

ระบบจัดการข้อมูลการเช่ารถ
Car Rental Management System

นางสวณันทิชา ไชยขวัญดี	รหัส6806022510548 SEC3
นางสาวสุกฤตา โรจนบัณฑิต	รหัส6806022511056 SEC3
นายณัฐภัทร แก้วเก่า	รหัส6806022511072 SEC3
นางสาวขวัญชนิษฐา จันทรพิงพลาย	รหัส6806022511137 SEC3

โครงการนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2568

ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

คำนำ

โครงการเรื่อง “ระบบจัดการข้อมูลการเช่ารถ” จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของรายวิชา Computer Programming หลักสูตรวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ โดยมีวัตถุประสงค์เพื่อให้นักศึกษาได้นำความรู้ที่ได้เรียนมาประยุกต์ใช้ในการพัฒนาโปรแกรมขนาดเล็กที่สามารถทำงานได้จริง โปรแกรมดังกล่าวถูกพัฒนาด้วยภาษา Python โดยมุ่งเน้นการช่วยบริหารจัดการข้อมูลการเช่ารถให้สะดวกและถูกต้องมากขึ้น เช่น การบันทึกและค้นหาข้อมูลลูกค้า ข้อมูลรถ รวมถึงการจัดทำรายงานการเช่า ทั้งนี้เพื่อเป็นการฝึกทักษะการวิเคราะห์ ออกแบบ และเขียนโปรแกรมของผู้จัดทำ ไม่ได้มีขอบเขตกว้างขวางเหมือนระบบจริง แต่หวังว่าจะเป็นประโยชน์สำหรับผู้สนใจการพัฒนาระบบงานในลักษณะนี้ต่อไป

ผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษา ที่ให้คำแนะนำและข้อเสนอแนะตลอดการทำโครงการ ผู้จัดทำหวังว่าโครงการฉบับนี้จะเป็นประโยชน์แก่ผู้สนใจ และสามารถนำไปประยุกต์ใช้ในการพัฒนาระบบงานในด้านอื่น ๆ ต่อไปได้

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	ข
สารบัญภาพ	ง
สารบัญภาพ(ต่อ)	จ
สารบัญภาพ(ต่อ)	ฉ
สารบัญตาราง	ช
บทที่ 1 บทนำ	1
1.1 วัตถุประสงค์ของโครงการ	1
1.2 ขอบเขตของโครงการ	1
1.3 ประโยชน์ที่คาดว่าจะได้รับ	2
1.4 เครื่องมือที่คาดว่าจะต้องใช้	2
บทที่ 2 เอกสารที่เกี่ยวข้อง	3
2.1 งานวิจัยที่เกี่ยวข้อง	3
2.2 โดเมนของระบบ	3
2.3 ผู้ใช้เป้าหมาย	3
2.4 ข้อกำหนดเชิงหน้าที่ของระบบ	4
2.5 โครงสร้างข้อมูล Cars	5
2.6 โครงสร้างข้อมูล Customers	6
2.7 โครงสร้างข้อมูล Contracts	8
2.8 ไฟล์ report.txt	10

สารบัญ(ต่อ)

	หน้า
บทที่ 3 การใช้งานจัดการข้อมูลการเช่ารถ	13
3.1 การใช้งานโปรแกรมเช่ารถ	13
3.2 การใช้งานโปรแกรมเพิ่มข้อมูล	17
3.3 การงานโปรแกรมการแก้ไขข้อมูล	19
3.4 การใช้งานโปรแกรมลบข้อมูล	21
3.5 การใช้งานโปรแกรมแสดงข้อมูล	23
3.6 เรียกดูสัญญาเช่าแบบกรอง	25
บทที่ 4 อธิบายการทำงานของโค้ด	27
4.1 ฟังก์ชันไบนารีพื้นฐานโปรแกรมระบบเช่ารถ(Car Rental System)	27
4.2 ฟังก์ชันจัดการระบบเช่ารถ	30
4.3 ฟังก์ชันเมนูจัดการข้อมูลระบบเช่ารถ	41
4.4 ฟังก์ชันสร้างรายงาน	44
4.5 ฟังก์ชันเมนูหลักของระบบ (Main Program)	51
บทที่ 5 สรุปผลการดำเนินงานและข้อเสนอแนะ	53
5.1 สรุปผลงาน	53
5.2 อภิปรายผล	53
5.3 ข้อเสนอแนะ	53
5.4 สิ่งที่ได้จัดทำได้รับในการพัฒนาโครงการ	54

