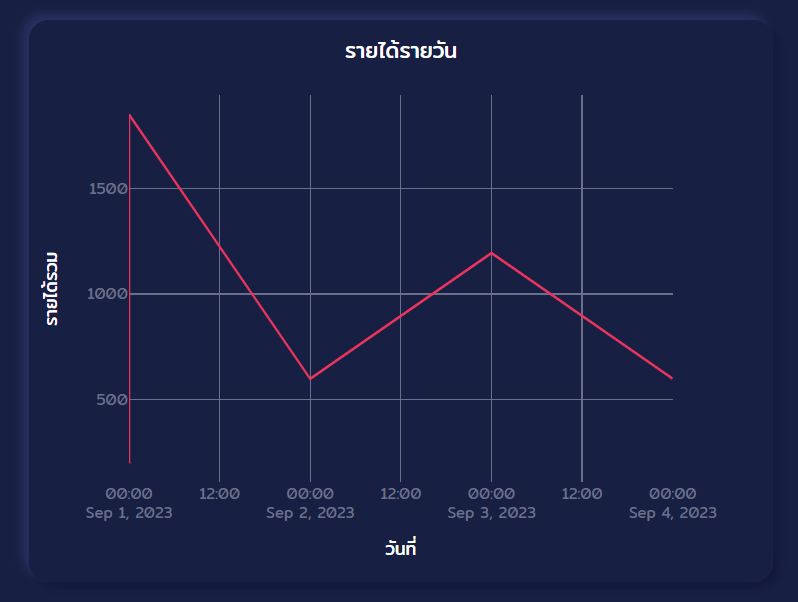
6.4 แสดง Top sales ยอดรวมทั้ง 3 สาขา



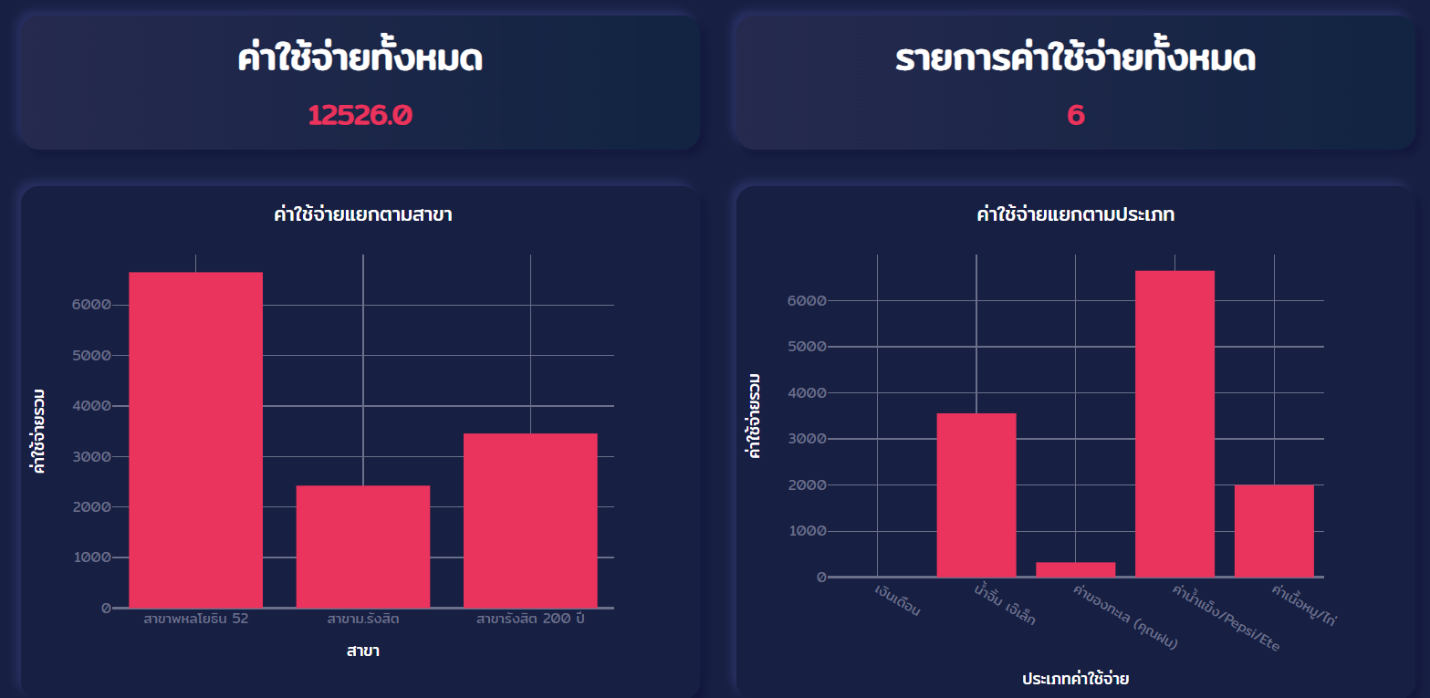
6.5 เลือกหมวดหมู่วิเคราะห์ และเลือกหมวดหมู่ “สรุปรายได้” แสดงข้อมูลยอดรวมรายได้ก่อนหักส่วนลดและหลังหักส่วนลดเป็นตัวเลข และแสดงข้อมูลรายได้แยกตามสาขา และรายได้แยกประเภทตามหมวดหมู่เป็นกราฟแท่ง



