



รายงาน

เว็บไซต์เพื่อการจัดการข้อมูลสำหรับร้านค้าชาบู 3 ฟีน้อง

จัดทำโดย

กวีพัฒน์	ภควัตสิริกุล	6342002026
ชนกันต์	กนกมณีรัตน์	6342015226
ปริญญญา	จารุอรียานนท์	6342057626
ณัฐนิชา	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 และที่สำคัญคือจะช่วยแก้ไขปัญหาของร้าน Shabu 3 Peenong ได้ หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด คณะผู้จัดทำขอน้อมรับไว้และขออภัยมา ณ ที่นี้

คณะผู้จัดทำ

สารบัญ

1. บทนำ	1
หัวข้อที่ศึกษา.....	1
ที่มาและความสำคัญของปัญหา.....	1
ปัญหาที่ต้องการแก้ไข.....	1
แนวทางการแก้ไข	2
ประโยชน์ที่คาดว่าจะได้รับ	2
2. การออกแบบฐานข้อมูล	3
เครื่องมือที่ใช้.....	3
การออกแบบ Entity Relationship Diagram	3
คำอธิบายตัวแปร	5
Programmable Object.....	11
3. การออกแบบ Web Application	22
ผู้ใช้งาน (Users).....	22
ระบบและขอบเขตการทำงานของ Web Application.....	22
ขอบเขตที่นอกเหนือจากการดำเนินงาน	23
4. คู่มือการใช้งานเว็บไซต์	24
5. การสรุปผลและข้อเสนอแนะ	42
การสรุปผล.....	42
ข้อเสนอแนะ.....	42
บรรณานุกรม	43
ภาคผนวก	44

URL GitHub: https://github.com/patinya2001/Database_Programming.git

1.บทนำ

หัวข้อที่ศึกษา

ระบบการวิเคราะห์และรายงานผลประกอบการร้านชาบู 3 ฟีน้องผ่านเว็บไซต์เพื่อการจัดการ

ที่มาและความสำคัญของปัญหา

ในปัจจุบันการแข่งขันในตลาดธุรกิจชาบูบุฟเฟ่ต์ในประเทศไทยมีการเติบโตเป็นอย่างมาก โดยเฉพาะอย่างยิ่งในการค้นหาธุรกิจบุฟเฟ่ต์ในประเทศไทยเทียบตั้งแต่พ.ศ. 2547 จนถึง ปี 2566 ที่มากขึ้นถึง 1 แสนครั้ง ต่อ เดือน (สุรสิทธิ์ สัจจะเดร์, 2565) นอกจากนี้การสืบค้นหา Brand Keywords ประเภทบุฟเฟ่ต์ชาบูยังมาจากร้อยละ 15.4 ของการสืบค้นทั้งหมด การบริหารจัดการต้นทุนและผลประกอบการขายเพื่อการแข่งกับกับคู่แข่งจึงเป็นส่วนที่มีความสำคัญเป็นอย่างยิ่งสำหรับผู้ประกอบการ พบว่าทางร้านชาบู 3 ฟีน้องจัดการข้อมูลผลการดำเนินงานทุกเดือนจากการดึงข้อมูลระบบ POS บน loyverse และนำมาทำรายงานบน Microsoft Excel ทุกสิ้นเดือน เป็นผลให้ผู้ประกอบการร้านชาบู 3 ฟีน้องต้องเสียเวลาและทรัพยากรในการจัดการรูปดังกล่าวเป็นจำนวนมาก ร้านจึงมีความจำเป็นต้องจัดการรายงานสรุปผลการดำเนินงานให้สะดวกรวดเร็วมากขึ้นด้วยการจัดการระบบฐานข้อมูลจากการดำเนินงานจะพบว่าในเดือนล่าสุดธุรกิจมีการขาดทุนทำให้ผู้ประกอบการต้องการทราบข้อมูลการจัดการและบริหารต้นทุนของการดำเนินงานทั้งหมด 3 สาขา อีกทั้งการจัดการในระบบ POS บน loyverse สามารถติดตามจำนวนยอดขายได้ในช่วงสิ้นเดือน แต่หากทางร้านชาบู 3 ฟีน้องต้องการข้อมูลสรุปยอดขายเพื่อนำไปจัดทำ การส่งเสริมทางการตลาดจะทำให้ต้องรอข้อมูลจนถึงสิ้นเดือนส่งผลให้ธุรกิจเสียโอกาสที่จะสร้างยอดขาย ด้วยปัญหาดังกล่าว ทางคณะผู้จัดจึงได้ทำการสร้างระบบฐานข้อมูลและ Web Application เพื่อใช้ในการวิเคราะห์ข้อมูลยอดขายและต้นทุนจากการดำเนินงาน รวมถึงการทำรายงานผลประกอบการจากการวิเคราะห์ข้อมูลให้สะดวกและรวดเร็วมากขึ้น

ปัญหาที่ต้องการแก้ไข

1. ลดขั้นตอนในการวิเคราะห์สรุปผลการดำเนินงาน
2. ต้องการวิเคราะห์ข้อมูลต้นทุนจากการดำเนินงาน
3. ต้องการวิเคราะห์สินค้าขายดีในแต่ละเดือน
4. อัปเดตข้อมูลที่เกิดขึ้น เช่น ค่าใช้จ่ายรายวัน การเข้าทำงานของพนักงาน

5. ตรวจสอบความถูกต้องของฐานข้อมูล ว่ามีความสอดคล้องกันหรือไม่ เช่น ข้อมูลใบเสร็จกับจำนวนผลรวมของการขาย