สารบัญภาพ

	หน้า
ภาพที่ 2-1 ไฟล์ report.txt	9
ภาพที่ 3-1 การเลือกใช้งานฟังก์ชัน Manage Car	12
ภาพที่ 3-2 เมนูของ Manage Car	13
ภาพที่ 3-3 การเลือกใช้งานฟังก์ชัน Manage Customers	14
ภาพที่ 3-4 เมนูของ Manage Customers	15
ภาพที่ 3-5 การเลือกใช้งานฟังก์ชัน Manage Contracts	15
ภาพที่ 3-6 เมนูของ Manage Contracts	15
ภาพที่ 3-7 การเลือกใช้งานฟังก์ชัน Generate report	16
ภาพที่ 3-8 การเลือกใช้งานฟังก์ชัน Exit	16
ภาพที่ 3-9 การเลือกใช้งานฟังก์ชัน Add (เพิ่ม)รถ	17
ภาพที่ 3-10 การเพิ่มรถ	17
ภาพที่ 3-11 การเลือกใช้งานฟังก์ชัน Add (เพิ่ม)ข้อมูลลูกค้า	17
ภาพที่ 3-12 การเพิ่มข้อมูลลูกค้า	18
ภาพที่ 3-13 การเลือกใช้งานฟังก์ชัน Add (เพิ่ม)ข้อมูลลูกค้า	18
ภาพที่ 3-14 การเพิ่มข้อมูลสัญญาเช่า	18
ภาพที่ 3-15 การเลือกใช้งานฟังก์ชัน การแก้ไขข้อมูลรถ	19
ภาพที่ 3-16 การแก้ไขข้อมูลของรถ	19
ภาพที่ 3-17 การเลือกใช้งานฟังก์ชันการแก้ไขข้อมูลลูกค้า	19
ภาพที่ 3-18 การแก้ไขข้อมูลลูกค้า	20
ภาพที่ 3-19 การเลือกใช้งานฟังก์ชันการแก้ไขข้อมูลสัญญาเช่า	20
ภาพที่ 3-20 การแก้ไขข้อมูลลูกค้า	20
ภาพที่ 3-21 การเลือกใช้งานฟังก์ชันการลบข้อมูลรถ	21

สารบัญภาพ(ต่อ)

ภาพที่ 3-21 การเลือกใช้งานฟังก์ชันการลบข้อมูลรถ	21
ภาพที่ 3-22 การลบข้อมูลรถ	21
ภาพที่ 3-23 การเลือกใช้งานฟังก์ชันการลบข้อมูลลูกค้า	21
ภาพที่ 3-24 การลบข้อมูลลูกค้า	22
ภาพที่ 3-25 การเลือกใช้งานฟังก์ชันการลบข้อมูลสัญญาเช่า	22
ภาพที่ 3-26 การลบข้อมูลสัญญาเช่า	22
ภาพที่ 3-27 การเลือกใช้งานฟังก์ชันการแสดงผลข้อมูลรถทั้งหมด	23
ภาพที่ 3-28 การแสดงผลข้อมูลรถทั้งหมด	23
ภาพที่ 3-29 การเลือกใช้งานฟังก์ชันการแสดงผลข้อมูลลูกค้าทั้งหมด	23
ภาพที่ 3-30 การแสดงผลข้อมูลลูกค้าทั้งหมด	24
ภาพที่ 3-31 การเลือกใช้งานฟังก์ชันการแสดงผลข้อมูลสัญญาเช่าทั้งหมด	24
ภาพที่ 3-32 การแสดงผลข้อมูลสัญญาเช่าทั้งหมด	24
ภาพที่ 3-33 การเลือกใช้งานฟังก์ชันการแสดงผลข้อมูลสัญญาเช่าแบบกรอง	25
ภาพที่ 3-34 ปีและเดือน	25
ภาพที่ 3-35 สถานะ	25
ภาพที่ 3-36 รหัสลูกค้าที่เช่ารถ	26
ภาพที่ 3-37 รหัสรถที่ถูกเช่า	26
ภาพที่ 4-1 _fix_bytes	27
ภาพที่ 4-2 _unfix_bytes	27
ภาพที่ 4-3 _ensure_files()	28
ภาพที่ 4-4 _ensure_files()	28
ภาพที่ 4-5 _write_record()	28
ภาพที่ 4-6 def _detect_contract_rec_size()	29

สารบัญภาพ(ต่อ)

	หน้า
ภาพที่ 4-7 def _detect_contract_rec_size()	29
ภาพที่ 4-8 ฟังก์ชัน add_car()	30
ภาพที่ 4-9 ฟังก์ชัน update_car()	31
ภาพที่ 4-10 ฟังก์ชัน delete_car()	33
ภาพที่ 4-11 ฟังก์ชัน add_customer()	33
ภาพที่ 4-12 ฟังก์ชัน update_customer()	34
ภาพที่ 4-13 ฟังก์ชัน delete_customer()	35
ภาพที่ 4-14 ฟังก์ชัน add_contract()	37
ภาพที่ 4-15 ฟังก์ชัน update_contract()	38
ภาพที่ 4-16 ฟังก์ชัน delete_contract()	39
ภาพที่ 4-17 ฟังก์ชัน view_filter()	40
ภาพที่ 4-18 ฟังก์ชัน manage_cars_menu()	41
ภาพที่ 4-19 ฟังก์ชัน manage_customers_menu()	42
ภาพที่ 4-20 ฟังก์ชัน manage_contracts_menu()	43
ภาพที่ 4-21 ฟังก์ชัน generate_report() ใช้รวบรวมข้อมูล	44
ภาพที่ 4-22 ฟังก์ชัน generate_report() ใช้อ่านและโหลดข้อมูลสัญญาเช่ารถ	45
ภาพที่ 4-23 ฟังก์ชัน generate_report() กำหนดรูปแบบและโครงสร้างของรายงานสรุป	47
ภาพที่ 4-24 ฟังก์ชัน generate_report() ใช้กำหนดรูปแบบและโครงสร้างของรายงานสรุป	47
ภาพที่ 4-25 ฟังก์ชัน generate_report() ใช้จัดรูปแบบการแสดงผล	48
ภาพที่ 4-26 ฟังก์ชัน generate_report() ใช้สร้างรายงานสรุปผลการดำเนินงานในรูปแบบต่างๆ	49
ภาพที่ 4-27 ฟังก์ชัน generate_report() ใช้ทำหน้าที่สรุปข้อมูลระบบและบันทึกผลออก	50
ภาพที่ 4-28 ฟังก์ชัน main()	52