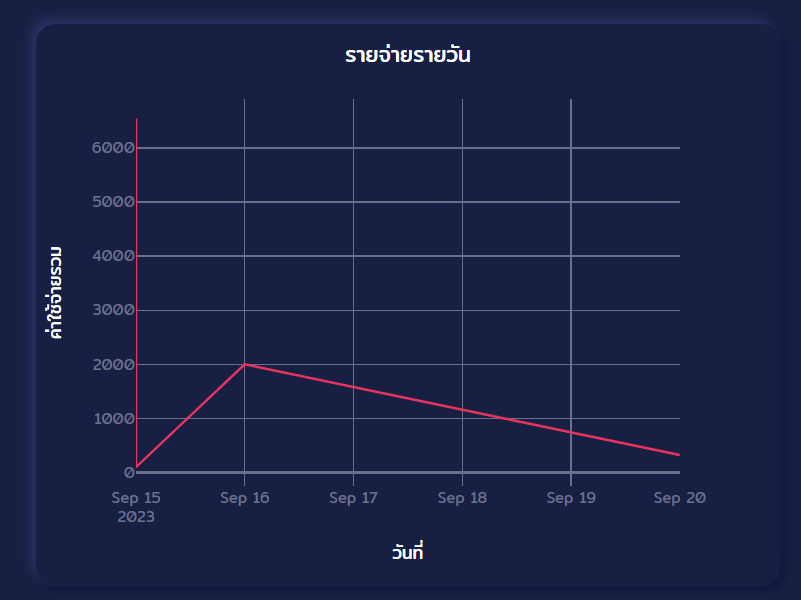
6.6 แสดงข้อมูลสรุปรวม กราฟของรายได้รายวัน



6.7 เลือกหมวดหมู่วิเคราะห์ และเลือกหมวดหมู่ “สรุปค่าใช้จ่าย” แสดงข้อมูลยอดรวมค่าใช้จ่ายทั้งหมดและรายการค่าใช้จ่ายทั้งหมด และแสดงข้อมูลค่าใช้จ่ายแยกตามสาขา และค่าใช้จ่ายแยกประเภทตามหมวดหมู่เป็นกราฟแท่ง



6.8 แสดงข้อมูลสรุปรวม กราฟของค่าใช้จ่ายรายวัน



6.9 เลือกหมวดหมู่วิเคราะห์ และเลือกหมวดหมู่ “สรุปพนักงาน” แสดงข้อมูลรวมพนักงานมาสาย รวมพนักงานไม่มาทำงาน และข้อมูลพนักงานแบ่งแยกตามสาขา



6.10 แสดงพนักงานมาสายแยกรายวันและพนักงานที่ไม่มาทำงานแยกรายวัน



# **5. การสรุปผลและข้อเสนอแนะ**

## **การสรุปผล**

จากการจัดทำเว็ปไซต์ร้านชาบู 3 พี่น้อง พบว่า เว็บไซต์สามารถลดขั้นตอนการวิเคราะห์ข้อมูลผลการดำเนินงาน เพราะเว็บไซต์มีเมนูสำหรับการวิเคราะห์ข้อมูลตัวเลขและกราฟแยกเป็นหมวดหมู่ได้แก่ สรุปภาพรวม, สรุปรายได้, สรุปค่าใช้จ่าย และสรุปพนักงาน รวมถึงมีการวิเคราะห์ต้นทุนจากการดำเนินงาน โดยการดูข้อมูลหน้าการวิเคราะห์ค่าใช้จ่ายที่เกิดขึ้นแยกตามประเภทค่าใช้จ่าย และสรุปรวมค่าใช้จ่ายที่เกิดขึ้นแยกตามสาขา ทำให้ทราบถึงต้นทุนที่เกิดขึ้นจากการดำเนินงาน นอกจากนี้เว็บไซต์ยังแสดงผลสินค้าขายดีในหน้าการวิเคราะห์ของหมวดหมู่สรุปภาพรวมในรูปแบบกราฟ Top 10 sales ทำให้ทราบรายการขายดีประจำเดือน