แนวทางการแก้ไข

พัฒนา Web Application ที่เชื่อมต่อกับฐานข้อมูลเชิงความสัมพันธ์

ประโยชน์ที่คาดว่าจะได้รับ

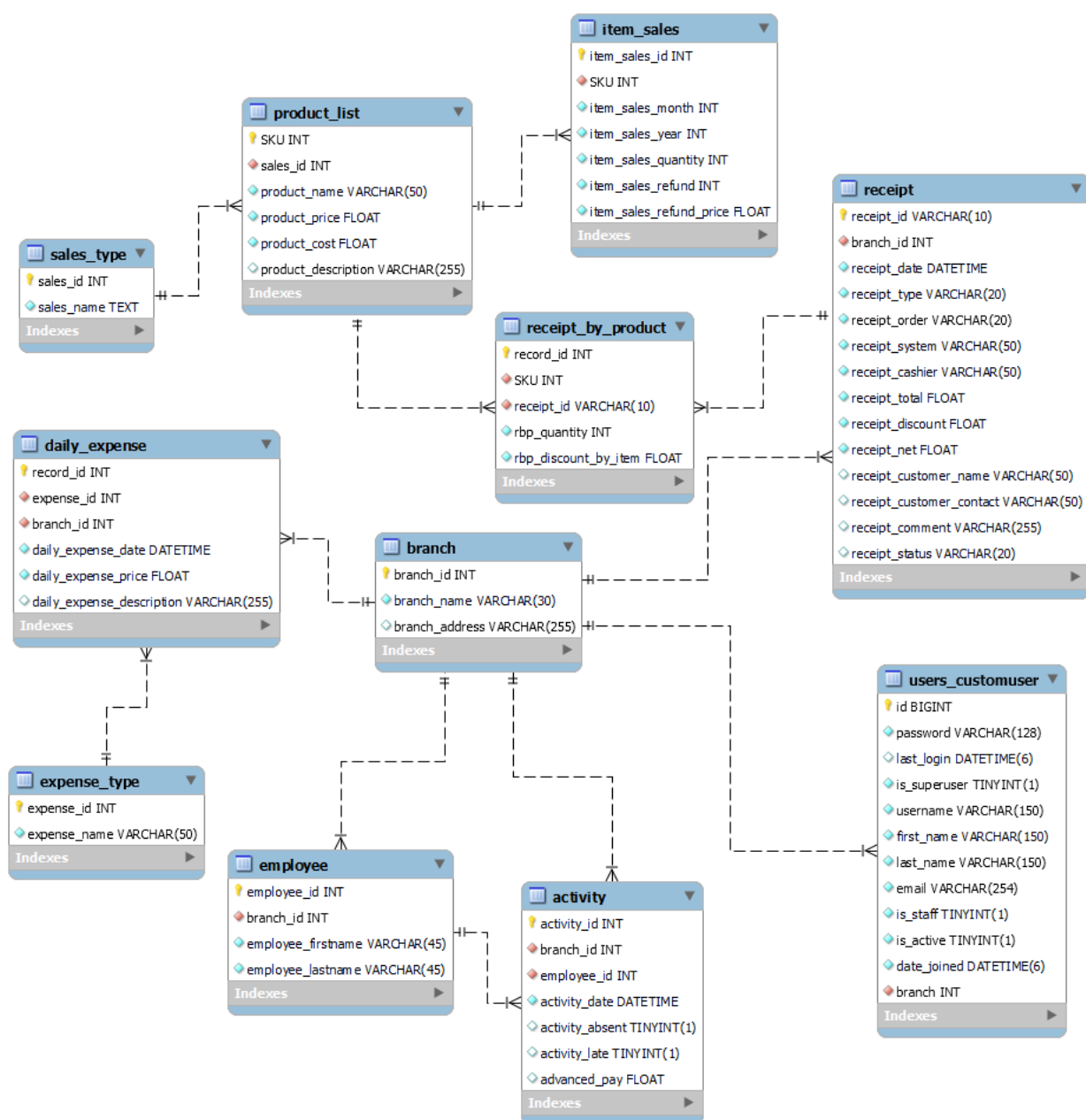
1. ช่วยสรุปผลการดำเนินงานให้สะดวกและรวดเร็วขึ้น
2. ช่วยจัดทำการวิเคราะห์ข้อมูลต้นทุนจากการดำเนินงาน
3. ช่วยจัดทำการวิเคราะห์สินค้าขายดีในแต่ละเดือนเพื่อนำไปต่อยอดในด้านการส่งเสริมการขายต่อไป
4. สามารถสร้างการสรุปและแสดงข้อมูลออกมาให้อยู่ในรูปแบบของแผนภาพ กราฟ
5. สามารถจัดการสิทธิในการเข้าถึงเพื่อดูและแก้ไขระบบฐานข้อมูลที่แสดงผลบนเว็บไซต์ให้การจัดเก็บข้อมูลมีความปลอดภัยต่อธุรกิจมากขึ้น

2. การออกแบบฐานข้อมูล

เครื่องมือที่ใช้

MySQL WorkBench

การออกแบบ Entity Relationship Diagram



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

1. **daily_expense**
เก็บค่าใช้จ่ายรายวันที่เกิดขึ้น
2. **expense_type**
เก็บข้อมูลประเภทค่าใช้จ่ายที่เกิดขึ้น
3. **employee**
เก็บข้อมูลพนักงานและสาขาที่ทำงาน
4. **branch**
เก็บข้อมูลชื่อสาขาและที่ตั้งสาขา
5. **activity**
เก็บข้อมูลการบันทึกการเข้าทำงานของพนักงานแยกตามสาขาและการเบิกจ่ายค่าแรงล่วงหน้า
6. **users_customer**
เก็บข้อมูลเกี่ยวกับผู้ใช้งานเว็บไซต์ของร้าน
7. **receipt**
เก็บข้อมูลลูกค้า รายการสินค้าที่สั่ง และยอดขาย
8. **receipt_by_product**
เก็บข้อมูลใบเสร็จและจำนวนสินค้าที่ขายได้ในใบเสร็จ
9. **item_sales**
เก็บรายการขายสินค้า
10. **product_list**
เก็บข้อมูลสินค้าที่ขายในร้าน
11. **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	advanced_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_description)

Output: -

Process: รับข้อมูล SKU, sales_id, sales, product_name, product_price, product_cost และ product_description จากนั้นเช็คค่า product_description มีค่าหรือไม่ หากไม่มีจะกำหนดเป็น null และเช็คค่า 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

2. 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 จากนั้นเช็คค่า daily_expense_description มีค่า

หรือไม่ หากไม่มีจะกำหนดเป็น null และนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง daily_expense

3. 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 จากนั้นเช็คว่าเป็น admin หรือไม่ หากใช่จะนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง activity หากไม่ใช่จะไม่สามารถนำข้อมูลนำเข้าเหล่านี้ insert เข้าตาราง activity ได้

4. 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

5. InsertReceipt

Description: ใช้ในการ insert ข้อมูลเข้าตาราง receipt

Input: รหัสใบเสร็จ (receipt_id)
 ชื่อสาขา (branch)
 วันที่ออกใบเสร็จ (receipt_date)
 ประเภทใบเสร็จ (receipt_type)
 รายการสินค้าที่ส่ง (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

6. 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

2. 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 เพื่อคำนวณหารายได้และค่าใช้จ่ายเฉลี่ยต่อ

เดือนแล้วตั้งชื่อคอลัมน์ใหม่ว่า avg_income และ avg_expense โดย GROUP BY month_year

3. 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

4. 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

5. 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

6. 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

7. 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

8. 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

9. 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 receipt_date ORDER BY receipt_date

10. 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

11. 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 เพื่อคำนวณค่าใช้จ่ายรวมแล้วตั้งชื่อคอลัมน์ใหม่ว่า total_expense โดย GROUP BY month_year และ daily_expense_date จากนั้น ORDER BY daily_expense_date

12. 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 เพื่อคำนวณหาจำนวนครั้งที่พนักงานมาสายและไม่มาทำงาน แล้วตั้งชื่อคอลัมน์ใหม่ว่า count_late และ count_absent โดย GROUP BY month_year

13. 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

14. 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 มาเช็คว่ามีรหัสสินค้าอยู่ในระบบหรือไม่ หากไม่มีรหัสสินค้าในระบบจะเช็คค่า 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

2. 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_expense หากน้อยกว่าหรือเท่ากับ 0 จะไม่นำข้อมูลค่าใช้จ่าย insert เข้าตาราง daily_expense และแจ้งเตือนค่าใช้จ่ายต้องมีค่ามากกว่า 0 กลับไป

3. การออกแบบ Web Application

ผู้ใช้งาน (Users)

1. ผู้ประกอบการ

นำเข้าข้อมูลจากระบบ POS บันทึกข้อมูล แก้ไขข้อมูล และวิเคราะห์ข้อมูลผลการดำเนินงานรายเดือน และดูต้นทุนและยอดขายจากการดำเนินงาน เพื่อเป็นประโยชน์ในการบริหารและจัดการธุรกิจ

2. ผู้จัดการสาขา หรือ ตัวแทนของแต่ละสาขา

บันทึกข้อมูลการจ่ายเงินของสาขา และบันทึกการเข้ามางานและมาสายของพนักงาน

ระบบและขอบเขตการทำงานของ Web Application

1. ระบบ Login-Logout

ให้ผู้จัดการสาขาและเจ้าของร้าน Login เข้ามาในเว็บไซต์เพื่อแก้ไขและบันทึกการทำงานต่างๆในร้านขาย 3 ฟังก์ชัน เมื่อใช้งานเสร็จเรียบร้อย สามารถกด Logout จากระบบ

2. การบันทึกรายการค่าใช้จ่ายในร้าน

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

3. การอัปโหลดข้อมูลจากระบบ Loyverse

เจ้าของร้านค่านำเข้าข้อมูลจาก Loyverse จากนั้นระบบจะทำการบันทึกไฟล์ csv จาก Loyverse เข้ามายังฐานข้อมูล โดยจะแสดงผลไปยังหน้าเว็บไซต์ว่านำเข้าข้อมูลอะไรบ้างเพื่อให้ผู้ใช้งานสามารถตรวจสอบความถูกต้องของข้อมูลได้