สารบัญตาราง

	หน้า
ตารางที่ 2.1 ข้อกำหนดเชิงหน้าที่ของระบบ	4
ตารางที่ 2.2 โครงสร้างข้อมูล Cars	5
ตารางที่ 2.3 โครงสร้างข้อมูล Customers	6
ตารางที่ 2.4 โครงสร้างข้อมูล Contracts	8

บทที่ 1

บทนำ

1.1 วัตถุประสงค์ของโครงการ

1.1.1 เพื่อพัฒนาระบบจัดการข้อมูลการเช่ารถที่สามารถบันทึกและจัดเก็บข้อมูลลูกค้า ข้อมูลรถ และการเช่าได้อย่างเป็นระบบ

1.1.2 เพื่อเพิ่มความสะดวกรวดเร็ว และความถูกต้องในการสืบค้นและจัดทำรายงาน

1.1.3 เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วย Python

1.1.4 เพื่อเรียนรู้วิธีการจัดการข้อมูลและไฟล์

1.2 ขอบเขตของโครงการ

1.2.1 ระบบนี้เป็นระบบจัดเก็บและจัดการข้อมูลการเช่ารถสำหรับบริษัทให้เช่ารถทุกขนาด รองรับการจัดเก็บข้อมูลรถ ลูกค้า และสัญญาเช่า เพื่ออำนวยความสะดวกในการเช่า การคืนรถ และการติดตามประวัติการเช่า

1.2.2 ผู้ใช้เป้าหมาย

1. พนักงานรับจอง/เช่า (staff) — เพิ่ม/แก้ไข/ลบ/ค้นหาข้อมูลรถ ลูกค้า และสัญญาเช่า
2. ผู้จัดการ (manager) — ตรวจสอบรายงานและสถิติการเช่า
3. เจ้าหน้าที่บัญชี — ดึงข้อมูลสัญญาและยอดการเช่า

1.2.3 งานประจำ

1. รับจอง/ทำสัญญาเช่ารถใหม่
2. บันทึกการคืนรถและคำนวณค่าใช้จ่าย
3. จัดการข้อมูลรถและลูกค้า
4. สร้างรายงานสรุปยอดเช่าและสถานะรถ

1.2.4 ปริมาณข้อมูลที่คาด

1. รถ 50–500 คัน
2. ลูกค้า 500–2000 คน
3. สัญญาเช่า หลักพันรายการต่อปี

1.3 ประโยชน์ที่คาดว่าจะได้รับ

- 1.3.1 ได้โปรแกรมต้นแบบสำหรับจัดการข้อมูลการเช่ารถที่ใช้งานได้จริงในระดับเบื้องต้น
- 1.3.2 เรียนรู้การจัดการข้อมูลและไฟล์
- 1.3.3 นักศึกษาได้ฝึกทักษะการวิเคราะห์ออกแบบ
- 1.3.4 นักศึกษาได้เรียนรู้การพัฒนาโปรแกรมด้วยภาษา Python

1.4 เครื่องมือที่คาดว่าจะต้องใช้

- 1.4.1 โปรแกรม Visual Studio Code
- 1.4.2 Microsoft Office

บทที่ 2

เอกสารที่เกี่ยวข้อง

2.1 งานวิจัยและโครงการที่เกี่ยวข้อง

จากการศึกษาข้อมูลที่เกี่ยวข้อง พบว่ามีการจัดการการเช่ารถในหลายรูปแบบโดยระบบทั่วไป จะเน้นการเก็บข้อมูลรถ ข้อมูลลูกค้า และข้อมูลสัญญาเช่า เพื่อความสะดวกในการให้บริการ เช่น การจอง การคืนรถ และการติดตามประวัติการเช่า ระบบที่ดีควรช่วยให้ผู้ใช้สามารถค้นหาข้อมูลได้รวดเร็ว มีรายงานสรุป และลดความผิดพลาดจากการจัดการด้วยเอกสารแบบเดิม

2.2 โดเมนของระบบ

ระบบจัดการข้อมูลการเช่ารถ (Car Rental Management System) มีหน้าที่ในการจัดเก็บ และบริหารข้อมูลที่เกี่ยวข้องกับรถยนต์ ลูกค้า และสัญญาเช่า โดยสามารถรองรับการใช้งานได้ทั้ง บริษัทขนาดเล็กและขนาดใหญ่ ระบบนี้ช่วยให้พนักงานสามารถทำงานได้สะดวกมากขึ้น ตั้งแต่การเพิ่ม/แก้ไข/ลบข้อมูล ไปจนถึงการค้นหาและการออกสัญญาเช่า