ในส่วนการจัดการของข้อมูล ระบบมีตัวเลือกให้สามารถบันทึกข้อมูลที่เกิดขึ้นโดยใช้ช่วงเวลาในการบันทึกจึงสามารถติดตามผลได้อย่างอย่างง่าย เป็นให้สามารถติดตามผลลัพธ์การดำเนินงานต่าง ๆ ได้ และเมื่อมีการบันทึกข้อมูลในระบบ จะมีตารางแสดงผลข้อมูลที่ได้อัพโหลดไว้ ให้ตรวจสอบความถูกต้องก่อนบันทึกข้อมูล เป็นผลให้ข้อมูลมีความถูกต้องและสอดคล้องกันอย่างมีความสัมพันธ์ (ERD)

## **ข้อเสนอแนะ**

1. เพิ่มการบันทึกข้อมูลที่เกี่ยวข้องกับ Supplier ให้ทราบถึงราคาต้นทุนวัตถุดิบอย่างละเอียด
2. เพิ่มฟังก์ชันการบันทึกค่าวัตถุดิบที่สั่งมาจาก Supplier แยกออกมาจากการบันทึกค่าใช้จ่ายอื่น ๆ เพื่อให้สามารถตรวจสอบความถูกต้องของรายการสินค้า ปริมาณ และมูลค่าได้แม่นยำยิ่งขึ้น
3. เพิ่มการบันทึกต้นทุนของรายการสินค้าบุฟเฟ่ต์ เพื่อให้การจัดทำสรุปผลการดำเนินงานกำไรขาดทุนครบถ้วนถูกต้อง
4. ร้านชาบู 3 พี่น้องควรมีการแบ่งแยกประเภทให้มีแต่ละประเภทมีขนาดเล็กลง เนื่องจากสำหรับสินค้าที่ขายสำหรับการสั่งซื้อแบบบุฟเฟ่ต์จะถูกจัดให้อยู่ในกลุ่ม “เนื้อสัตว์และอื่น ๆ” ทำให้การวิเคราะห์ต้นทุนต่าง ๆ ทำได้ยาก

# **บรรณานุกรม**

Minjy Na Nakhon. (12 ตุลาคม 2565). สรุปเทรนด์ ธุรกิจ บุฟเฟ่ต์ จากงานเสวนา “Thailand Buffet Network 2022” ttps://business.hungryhub.com/restaurant-tips/เทรนด์-ธุรกิจ- บุฟเฟ่ต์/

Withoutcoffee Icantdedev. (17 พฤศจิกายน 2566). พัฒนาเว็บด้วย Django Framework (Python) ฉบับเต็มปี 2023 https://devhub.in.th/blog/django-python

# **ภาคผนวก**

USE shabu3peenong;

CREATE OR REPLACE VIEW home\_daily AS

SELECT

CAST(receipt\_date as date) AS `วันที่`,

SUM(receipt\_total) AS `รายได้สุทธิรวม`

FROM receipt

GROUP BY CAST(receipt\_date as date)

ORDER BY CAST(receipt\_date as date);

DELIMITER //

CREATE PROCEDURE InsertProduct

(IN SKU INT, IN sale\_id INT, IN sales VARCHAR(50), IN product\_name VARCHAR(50), IN product\_price FLOAT,

IN product\_cost FLOAT, IN product\_description VARCHAR(255))

BEGIN

    DECLARE sales\_name\_id INT;

    IF product\_description = '' OR product\_description = ' ' THEN

        SET product\_description = NULL;

    END IF;

    IF sale\_id IS NULL THEN

        SELECT sales\_id INTO sales\_name\_id FROM sales\_type WHERE sales\_name = sales;

    END IF;

    INSERT INTO product\_list (SKU, sales\_id, product\_name, product\_price, product\_cost, product\_description)

    VALUES (SKU, COALESCE(sales\_name\_id, sale\_id), product\_name, product\_price, product\_cost, product\_description);

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER before\_insert\_product\_list

BEFORE INSERT ON product\_list

FOR EACH ROW

BEGIN

    DECLARE exists\_SKU INT;