4. การบันทึกข้อมูลพนักงาน

ผู้จัดการสาขานำเข้าข้อมูลการเข้าออกการทำงานของพนักงาน เว็บไซต์จะแสดงหน้าจอให้เลือกสาขา ชื่อพนักงาน และ Checkbox ว่าขาดงาน มาสายหรือไม่ จากนั้นกดบันทึก ระบบจะเชื่อมต่อกับฐานข้อมูลและบันทึกรายการที่เกิดขึ้น

5. การสร้างสรุปกราฟผลการดำเนินงาน

เจ้าของสาขาเลือกเมนูสรุปผลการดำเนินงาน ระบบจะแสดงผล Dashboard แสดงข้อมูลที่เจ้าของสาขาเลือกจากทั้งหมด 4 รายการ ได้แก่

- ภาพรวม
- รายได้

- ค่าใช้จ่าย
- พนักงาน

จากนั้นแสดงผลการดำเนินงานแยกตามเดือน โดยจะประยอตรวม ผลการดำเนินงานแยกตามสาขา และ ผลการดำเนินงานแยกตามวัน

ขอบเขตที่นอกเหนือจากการดำเนินงาน

1. ความสวยงามของ Web Application
2. ความถูกต้องของข้อมูลและความสอดคล้องที่นำเข้าสู่ฐานข้อมูล
3. การเพิ่มความปลอดภัยของฐานข้อมูลที่เข้าถึงข้อมูลของร้านชาบู 3 ฟัน้อง เนื่องจากการใช้ username และ password เพื่อจำกัดความสามารถในการเข้าถึงของข้อมูล ป้องกันข้อมูลรั่วไหลและข้อมูลผิดพลาด

2.2 แสดงหน้าแรกของบัญชี admin



2.3 เข้าสู่ระบบด้วยผู้ใช้ manager ของแต่ละสาขา และใส่รหัสผ่าน

חנוך 3 פניון [หน้าแรก](#) [เข้าสู่ระบบ](#)

กรุณาเข้าสู่ระบบ

ชื่อผู้ใช้:

รหัสผ่าน:

[เข้าสู่ระบบ](#)

2.4 แสดงหน้าแรกของบัญชีของ Manager

חנוך 3 פניון [หน้าแรก](#) [ยินดีต้อนรับ คุณ manager01 \(สาขาพหลโยธิน 52\)](#) [ออกจากระบบ](#)

เมนู

- บันทึกผลประกอบการ
- คู่มือการใช้งาน

3. การเพิ่มผู้ใช้งานของแต่ละสาขา

3.1 คลิกไปที่เมนู “เพิ่มผู้ใช้งาน” จะสามารถเพิ่มผู้ใช้งานในแต่ละสาขาได้ ด้วยบัญชี admin

สาขา 3 พนักงาน หน้าแรก เพิ่มผู้ใช้งาน ยินดีต้อนรับ คุณ admin (ผู้จัดการ) ออกจากระบบ

เมนู

บันทึกข้อมูล

นำเข้าข้อมูล

แก้ไขข้อมูล

วิเคราะห์ข้อมูล

คู่มือการใช้งาน

ชื่อผู้ใช้:

Branch:

รหัสผ่าน:

ยืนยันรหัสผ่าน:

เพิ่มผู้ใช้งาน

3.2 การเพิ่มผู้ใช้งาน จำเป็นต้องกรอก ชื่อผู้ใช้งาน สาขา และยืนยันรหัสผ่านสำหรับผู้ใช้งานที่เพิ่มใหม่

สาขา 3 พนักงาน หน้าแรก เพิ่มผู้ใช้งาน ยินดีต้อนรับ คุณ admin (ผู้จัดการ) ออกจากระบบ

เมนู

บันทึกข้อมูล

นำเข้าข้อมูล

แก้ไขข้อมูล

วิเคราะห์ข้อมูล

คู่มือการใช้งาน

ชื่อผู้ใช้:

manager01

Branch:

สาขาพหลโยธิน 52

รหัสผ่าน:

.....

ยืนยันรหัสผ่าน:

.....

เพิ่มผู้ใช้งาน

3.3 หากรหัสผ่านที่ตั้งง่ายเกินไปและมีข้อความขึ้นเตือน

ชื่อผู้ใช้:

manager01

Branch:

สาขาพลโยธิน 52 ▾

รหัสผ่าน:

.....

ยืนยันรหัสผ่าน:

.....

This password is too common.
This password is entirely numeric.

เพิ่มผู้ใช้งาน

3.4 การบันทึกเสร็จเรียบร้อย

เมนู 3 ฟังก์ชัน หน้าแรก เพิ่มผู้ใช้งาน ยินดีต้อนรับ คุณ admin (ผู้จัดการ) ออกจากระบบ

เมนู

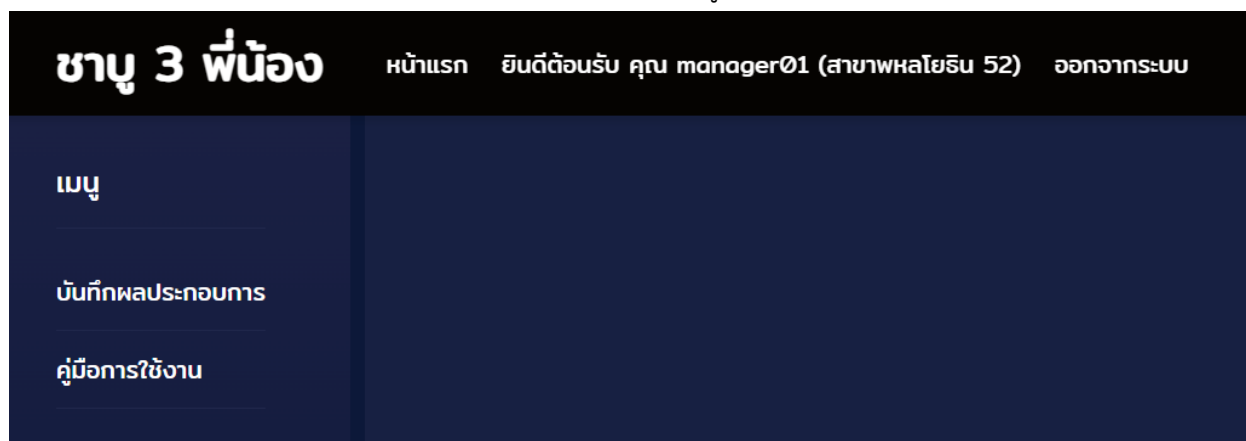
- บันทึกข้อมูล
- นำเข้าข้อมูล
- แก้ไขข้อมูล
- วิเคราะห์ข้อมูล
- คู่มือการใช้งาน