2.3 ผู้ใช้เป้าหมาย

2.3.1 พนักงานรับจอง/เช่า (Staff): ทำหน้าที่เพิ่ม แก้ไข ลบ และค้นหาข้อมูลรถ ลูกค้า และสัญญาเช่า

2.3.2 ผู้จัดการ (Manager): ตรวจสอบรายงานและสถิติการเช่า

2.3.3 เจ้าหน้าที่บัญชี (Accountant): ดึงข้อมูลการเช่าเพื่อใช้ในการคำนวณยอดและจัดการทางการเงิน

2.4 ข้อกำหนดเชิงหน้าที่ของระบบ

เพื่อให้ระบบสามารถทำงานได้ตามวัตถุประสงค์ ได้กำหนดฟังก์ชันหลัก ดังนี้

รหัสฟังก์ชัน	รายละเอียด
FR1	เพิ่ม/แก้ไข/ลบข้อมูลรถ เช่น ทะเบียน ประเภท รถ ราคาเช่า สถานะ (available/rented)
FR2	เพิ่ม/แก้ไข/ลบข้อมูลลูกค้า เช่น ชื่อ นามสกุล เลขบัตรประชาชน เบอร์ติดต่อ
FR3	สร้างสัญญาเช่า โดยเชื่อมโยงลูกค้าและรถ กำหนดวันเริ่มและวันสิ้นสุด
FR4	ปิดสัญญาเช่า คำนวณค่าใช้จ่ายจริง และอัปเดต สถานะรถ
FR5	ค้นหารถ ลูกค้า หรือสัญญาตามคีย์หรือเงื่อนไข
FR6	สร้างรายงานการเช่ารถ (report.txt) เช่น จำนวนสัญญาในช่วงเวลา ยอดรายได้ต่อเดือน

ตารางที่ 2.1 ข้อกำหนดเชิงหน้าที่ของระบบ

2.5 โครงสร้างข้อมูล Cars

ระบบนี้ออกแบบให้มีการเก็บข้อมูลหลัก 3 ส่วน ได้แก่ ข้อมูลรถ (Cars) ข้อมูลลูกค้า (Customers) และข้อมูลสัญญาเช่า (Contracts) โครงสร้างข้อมูลรถประกอบด้วยรหัสรถ ทะเบียน ยี่ห้อ รุ่น และราคาเช่า โดยมีฟิลด์ status และ rented สำหรับบอกสถานะว่ารถพร้อมให้เช่าหรือไม่

Field	Type	Length	Description
car_id	CHAR(10)	10	รหัสรถ (unique)
plate	CHAR(10)	10	ทะเบียนรถ
brand	CHAR(30)	30	ยี่ห้อรถ
model	CHAR(20)	20	รุ่นรถ
rate	FLOAT	10	ราคาเช่า

ตารางที่ 2.2 โครงสร้างข้อมูล Cars

2.5.1 car_id

เป็นรหัสประจำรถ ใช้ในการระบุรถแต่ละคันไม่ให้ซ้ำกัน ถือเป็น “Primary Key” ของตาราง เพื่อใช้ในการเชื่อมโยงกับข้อมูลสัญญาเช่าหรือข้อมูลอื่น ๆ ภายในระบบ

2.5.2 plate

คือหมายเลขทะเบียนรถ ซึ่งเป็นข้อมูลสำคัญที่ใช้แยกรถแต่ละคันในชีวิตจริง และช่วยให้พนักงานสามารถตรวจสอบหรือค้นหาได้ง่าย

2.5.3 brand

หมายถึงยี่ห้อของรถ เช่น Toyota, Honda, Nissan เป็นต้น ใช้สำหรับจำแนกประเภทของรถที่บริษัทให้บริการ

2.5.4 model

คือรุ่นของรถ เช่น Civic, Fortuner, Almera เป็นต้น เพื่อให้ทราบรายละเอียดของรถแต่ละยี่ห้อได้ชัดเจนมากขึ้น

2.5.5 rate

คือราคาเช่าของรถ โดยปกติจะคิดเป็น “ราคาต่อวัน” ข้อมูลนี้จะถูกนำไปใช้คำนวณค่าใช้จ่ายรวมในสัญญาเช่า

2.6 โครงสร้างข้อมูล Customers

โครงสร้างข้อมูล Customers ใช้เก็บข้อมูลลูกค้าเพื่อระบุตัวตนและสามารถติดต่อได้ โดยมีรหัสลูกค้าเป็นprimary key

Field	Type	Length	Description
cust_id	CHAR(10)	10	รหัสลูกค้า
name	CHAR(50)	50	ชื่อลูกค้า
id_card	CHAR(13)	13	เลขบัตรประชาชน
phone	CHAR(15)	15	เบอร์โทร
email	CHAR(50)	50	อีเมล

ตารางที่ 2.3 โครงสร้างข้อมูล Customers

2.6.1 cust_id

ใช้เป็นรหัสประจำตัวลูกค้า (Customer ID) ซึ่งไม่ซ้ำกันในระบบ มีหน้าที่ระบุเอกลักษณ์ของลูกค้าแต่ละคน ใช้เป็นคีย์หลัก (Primary Key) สำหรับเชื่อมโยงข้อมูลกับตารางอื่น เช่น สัญญาเช่า (Contracts)