    SELECT COUNT(SKU) INTO exists\_SKU

    FROM product\_list

    WHERE SKU = NEW.SKU;

    IF exists\_SKU > 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = 'รหัส SKU หรือสินค้ามีอยู่ในระบบแล้ว';

    ELSE

        IF NEW.SKU < 0 OR NEW.product\_price < 0 OR NEW.product\_cost < 0 THEN

            SIGNAL SQLSTATE '45000'

            SET MESSAGE\_TEXT = 'รหัส SKU ราคาสินค้า และต้นทุนสินค้าต้องมีค่ามากกว่าหรือเท่ากับ 0';

        END IF;

    END IF;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE InsertExpense

(IN expense\_id INT, IN branch\_id INT, IN daily\_expense\_date DATETIME, IN daily\_expense\_price FLOAT, IN daily\_expense\_description VARCHAR(255))

BEGIN

    IF daily\_expense\_description = '' OR daily\_expense\_description = ' ' THEN

        SET daily\_expense\_description = NULL;

    END IF;

    INSERT INTO daily\_expense (expense\_id, branch\_id, daily\_expense\_date, daily\_expense\_price, daily\_expense\_description)

    VALUES (expense\_id, branch\_id, daily\_expense\_date, daily\_expense\_price, daily\_expense\_description);

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER before\_insert\_daily\_expense

BEFORE INSERT ON daily\_expense

FOR EACH ROW

BEGIN

    IF NEW.daily\_expense\_price <= 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'ค่าใช้จ่ายต้องมีค่ามากกว่า 0';

    END IF;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE InsertActivity

(IN is\_superuser BOOLEAN, IN branch INT, IN employee INT, IN activity\_date DATETIME, IN activity\_absent TINYINT(1), IN activity\_late TINYINT(1))

BEGIN

    DECLARE employee\_branch\_id INT;

    IF is\_superuser = TRUE THEN

        SELECT branch\_id INTO employee\_branch\_id FROM employee WHERE employee\_id = employee;

        SET branch = NULL;

    END IF;

    INSERT INTO activity (branch\_id, employee\_id, activity\_date, activity\_absent, activity\_late)

    VALUES (COALESCE(employee\_branch\_id, branch), employee, activity\_date, activity\_absent, activity\_late);

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE UpdateAdvancedPay (IN employee INT, IN pay\_date DATETIME, IN amount FLOAT)

BEGIN

    IF amount > 0 THEN

UPDATE activity

SET advanced\_pay = advanced\_pay + amount

WHERE activity.employee\_id = employee AND DATE\_FORMAT(activity\_date, '%Y-%m-%d') = DATE\_FORMAT(pay\_date, '%Y-%m-%d');

    ELSE

SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = 'จำนวนเงินเบิกล่วงหน้าต้องมีค่ามากกว่า 0';

    END IF;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE InsertReceipt

(IN receipt\_id VARCHAR(10), IN branch VARCHAR(50), IN receipt\_date DATETIME, IN receipt\_type VARCHAR(20), IN receipt\_order VARCHAR(20),

 IN receipt\_system VARCHAR(50), IN receipt\_cashier VARCHAR(50), IN receipt\_total FLOAT, IN receipt\_discount FLOAT, IN receipt\_net FLOAT,

 IN receipt\_customer\_name VARCHAR(50), IN receipt\_customer\_contact VARCHAR(50), IN receipt\_comment VARCHAR(255), IN receipt\_status VARCHAR(20))

BEGIN

    DECLARE d\_branch\_id INT;

    SET d\_branch\_id = CASE

        WHEN TRIM(SUBSTRING\_INDEX(SUBSTRING\_INDEX(branch, '|', -1), '|', 1)) = 'สาขาพหลโยธิน 52' THEN 1

        WHEN TRIM(SUBSTRING\_INDEX(SUBSTRING\_INDEX(branch, '|', -1), '|', 1)) = 'สาขาม.รังสิต' THEN 2

        WHEN TRIM(SUBSTRING\_INDEX(SUBSTRING\_INDEX(branch, '|', -1), '|', 1)) = 'สาขารังสิต 200 ปี' THEN 3

        ELSE NULL

    END;

    INSERT INTO receipt (receipt\_id, branch\_id, receipt\_date, receipt\_type, receipt\_order,

    receipt\_system, receipt\_cashier, receipt\_total, receipt\_discount, receipt\_net,

    receipt\_customer\_name, receipt\_customer\_contact, receipt\_comment, receipt\_status)