บันทึกสำเร็จ

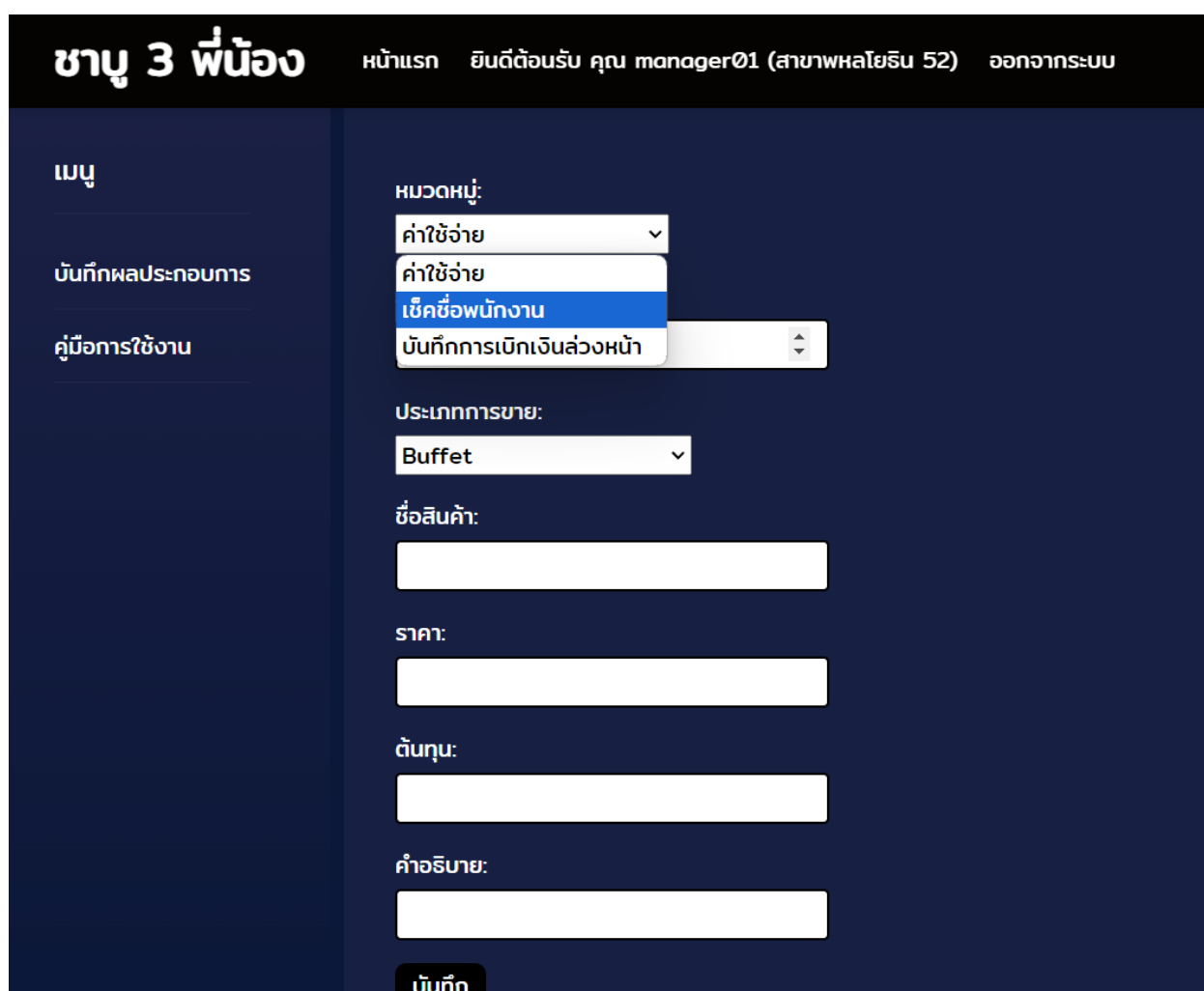
กลับสู่หน้าแรก บันทึกอีกครั้ง

4. ระบบการบันทึกข้อมูล

4.1 แสดงหน้าแรกที่บัญชี manager สามารถเข้าถึงข้อมูลได้



4.2 ระบบจะแสดงเฉพาะเมนูที่ Manager มีหน้าที่ ได้แก่บันทึกข้อมูล ค่าใช้จ่าย เช็คชื่อพนักงาน และการเบิกเงินล่วงหน้าของพนักงาน



4.3 ในขณะที่ Admin สามารถเข้าถึงข้อมูลการเพิ่มสินค้า ค่าใช้จ่าย เช็คชื่อพนักงาน และการเบิกเงินล่วงหน้าของพนักงาน อีกทั้งการแก้ไขและวิเคราะห์ข้อมูล

เมนู

- บันทึกข้อมูล
- นำเข้าข้อมูล
- แก้ไขข้อมูล
- วิเคราะห์ข้อมูล
- คู่มือการใช้งาน

เมนู 3 ฟังก์ชัน

[หน้าแรก](#)
[เพิ่มผู้ใช้งาน](#)
[ยืนยันต้อนรับ คุณ admin \(ผู้จัดการ\)](#)
[ออกจากระบบ](#)

หมวดหมู่:

เพิ่มสินค้า

เพิ่มสินค้า

ค่าใช้จ่าย

เช็คชื่อพนักงาน

บันทึกการเบิกเงินล่วงหน้า

ประเภทการขาย:

Buffet

ชื่อสินค้า:

ราคา:

ต้นทุน:

คำอธิบาย:

บันทึก

4.4 หากต้องการเพิ่มรายการสำหรับเมนูสินค้า ให้เลือกไปยังหมวดหมู่ “เพิ่มสินค้า” กรอกรหัส SKU ราคา ต้นทุน คำอธิบาย

เมนู 3 ฟังก์ชัน
[หน้าแรก](#)
[เพิ่มผู้ใช้งาน](#)
[ยินดีต้อนรับ คุณ admin \(ผู้จัดการ\)](#)
[ออกจากระบบ](#)

เมนู

บันทึกข้อมูล

นำเข้าข้อมูล

แก้ไขข้อมูล

วิเคราะห์ข้อมูล

คู่มือการใช้งาน

หมวดหมู่:

เพิ่มสินค้า

รหัส SKU:

777

ประเภทการขาย:

Buffet

ชื่อสินค้า:

Test

ราคา:

700

ต้นทุน:

300

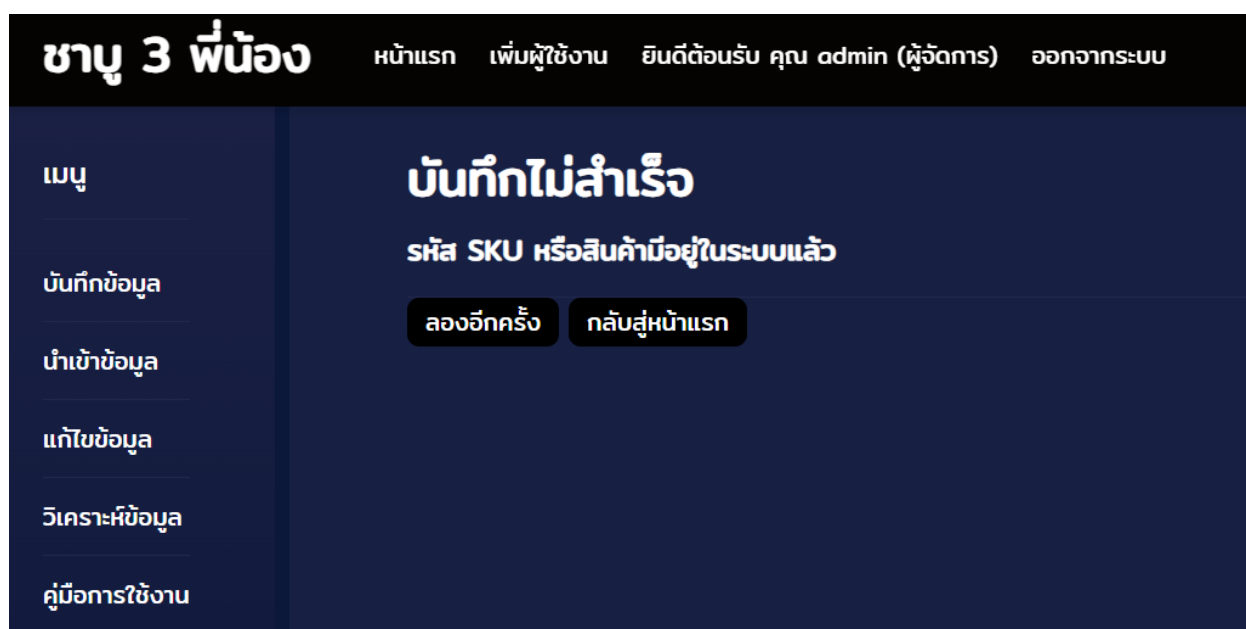
คำอธิบาย:

บันทึก

4.5 เมื่อใส่ข้อมูลเสร็จ ให้กดบันทึก หากรายการสินค้าดังกล่าวอยู่แล้วระบบจะทำการแจ้งเตือน ว่ามีสินค้าดังกล่าวอยู่แล้วและให้ผู้ใช้งานลองอีกครั้ง



4.6 หากรายการสินค้าดังกล่าวอยู่แล้วระบบจะทำการแจ้งเตือน ว่ามีสินค้าดังกล่าวอยู่แล้วและให้ผู้ใช้งานลองอีกครั้ง