2.6.2 name

เก็บชื่อ-นามสกุลของลูกค้า เพื่อใช้ระบุตัวบุคคลที่เข้ามาใช้บริการเช่ารถ เป็นข้อมูลสำคัญที่แสดงในสัญญาและรายงานการเช่า

2.6.3 id_card

หมายเลขบัตรประชาชนของลูกค้า ใช้เพื่อยืนยันตัวตนของผู้เช่ารถและป้องกันการปลอมแปลงข้อมูล เป็นข้อมูลที่ใช้ในการตรวจสอบประวัติของลูกค้าได้

2.6.4 phone

เบอร์โทรศัพท์ติดต่อของลูกค้า ใช้สำหรับติดต่อสอบถาม แจ้งข้อมูลการเช่ารถ หรือติดต่อในกรณีเกิดปัญหาระหว่างการเช่ารถ

2.6.5 email

ที่อยู่อีเมลของลูกค้า ใช้สำหรับการติดต่อในรูปแบบออนไลน์ เช่น การส่งเอกสารยืนยันการ

เช่า หรือใบแจ้งเดือนการคืนรถ

2.7 โครงสร้างข้อมูล Contracts

โครงสร้างข้อมูล Contracts ใช้สำหรับเก็บรายละเอียดการเช่ารถแต่ละครั้ง โดยเชื่อมโยงลูกค้ากับรถที่ถูกเช่า รวมถึงระบุวันเริ่ม วันสิ้นสุด และค่าใช้จ่ายรวม

Field	Type	Length	Description
contract_id	CHAR(10)	10	รหัสสัญญา
cust_id	CHAR(10)	10	ชื่อลูกค้า
car_id	CHAR(10)	10	ทะเบียนรถ
start_date	CHAR(10)	10	วันเริ่มเช่า
end_date	CHAR(10)	10	วันสิ้นสุด
days	INT	4	จำนวนวันที่เช่า
qty	INT	4	จำนวนรถที่เช่า
total_cents	FLOAT	8	ค่าใช้จ่ายรวม
status	CHAR(10)	10	active/closed

ตารางที่ 2.4 โครงสร้างข้อมูล Contracts

2.7.1 contract_id

เป็นรหัสประจำสัญญาเช่า ใช้ระบุเอกลักษณ์ของแต่ละสัญญาไม่ให้ซ้ำกัน ใช้เป็นคีย์หลัก (Primary Key) สำหรับเชื่อมโยงข้อมูลกับลูกค้าและรถยนต์ในระบบ

2.7.2 cust_id

เป็นรหัสลูกค้าที่ทำการเช่ารถ ข้อมูลนี้เชื่อมโยงกับตารางลูกค้า (Customers) เพื่อแสดงว่าผู้เช่ารถคือใคร

2.7.3 car_id

เป็นรหัสรถที่ถูกนำมาเช่า เพื่อระบุรถยนต์คันที่อยู่ในสัญญานั้น ๆ โดยข้อมูลนี้เชื่อมโยงกับตารางรถ (Cars)

2.7.4 start date

วันที่เริ่มต้นการเช่ารถ ใช้สำหรับคำนวณระยะเวลาเช่าและค่าใช้จ่าย

2.7.5 end date

วันที่สิ้นสุดการเช่ารถใช้ร่วมกับ start_date เพื่อคำนวณจำนวนวันที่เช่าจริงและค่าใช้จ่าย

รวม

2.7.8 days

จำนวนวันที่ทั้งหมดที่ลูกค้าในสัญญานั้น ซึ่งนำมาคำนวณร่วมกับจำนวนวันที่เช่าเพื่อหา
ค่าใช้จ่ายทั้งหมด

2.7.10 qty

จำนวนรถที่ลูกค้าเช่าในสัญญานั้น ใช้ในกรณีที่มีการเช่ามากกว่าหนึ่งคัน

2.7.9 total_cents

ค่าใช้จ่ายรวมของการเช่ารถ คำนวณจาก (จำนวนวันที่เช่า × ราคาเช่าต่อวัน) เป็นข้อมูล
สำคัญในสัญญาและรายงานการเช่า

2.7.11 status

สถานะของสัญญาเช่า เช่น “active” หมายถึงสัญญาที่ยังดำเนินการอยู่ หรือ “closed”
หมายถึงสัญญาที่สิ้นสุดและคืนรถเรียบร้อยแล้ว

3.ทะเบียนรถ / รุ่นรถ (Plate / Model) เช่น ป้าย 123, Toyota Camry

4. วันเริ่มต้นและวันสิ้นสุดการเช่า (Start Date – End Date) เช่น 2025-09-20 ถึง 2025-09-25

5. จำนวนวันเช่า (Days) เช่น 5 วัน

6. ราคาเช่าต่อวัน (Rate) เช่น 2500.00 บาท

7. ค่าใช้จ่ายรวม (Total Cost) เช่น 12500.00 บาท

8. สถานะสัญญา (Status) เช่น active (ยังเช่าอยู่) หรือ closed (คืนรถแล้ว)