    VALUE (receipt\_id, d\_branch\_id, receipt\_date, receipt\_type, receipt\_order, receipt\_system, receipt\_cashier,

            receipt\_total, receipt\_discount, receipt\_net, receipt\_customer\_name, receipt\_customer\_contact,

            receipt\_comment, receipt\_status);

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE InsertReceiptByProduct (IN SKU INT, IN receipt\_id VARCHAR(10), IN rbp\_quantity INT, IN rbp\_discount\_by\_item FLOAT)

BEGIN

INSERT INTO receipt\_by\_product (SKU, receipt\_id, rbp\_quantity, rbp\_discount\_by\_item)

    VALUE (SKU, receipt\_id, rbp\_quantity, rbp\_discount\_by\_item);

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE InsertItemSales

(IN SKU INT, IN item\_sales\_month INT, IN item\_sales\_year INT, IN item\_sales\_quantity INT, IN item\_sales\_refund INT, IN item\_sales\_refund\_price FLOAT)

BEGIN

    INSERT INTO item\_sales (SKU, item\_sales\_month, item\_sales\_year, item\_sales\_quantity, item\_sales\_refund, item\_sales\_refund\_price)

    VALUES (SKU, item\_sales\_month, item\_sales\_year, item\_sales\_quantity, item\_sales\_refund, item\_sales\_refund\_price);

END //

DELIMITER ;

-- ------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE OR REPLACE VIEW summary\_view AS

SELECT

    month\_year,

    SUM(total\_expense) AS total\_expense,

    SUM(avg\_expense)  AS avg\_expense,

    SUM(total\_income) AS total\_income,

    SUM(avg\_income) AS avg\_income

FROM (

    SELECT

        DATE\_FORMAT(daily\_expense\_date, '%Y-%m') AS month\_year,

        SUM(daily\_expense\_price) AS total\_expense,

        AVG(daily\_expense\_price) AS avg\_expense,

        0 AS total\_income,

        0 AS avg\_income

    FROM

        daily\_expense

    GROUP BY

        month\_year

    UNION

    SELECT

        DATE\_FORMAT(receipt\_date, '%Y-%m') AS month\_year,

        0 AS total\_expense,

        0 AS avg\_expense,

        SUM(receipt\_net) AS total\_income,

        AVG(receipt\_net) AS avg\_income

    FROM

        receipt

    GROUP BY

        Month\_year

UNION

    SELECT

        DATE\_FORMAT(daily\_expense\_date, '%Y-%m') AS month\_year,

        SUM(daily\_expense\_price) AS total\_expense,

        AVG(daily\_expense\_price) AS avg\_expense,

        0 AS total\_income,

        0 AS avg\_income

    FROM

        daily\_expense

    GROUP BY

        month\_year

    UNION

    SELECT

        DATE\_FORMAT(activity\_date, '%Y-%m') AS month\_year,

        350 \* SUM(NOT activity\_absent) - 50 \* SUM(activity\_late) - SUM(advanced\_pay) AS total\_expense,

        350 \* AVG(NOT activity\_absent) - 50 \* AVG(activity\_late) - AVG(advanced\_pay) AS avg\_expense,

        0 AS total\_income,

        0 AS avg\_income

    FROM

        activity

    GROUP BY

        Month\_year

) AS subquery

GROUP BY month\_year;

CREATE VIEW total\_income\_by\_branch AS

SELECT

DATE\_FORMAT(receipt\_date, '%Y-%m') AS month\_year,

b.branch\_name,

SUM(r.receipt\_net) AS total\_income

FROM receipt r

INNER JOIN branch b ON r.branch\_id = b.branch\_id

GROUP BY month\_year,  b.branch\_name;

CREATE OR REPLACE VIEW total\_expense\_by\_branch AS

SELECT

month\_year,

b.branch\_name,

    SUM(total\_advanced\_pay) + SUM(total\_daily\_expense\_price) AS total\_expense

FROM (

SELECT

DATE\_FORMAT(activity\_date, '%Y-%m') AS month\_year,

branch\_id AS branch,

SUM(advanced\_pay) AS total\_advanced\_pay,

0 AS total\_daily\_expense\_price

FROM activity

GROUP BY month\_year, branch\_id

UNION

SELECT

DATE\_FORMAT(daily\_expense\_date, '%Y-%m') AS month\_year,

branch\_id AS branch,

0 AS total\_advanced\_pay,

SUM(daily\_expense\_price) AS total\_daily\_expense\_price

FROM daily\_expense

GROUP BY month\_year, branch\_id

) AS subquery

INNER JOIN branch b ON branch\_id = branch

GROUP BY month\_year, branch;