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

เมนู
 บันทึกข้อมูล
 นำเข้าข้อมูล
 แก้ไขข้อมูล
 วิเคราะห์ข้อมูล
 คู่มือการใช้งาน

เมนู 3 ฟังก์ชัน
 หน้าแรก เพิ่มผู้ใช้งาน ยินดีต้อนรับ คุณ admin (ผู้จัดการ) ออกจากระบบ

หมวดหมู่:
 ค่าใช้จ่าย

ประเภทค่าใช้จ่าย:
 ค่าน้ำแข็ง/Pepsi/Ete

สาขา:
 สาขาพลโยธิน 52

วันที่-เวลา:
 12/19/2023 10:05 AM

จำนวน (บาท):
 100

คำอธิบาย:

บันทึก

4.8 การเช็คชื่อพนักงาน ให้เลือกหมวดหมู่ “เช็คชื่อพนักงาน” เลือกชื่อพนักงาน วันที่และเวลา ตีการขาดงานหรือมาสาย (หากมี) และกดบันทึก

เมนู 3 ฟังก์ชัน หน้าแรก เพิ่มผู้ใช้งาน ยินดีต้อนรับ คุณ admin (ผู้จัดการ) ออกจากระบบ

เมนู

- บันทึกข้อมูล
- นำเข้าข้อมูล
- แก้ไขข้อมูล
- วิเคราะห์ข้อมูล
- คู่มือการใช้งาน

หมวดหมู่:
เช็คชื่อพนักงาน ▾

พนักงาน:
A สาขาพลโยธิน 52 ▾

วันที่-เวลา:
12/19/2023 10:06 AM 📅

ขาดงาน:
☒

มาสาย (ถ้าขาดงานไม่ต้องใส่):
☐

บันทึก

4.9 บันทึกข้อมูลเบิกเงินล่วงหน้าของพนักงาน หากมีค่าเบิกล่วงหน้าติดลบระบบจะทำการแจ้งเตือน

เมนู 3 ฟังก์ชัน หน้าแรก เพิ่มผู้ใช้งาน ยินดีต้อนรับ คุณ admin (ผู้จัดการ) ออกจากระบบ

เมนู

- บันทึกข้อมูล
- นำเข้าข้อมูล
- แก้ไขข้อมูล
- วิเคราะห์ข้อมูล
- คู่มือการใช้งาน

หมวดหมู่:
บันทึกการเบิกเงินล่วงหน้า ▾

พนักงาน:
A สาขาพลโยธิน 52 ▾

วันที่-เวลา:
12/19/2023 10:07 AM 📅

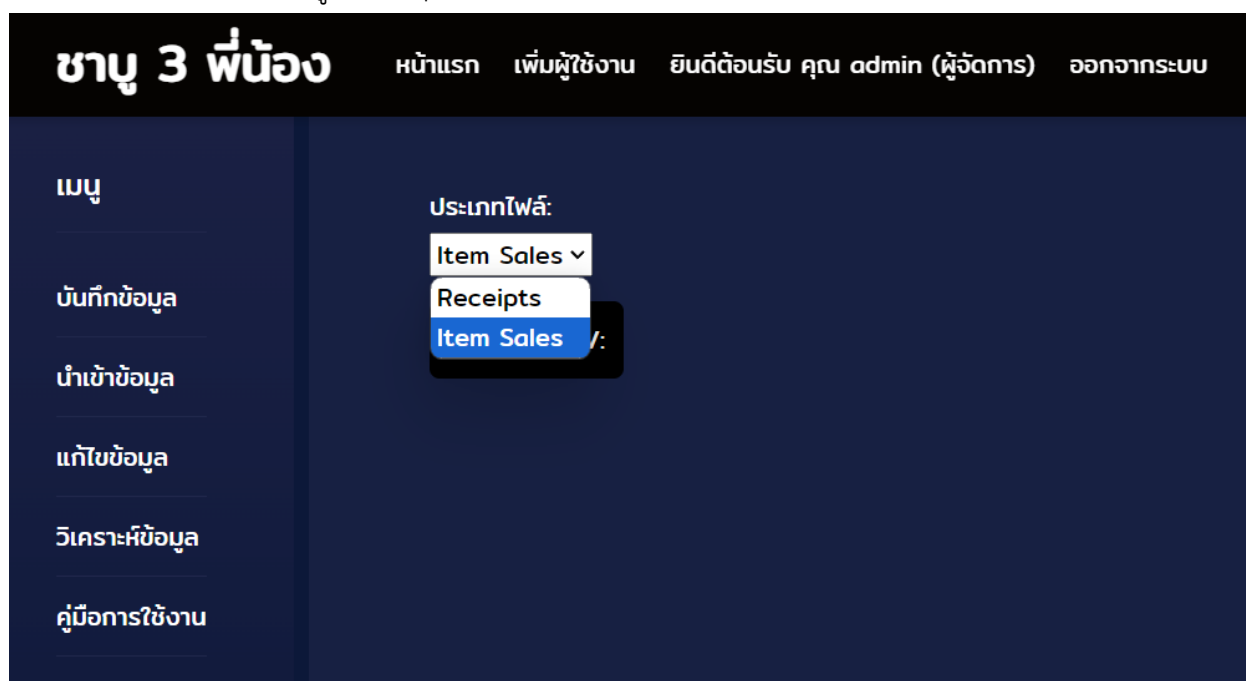
จำนวนเงิน:
100

บันทึก






5. การนำเข้าข้อมูล

5.1 เลื่อนำเข้าข้อมูล Receipt หรือ Item Sales



5.2 หากเลือก “Item Sales” ให้อัปโหลดไฟล์ item-sales

	item-sales-summary-2023-09-01-2023-0...	19/12/2023 10:08	Microsoft Excel Co...	22 KB
	receipts-2023-09-01-2023-09-01	19/12/2023 10:08	Microsoft Excel Co...	12 KB
	receipts-by-item-2023-09-01-2023-09-01	19/12/2023 10:07	Microsoft Excel Co...	68 KB

5.3 ตรวจสอบความถูกต้องก่อนบันทึก หากใน Item sales ระบบจะทำการสอบถามว่าต้องการบันทึก

เมนู
บันทึกข้อมูล
นำเข้าข้อมูล
แก้ไขข้อมูล
วิเคราะห์ข้อมูล
คู่มือการใช้งาน

เมนู 3 ฟังก์ชัน
หน้าแรก
เพิ่มผู้ใช้งาน
ยืนยันต้อนรับ คุณ admin (ผู้จัดการ)
ออกจากระบบ

บันทึกข้อมูล

รายการ	รหัสSKUสินค้า	ประเภท	สินค้าที่ขาย	ยอดขายรวม	สินค้าที่รับคืน	คืนเงิน	ส่วนลด	ยอดขายสุทธิ	ต้นทุนของสินค้า	กำไรรวม	กำไร	ภาษี
Buffet Deluxe	53	Buffet	470.0	126430.0	0.0	0.0	0.00	126430.00	126430.0	0.00	0.00%	0.0
Buffet Premium	322	Buffet	788.0	235612.0	0.0	0.0	0.00	235612.00	235612.0	0.00	0.00%	0.0
Buffet Staff Premium	323	Buffet	9.0	1350.0	0.0	0.0	0.00	1350.00	1350.0	0.00	0.00%	0.0
Buffet Starter	52	Buffet	347.0	65583.0	0.0	0.0	3.37	65579.63	65583.0	-3.37	-0.01%	0.0
Buffet เด็ก Deluxe	61	Buffet	10.0	1350.0	0.0	0.0	0.00	1350.00	1350.0	0.00	0.00%	0.0
Buffet เด็ก Premium	324	Buffet	7.0	1050.0	0.0	0.0	0.00	1050.00	1050.0	0.00	0.00%	0.0
Buffet เด็ก Starter	60	Buffet	2.0	180.0	0.0	0.0	0.00	180.00	180.0	0.00	0.00%	0.0
กระเทียม	175	Delivery	7.0	203.0	0.0	0.0	0.00	203.00	105.0	98.00	48.28%	0.0
กระเทียม	239	Take Home	2.0	30.0	0.0	0.0	0.00	30.00	30.0	0.00	0.00%	0.0
ควางดงใต้หัว	125	Delivery	1.0	29.0	0.0	0.0	0.89	28.11	15.0	13.11	46.64%	0.0
ทะเลสาบลิฟ	124	Delivery	4.0	116.0	0.0	0.0	0.00	116.00	60.0	56.00	48.28%	0.0