9. ส่วนสรุปรายได้รวม (Summary of Total Revenue)

a. แสดงยอดรายได้จากการเช่ารถในแต่ละเดือน เช่น

i. เดือน 07/2025 รายได้รวม 12,000 บาท

ii. เดือน 08/2025 รายได้รวม 42,700 บาท

iii. เดือน 09/2025 รายได้รวม 63,700 บาท

b. แสดงผลรวมรายได้ทั้งหมดจากทุกเดือนรวมกัน

2.8.4 ส่วนสรุปลูกค้าที่เช่ารถในแต่ละเดือน (Customer Summary by Month)

a. แสดงรายชื่อลูกค้าที่เช่ารถในแต่ละเดือน เช่น

เดือน 07/2025 → มีลูกค้า 1 คน เช่น วินัย เติงชู

เดือน 08/2025 → มีลูกค้า 5 คน เช่น ชำนาญ ชำนิ, สันติ ใจดี ฯลฯ

เดือน 09/2025 → มีลูกค้า 6 คน เช่น ชำนาญ ชำนิ, ชัยชัย ใจสู้ ฯลฯ

b. สรุปจำนวนลูกค้าทั้งหมดในแต่ละเดือน

2.8.5 ส่วนสรุปรถที่ถูกเช่ามากที่สุด (Most Rented Cars Summary)

a. แสดงข้อมูลว่ารถรุ่นใดถูกเช่ามากที่สุดในแต่ละเดือน เช่น

เดือน 07/2025 → Toyota Camry ถูกเช่า 1 คัน

เดือน 08/2025 → Toyota Camry ถูกเช่า 5 คัน

เดือน 09/2025 → Toyota Camry ถูกเช่า 6 คัน

b. พร้อมสรุปจำนวนครั้งที่รถแต่ละรุ่นถูกเช่ารวมทั้งหมด

2.8.6 ส่วนสรุปข้อมูลทั้งหมดของระบบ (Overall Summary)

a. แสดงจำนวนสัญญาเช่าทั้งหมด เช่น 12 รายการ

b. แสดงจำนวนลูกค้าทั้งหมด เช่น 12 คน

c. แสดงจำนวนรถทั้งหมดที่อยู่ในระบบ เช่น 12 คัน

บทที่ 3

การใช้งานระบบจัดการข้อมูลการเช่ารถ

ระบบจัดการข้อมูลการเช่ารถที่พัฒนาขึ้นนี้ มีวัตถุประสงค์เพื่อช่วยให้การจัดเก็บและบริหารข้อมูลของธุรกิจเช่ารถมีความสะดวกและเป็นระบบมากขึ้น โดยผู้ใช้สามารถจัดการข้อมูลรถยนต์ ข้อมูลลูกค้า ข้อมูลสัญญาเช่า รวมถึงสร้างรายงานสรุปผลได้ภายในโปรแกรมเดียว ระบบถูกพัฒนาด้วยภาษา Python ซึ่งมีความยืดหยุ่นและใช้งานง่าย เหมาะสำหรับโครงงานในระดับนักศึกษา

เมื่อเปิดใช้งานโปรแกรม จะปรากฏเมนูหลักที่รวมฟังก์ชันต่าง ๆ ไว้ในหน้าเดียว ผู้ใช้สามารถเลือกเมนูได้ตามต้องการ เช่น เมนูเพิ่มข้อมูลรถสำหรับบันทึกข้อมูลรถรุ่นใหม่ ๆ เมนูเพิ่มข้อมูลลูกค้าเพื่อเก็บรายละเอียดของผู้เช่ารถ เมนูสร้างสัญญาเช่าสำหรับเชื่อมข้อมูลระหว่างลูกค้าและรถยนต์ที่เช่า รวมถึงเมนูค้นหาข้อมูลเพื่อเรียกดูข้อมูลที่ต้องการได้อย่างรวดเร็ว นอกจากนี้ยังมีส่วนของรายงานสรุปผลการเช่ารถ ซึ่งแสดงยอดรายได้ จำนวนลูกค้า และรถที่ถูกเช่ามากที่สุดในแต่ละเดือนให้ผู้ใช้งานได้ง่าย เมื่อใช้งานเสร็จสามารถกดออกจากระบบได้ทันทีโดยข้อมูลจะถูกบันทึกอัตโนมัติ สำหรับผู้ใช้งานโปรแกรม

3.1 การใช้งานโปรแกรมเช่ารถ

3.1.1 กรอกรหัสเลข 1 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชัน Manage Car เพิ่มข้อมูลประกอบไปด้วย Add, Update, Delete, View all, Back

```
===== Car Rent System (Fixed-length Binary) =====
1) Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู)
2) Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู)
3) Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู/กรอง)
4) Generate Report (สร้างไฟล์รายงาน report.txt)
0) Exit (ออก)
=====
เลือกเมนู:
```

ภาพที่ 3-1 การเลือกใช้งานฟังก์ชัน Manage Car

3.1.2 เมื่อเมนูฟังก์ชัน Manage Car ขึ้นมาแล้วจากนั้นก็สมารถระบุเมนูที่ต้องการเลือกได้

```

--- Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: █