CREATE VIEW total\_income\_by\_type AS

SELECT

DATE\_FORMAT(receipt\_date, '%Y-%m') AS month\_year,

receipt\_type,

SUM(receipt\_net) AS total\_income

FROM receipt

GROUP BY month\_year, receipt\_type;

CREATE OR REPLACE VIEW total\_expense\_by\_type AS

SELECT

month\_year,

expense\_name,

    SUM(total\_advanced\_pay) + SUM(total\_daily\_expense\_price) AS total\_expense

FROM (

SELECT

DATE\_FORMAT(activity\_date, '%Y-%m') AS month\_year,

'เงินเดือน' AS expense\_name,

SUM(advanced\_pay) AS total\_advanced\_pay,

0 AS total\_daily\_expense\_price

FROM activity

GROUP BY month\_year

UNION

SELECT

DATE\_FORMAT(de.daily\_expense\_date, '%Y-%m') AS month\_year,

et.expense\_name AS expense\_name,

0 AS total\_advanced\_pay,

SUM(de.daily\_expense\_price) AS total\_daily\_expense\_price

FROM daily\_expense de

INNER JOIN expense\_type et ON et.expense\_id = de.expense\_id

GROUP BY month\_year, et.expense\_name

) AS subquery

GROUP BY month\_year, expense\_name;

CREATE OR REPLACE VIEW top\_ten\_sales AS

SELECT

CONCAT(its.item\_sales\_year-543, '-', LPAD(its.item\_sales\_month, 2, '0')) AS month\_year,

    pl.product\_name,

    SUM(its.item\_sales\_quantity) AS total\_sales

FROM item\_sales its

INNER JOIN product\_list pl ON pl.SKU = its.SKU

GROUP BY month\_year, pl.product\_name

ORDER BY total\_sales DESC

LIMIT 10;

CREATE OR REPLACE VIEW total\_income\_before\_after\_discount AS

SELECT

DATE\_FORMAT(receipt\_date,  '%Y-%m') AS month\_year,

SUM(receipt\_total) AS total\_income,

SUM(receipt\_net) AS total\_net\_income

FROM receipt

GROUP BY month\_year;

CREATE OR REPLACE VIEW total\_income\_by\_day AS

SELECT

DATE\_FORMAT(receipt\_date, '%Y-%m') AS month\_year,

receipt\_date,

SUM(receipt\_net) AS total\_income

FROM receipt

GROUP BY month\_year, receipt\_date

ORDER BY receipt\_date;

CREATE OR REPLACE VIEW total\_expense\_and\_count\_by\_month AS

SELECT

DATE\_FORMAT(daily\_expense\_date, '%Y-%m') AS month\_year,

SUM(daily\_expense\_price) AS total\_expense,

COUNT(record\_id) AS count

FROM daily\_expense

GROUP BY month\_year;

CREATE OR REPLACE VIEW total\_expense\_by\_day AS

SELECT

DATE\_FORMAT(daily\_expense\_date, '%Y-%m') AS month\_year,

daily\_expense\_date,

SUM(daily\_expense\_price) AS total\_expense

FROM daily\_expense

GROUP BY month\_year, daily\_expense\_date

ORDER BY daily\_expense\_date;

CREATE VIEW total\_emp\_late\_absent AS

SELECT

DATE\_FORMAT(activity\_date, '%Y-%m') AS month\_year,

SUM(activity\_late) AS count\_late,

    SUM(activity\_absent) AS count\_absent

FROM activity

GROUP BY month\_year;

CREATE VIEW total\_emp\_late\_absent\_by\_branch AS

SELECT

DATE\_FORMAT(activity\_date, '%Y-%m') AS month\_year,

b.branch\_name,

SUM(a.activity\_late) AS count\_late,

    SUM(a.activity\_absent) AS count\_absent

FROM activity a

INNER JOIN branch b ON a.branch\_id = b.branch\_id

GROUP BY month\_year, b.branch\_name

ORDER BY month\_year, SUM(a.activity\_late) DESC;

CREATE VIEW total\_emp\_late\_absent\_by\_day AS

SELECT

DATE\_FORMAT(activity\_date, '%Y-%m') AS month\_year,

DATE(activity\_date) AS by\_date,

SUM(activity\_late) AS count\_late,

    SUM(activity\_absent) AS count\_absent

FROM activity

GROUP BY month\_year, by\_date

ORDER BY by\_date;