เมนู 3 ฟังก์ชัน
หน้าแรก
เพิ่มผู้ใช้งาน
ยืนยันต้อนรับ คุณ admin (ผู้จัดการ)
ออกจากระบบ




บันทึกข้อมูล
บันทึกไม่สำเร็จ มีสินค้าที่คุณยังไม่ได้เพิ่มเข้าระบบ
คุณต้องการเพิ่มสินค้าตามรายการด้านล่างและบันทึกอีกครั้งหรือไม่

บันทึก
ย้อนกลับ

รหัสSKUสินค้า	ประเภท	รายการ	ราคาต่อหน่วย	ต้นทุนต่อหน่วย
963258741.0	Buffet	Buffet Deluxe	269.0	269.0
NaN	None	None	NaN	NaN

(หมายเหตุ ในตัวอย่างของระบบการ Trigger เป็นไฟล์ข้อมูลใหม่ที่มีการสร้างขึ้นเพื่อยกตัวอย่างเท่านั้น)

5.4 หากเลือก “Receipt” ให้อัปโหลดไฟล์ receipt

	item-sales-summary-2023-09-01-2023-0...	19/12/2023 10:08	Microsoft Excel Co...	22 KB
	receipts-2023-09-01-2023-09-01	19/12/2023 10:08	Microsoft Excel Co...	12 KB
	receipts-by-item-2023-09-01-2023-09-01	19/12/2023 10:07	Microsoft Excel Co...	68 KB

5.5 ตรวจสอบความถูกต้องก่อนบันทึก

เมนู

บันทึกข้อมูล

นำเข้าข้อมูล

แก้ไขข้อมูล

วิเคราะห์ข้อมูล

คู่มือการใช้งาน

เลือกไฟล์ CSV:

ชื่อไฟล์ receipts-by-item-2023-09-01-2023-09-01.csv

บันทึก

ย้อนกลับ

กรุณาตรวจสอบข้อมูล

วันที่	เลขที่ใบเสร็จ	ประเภทใบเสร็จ	ประเภท	รหัสSKUสินค้า	รายการ	ตัวแปร	ปรับเงื่อนไขการใช้งาน	จำนวน	ยอดขายรวม	ส่วนลด	ยอดขายสุทธิ	ต้นทุนของสินค้า	กำไรรวม	ภาษี	ทางเลือกส่งของเดออร์	ระบบขายหน้าร้าน
1/9/23 21:58	4-16451	ลดราคา	Buffet	322	Buffet Premium	NaN	Shabu	2.0	598.0	0.0	598.0	598.0	0.0	0.0	ตามใบรับ	iPad Center Phaholyothin 52
1/9/23 21:58	4-16451	ลดราคา	น้ำชุป	45	ซูปญี่ปุ่นน้ำดำ	NaN	NaN	1.0	0.0	0.0	0.0	0.0	0.0	0.0	ตามใบรับ	iPad Center Phaholyothin 52
1/9/23 21:58	4-16451	ลดราคา	น้ำชุป	46	ซูปดิมขำน้ำใส	NaN	NaN	1.0	0.0	0.0	0.0	0.0	0.0	0.0	ตามใบรับ	iPad Center Phaholyothin 52

6. การวิเคราะห์ข้อมูล

6.1 เลือกหมวดหมู่ “สรุปภาพรวม” และเลือกเดือนที่ต้องการวิเคราะห์ข้อมูล หากเดือนที่เลือกยังไม่มีข้อมูลที่เกี่ยวข้อง ระบบจะแสดงว่า “ไม่พบข้อมูลในฐานข้อมูล”