```

ภาพที่ 3-2 เมนูของ Manage Car

3.1.3 กรอกหมายเลข 2 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชัน Manage Customers เพิ่มข้อมูลที่ประกอบไปด้วย Add, Update, Delete, View all, Back

```

===== Car Rent System (Fixed-length Binary) =====
1) Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู)
2) Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู)
3) Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู/กรอง)
4) Generate Report (สร้างไฟล์รายงาน report.txt)
0) Exit (ออก)
=====
เลือกเมนู:

```

ภาพที่3-3การเลือกใช้งานฟังก์ชัน Manage Customers

3.1.4 เมื่อเมนูฟังก์ชัน Manage Customers ขึ้นมาแล้วจากนั้นก็สมารถระบุเมนูที่ต้องการเลือกได้

```

--- Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: █

```

ภาพที่ 3-4 เมนูของ Manage Customers

3.1.5 กรอกหมายเลข 3 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชัน Manage Contracts เพิ่มข้อมูลประกอบไปด้วย Add, Update, Delete, View all, Viwe Filters, Back

```
===== Car Rent System (Fixed-length Binary) =====
1) Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู)
2) Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู)
3) Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู/กรอง)
4) Generate Report (สร้างไฟล์รายงาน report.txt)
0) Exit (ออก)
=====
เลือกเมนู:
```

ภาพที่ 3-5 การเลือกใช้งานฟังก์ชัน Manage Contracts

3.1.6 เมื่อเมนูฟังก์ชัน Manage Contracts ขึ้นมาแล้วจากนั้นก็สมารถระบุเมนูที่ต้องการเลือกได้

```
--- Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
5) View Filters (ดูแบบกรอง)
0) Back (กลับ)
เลือก: █
```

ภาพที่ 3-6 เมนูของ Manage Contracts

3.1.7 กรอกหมายเลข 4 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชัน Generate report เพื่อเพิ่มไฟล์ report.txt ที่สามารถเขียน report ถึงห้องสมุดได้

```
===== Car Rent System (Fixed-length Binary) =====
1) Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู)
2) Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู)
3) Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู/กรอง)
4) Generate Report (สร้างไฟล์รายงาน report.txt)
0) Exit (ออก)
=====
เลือกเมนู:
```

ภาพที่ 3-7 การเลือกใช้งานฟังก์ชัน Generate report

3.1.8 กรอกหมายเลข 5 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชัน Exit เพื่อออกจากโปรแกรม

```
===== Car Rent System (Fixed-length Binary) =====
1) Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู)
2) Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู)
3) Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู/กรอง)
4) Generate Report (สร้างไฟล์รายงาน report.txt)
0) Exit (ออก)
=====
เลือกเมนู:
```

ภาพที่ 3-8 การเลือกใช้งานฟังก์ชัน Exit

3.2 การใช้งานโปรแกรมเพิ่มข้อมูล

3.2.1 กรอกหมายเลข 1 เพื่อเพิ่มข้อมูลทั้งหมดของรถ

```
--- Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: 1
```

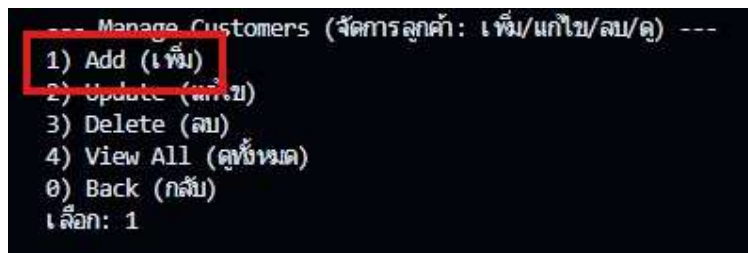
ภาพที่ 3-9 การเลือกใช้งานฟังก์ชัน Add (เพิ่ม)รถ

3.2.2 เมื่อกรอกเลือกหมายเลข 1 จะปรากฏหัวข้อการเพิ่มรถ(Add Car), Car ID, ยี่ห้อ (Brand), รุ่น(Model), ทะเบียน(Plate), ราคา/วัน หัวข้อทั้งหมดดังภาพที่ 3-10

```
เพิ่มรถ (Add Car)
Car ID (เช่น C0002): C0001
ยี่ห้อ (Brand): honda
รุ่น (Model): civic
ทะเบียน (Plate): กข1234
ราคา/วัน (เช่น 1500 หรือ 1500.50): 2000
บันทึกแล้ว
```

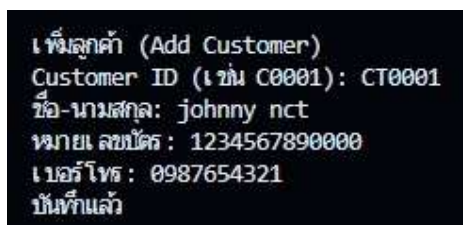
ภาพที่ 3-10 การเพิ่มรถ

3.2.3 กรอกหมายเลข 1 เพื่อเพิ่มข้อมูลลูกค้า



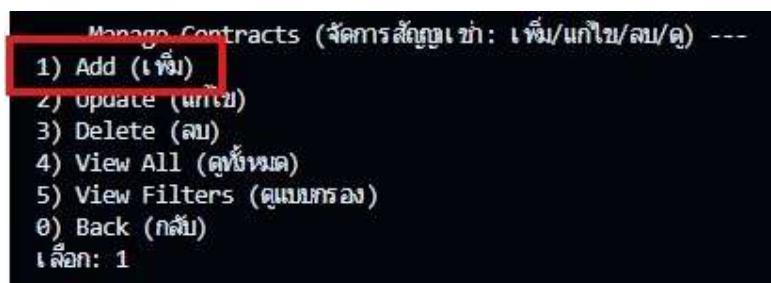
ภาพที่ 3-11 การเลือกใช้งานฟังก์ชัน Add (เพิ่ม)ข้อมูลลูกค้า