เมนู

บ้านแรกเพิ่มผู้ใช้งานยืนยันตัวตนรับคุณสมบัติ admin (ผู้จัดการ) ออกจากระบบ

หมวดหมู่:
สรุปภาพรวม

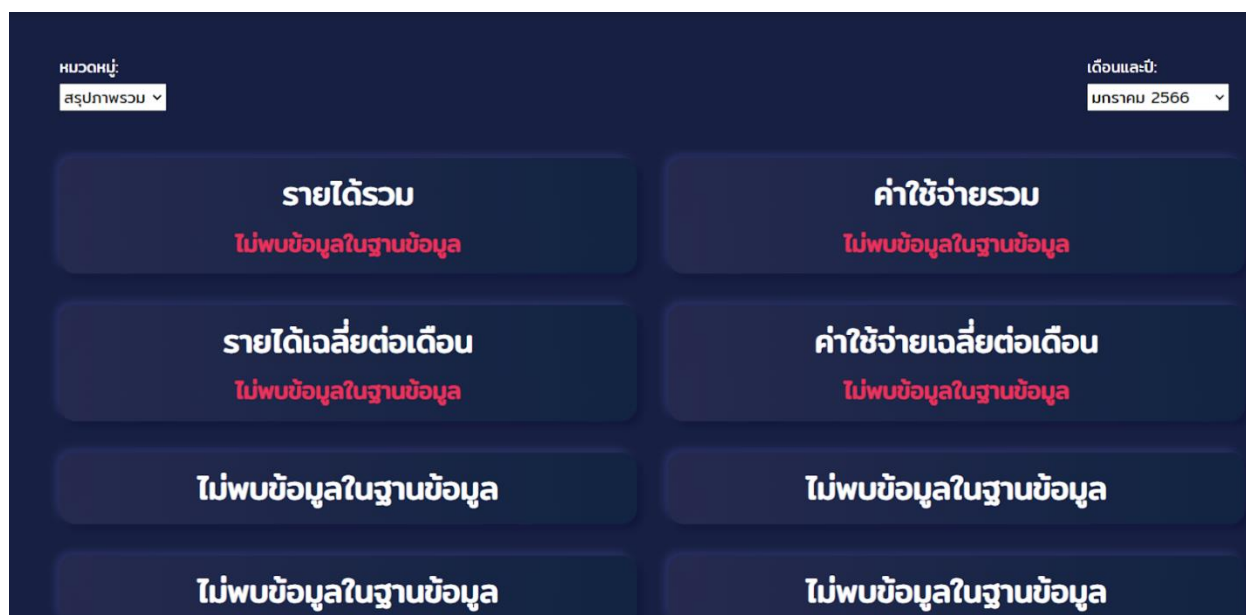
เดือนและปี:
กันยายน 2566

รายได้รวม
9256.0

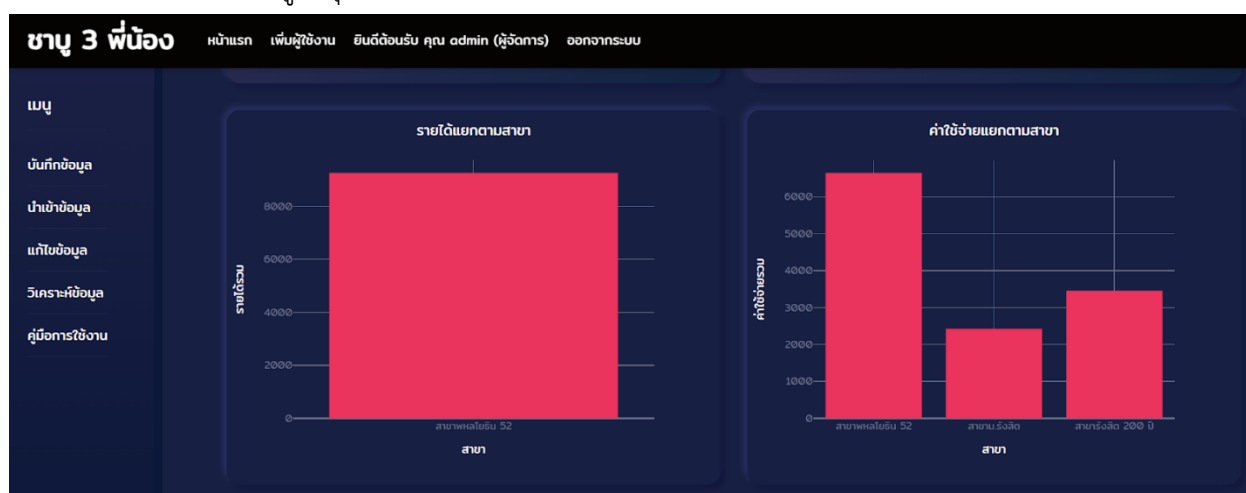
ค่าใช้จ่ายรวม
14126.0

รายได้เฉลี่ยต่อเดือน
578.5

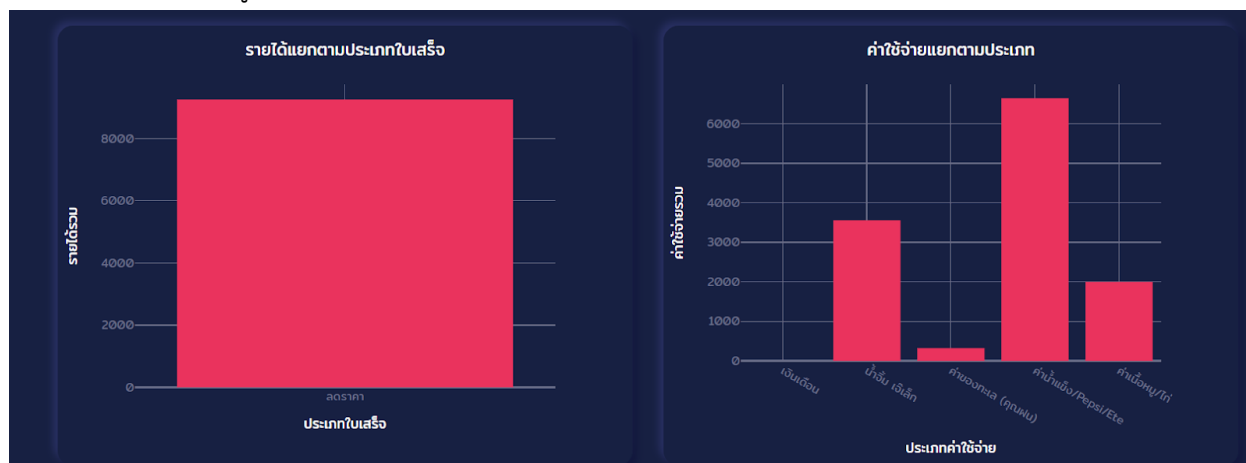
ค่าใช้จ่ายเฉลี่ยต่อเดือน
2287.6666666666665



6.2 แสดงผลข้อมูลสรุปรวม กราฟ ของรายได้และค่าใช้จ่าย



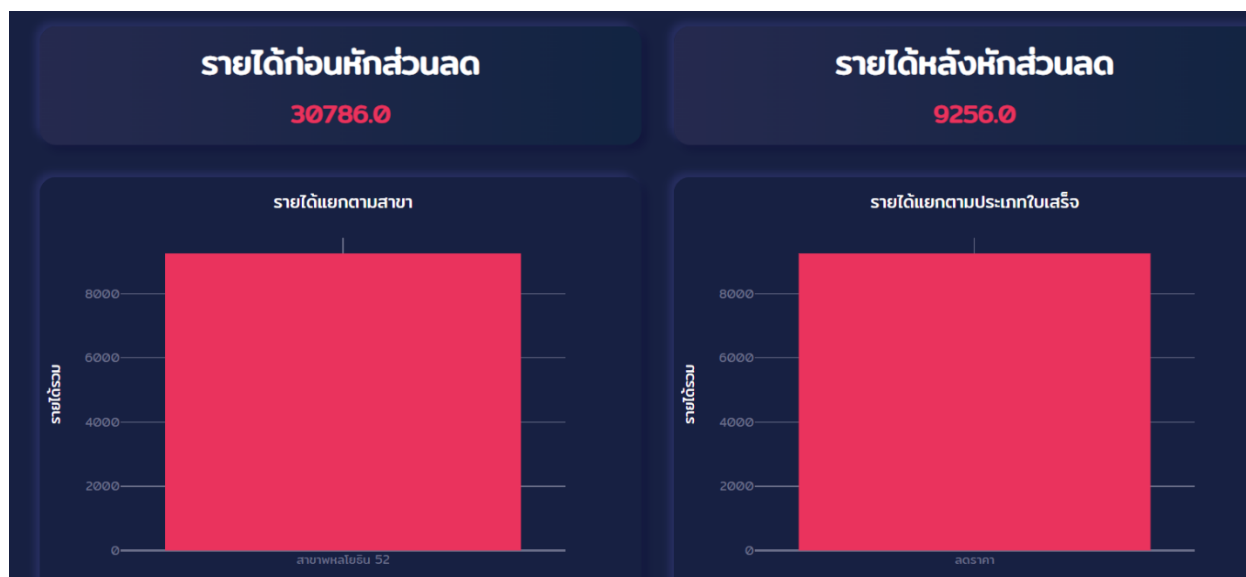
6.3 แสดงข้อมูลรายได้แยกตามประเภทใบเสร็จ (ใบเสร็จประเภท บุฟเฟต์, delivery) และค่าใช้จ่ายแยกประเภทตามหมวดหมู่



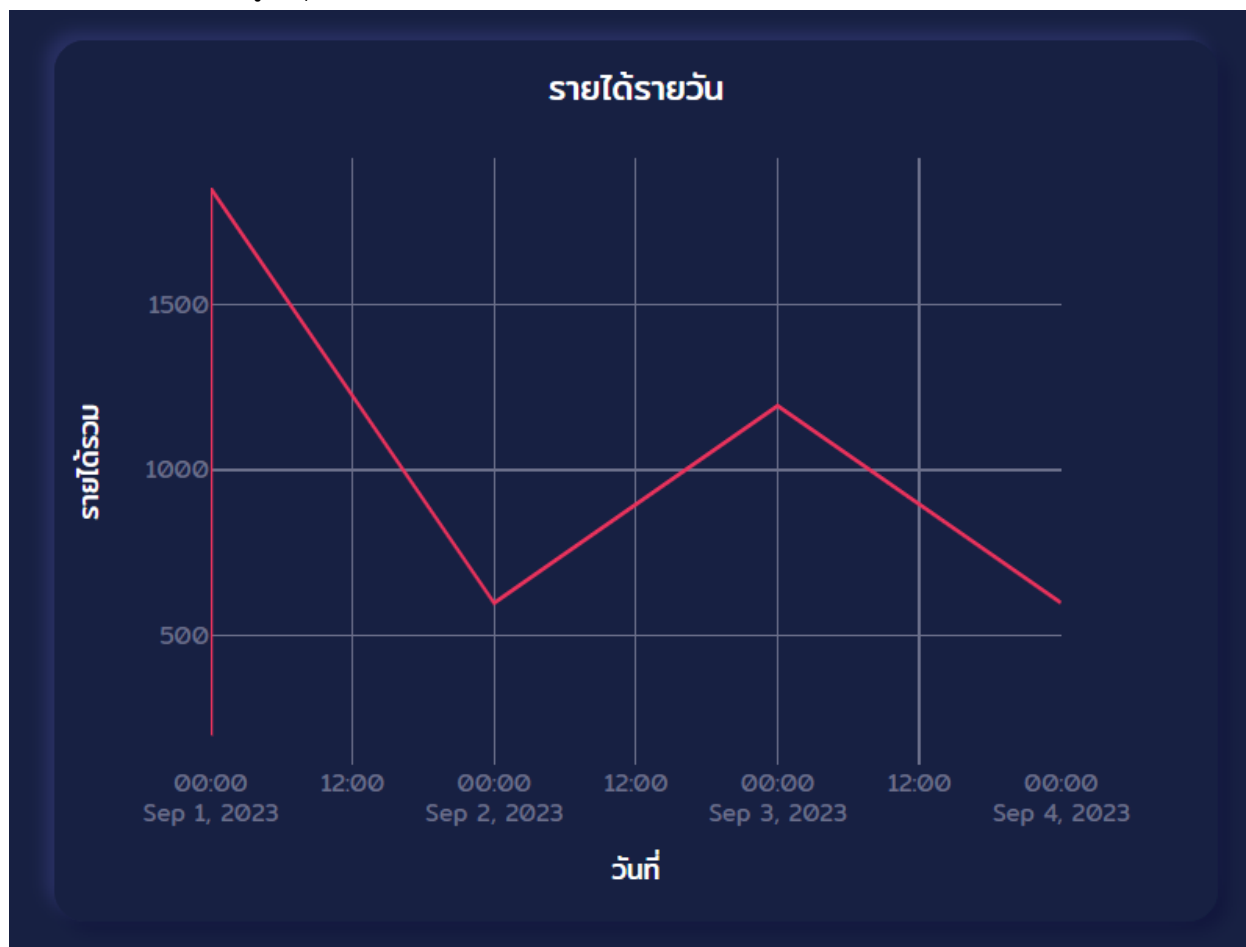
6.4 แสดง Top sales ยอดรวมทั้ง 3 สาขา



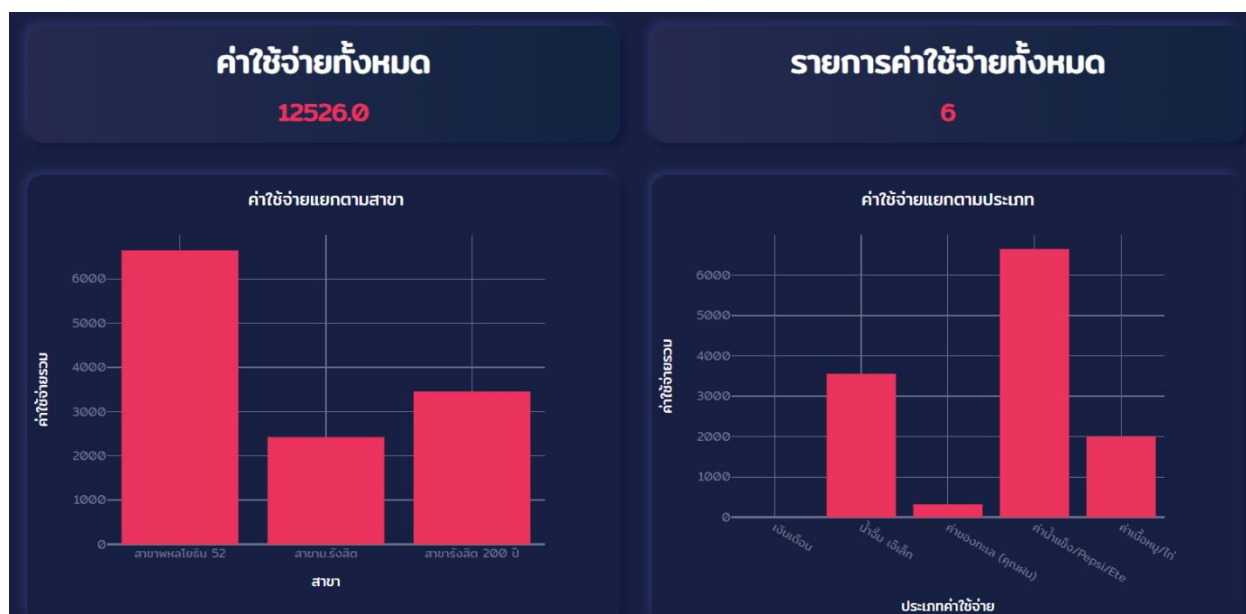
6.5 เลือกหมวดหมู่วิเคราะห์ และเลือกหมวดหมู่ “สรุปรายได้” แสดงข้อมูลยอดรวมรายได้ก่อนหักส่วนลด และหลังหักส่วนลดเป็นตัวเลข และแสดงข้อมูลรายได้แยกตามสาขา และรายได้แยกประเภทตามหมวดหมู่เป็นกราฟแท่ง



6.6 แสดงข้อมูลสรุปรวม กราฟของรายได้รายวัน

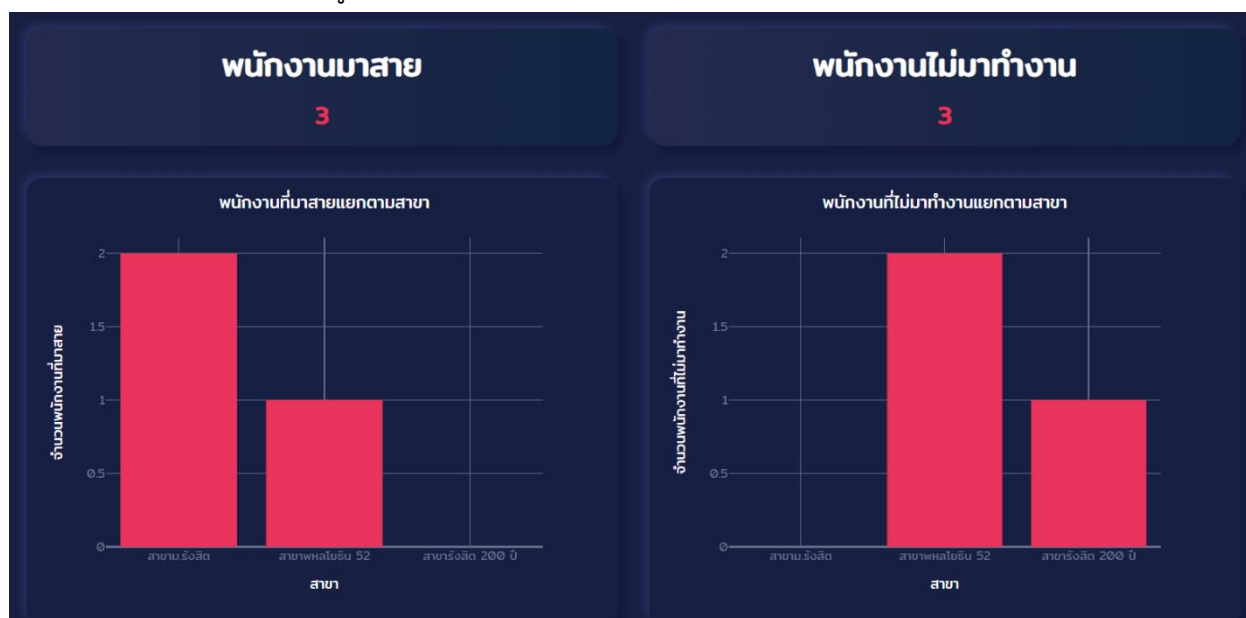


6.7 เลือกหมวดหมู่วิเคราะห์ และเลือกหมวดหมู่ “สรุปค่าใช้จ่าย” แสดงข้อมูลยอดรวมค่าใช้จ่ายทั้งหมด และรายการค่าใช้จ่ายทั้งหมด และแสดงข้อมูลค่าใช้จ่ายแยกตามสาขา และค่าใช้จ่ายแยกประเภทตามหมวดหมู่ เป็นกราฟแท่ง





6.9 เลือกหมวดหมู่วิเคราะห์ และเลือกหมวดหมู่ “สรุปพนักงาน” แสดงข้อมูลรวมพนักงานมาสาย รวมพนักงานไม่มาทำงาน และข้อมูลพนักงานแบ่งแยกตามสาขา



6.10 แสดงพนักงานมาสายแยกรายวันและพนักงานที่ไม่มาทำงานแยกรายวัน



5. การสรุปผลและข้อเสนอแนะ

การสรุปผล

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

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

ข้อเสนอแนะ

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

บรรณานุกรม

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

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;
```