3.2.4 เมื่อกรอกหมายเลข 1 จะปรากฏหัวข้อการเพิ่มข้อมูลลูกค้า Add Customer, Customer ID,ชื่อ-นามสกุล, หมายเลขบัตร, เบอร์โทร ดังภาพที่ 3-12



ภาพที่ 3-12 การเพิ่มข้อมูลลูกค้า

3.2.5 กรอกลีอกหมายเลข 1 เพื่อเพิ่มข้อมูลของสัญญาเช่า



ภาพที่ 3-13 การเลือกใช้งานฟังก์ชัน Add (เพิ่ม)ข้อมูลลูกค้า

3.2.6 เมื่อกรอกหมายเลข 1 จะปรากฏหัวข้อการเพิ่มข้อมูลของสัญญาเช่า Add Contract, Contract ID, Customer ID, Car ID, วันเริ่ม, วันสิ้นสุด, Car Rente, สถานะ

```
เพิ่มสัญญาเช่า (Add Contract)
Contract ID (เช่น CNT001): 680001
Customer ID (ลูกค้าต้องมียู่ก่อน): CT0001
Car ID (รถต้องมียู่ก่อน): C0001
วันเริ่ม (YYYY-MM-DD): 2025-1-1
วันสิ้นสุด (YYYY-MM-DD): 2025-12-31
Cars Rente (จำนวนรถ 1-7): 1
สถานะ (active/closed): active
บันทึกแล้ว
```

ภาพที่ 3-14 การเพิ่มข้อมูลสัญญาเช่า

3.3 การใช้งานโปรแกรมการแก้ไขข้อมูล

3.3.1 กรอกหมายเลข 2 เพื่อแก้ไขข้อมูลรถ

```
--- Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: 2
```

ภาพที่ 3-15 การเลือกใช้งานฟังก์ชัน การแก้ไขข้อมูลรถ

3.3.2 เมื่อกรอกหมายเลข 2 จะปรากฏหัวข้อการ การแก้ไขข้อมูลของรถ

```
แก้ไขรถ (Update Car)
Car ID: 0
ยี่ห้อใหม่ (เดิม 0, ว้าง=ข้าม): toyota
รุ่นใหม่ (เดิม 0, ว้าง=ข้าม): cross
ทะเบียนใหม่ (เดิม 0, ว้าง=ข้าม): ยน1234
ราคา/วันใหม่ (เดิม 0.00, ว้าง=ข้าม): 1000
อัปเดตแล้ว
```

ภาพที่ 3-16 การแก้ไขข้อมูลของรถ

3.3.3 กรอกหมายเลข 2 เพื่อแก้ไขข้อมูลลูกค้า

```

--- Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: 2

```

ภาพที่ 3-17 การเลือกใช้งานฟังก์ชันการแก้ไขข้อมูลลูกค้า

3.3.4 เมื่อกรอกหมายเลข 2 จะปรากฏหัวข้อการ การแก้ไขข้อมูลของลูกค้า

```

แก้ไขลูกค้า (Update Customer)
Customer ID: 0
ชื่อใหม่ (เดิม 0, วาง=ข้าม): tommy jerry
เบอร์ใหม่ (เดิม 00, วาง=ข้าม): 0123456789000
โทรใหม่ (เดิม 0, วาง=ข้าม): 0897654321
อันนี้ คัดแล้ว

```

ภาพที่ 3-18 การแก้ไขข้อมูลลูกค้า

3.3.5 กรอกหมายเลข 2 เพื่อแก้ไขข้อมูลสัญญาเช่า

```

--- Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
5) View Filters (ดูแบบกรอง)
0) Back (กลับ)
เลือก: 2

```

ภาพที่ 3-19 การเลือกใช้งานฟังก์ชันการแก้ไขข้อมูลสัญญาเช่า

3.3.6 เมื่อกรอกหมายเลข 2 จะปรากฏหัวข้อการ การแก้ไขข้อมูลของสัญญาเช่า

```
แก้ไขสัญญาเช่า (Update Contract)
Contract ID: 0
วันสิ้นสุดใหม่ (เดิม 0, วาง=ข้าม): 2025-12-30
สถานะใหม่ (เดิม active, วาง=ข้าม):
จำนวนรถใหม่ 1-7 (เดิม 1, วาง=ข้าม): 2
รูปแบบวันที่ไม่ถูกต้อง (ข้าม)
อันใดแล้ว
```

ภาพที่ 3-20 การแก้ไขข้อมูลลูกค้า

3.4 การใช้งานโปรแกรมลบข้อมูล

3.4.1 กรอกหมายเลข 3 เพื่อลบข้อมูลรถ

```
--- Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: 3
```

ภาพที่ 3-21 การเลือกใช้งานฟังก์ชันการลบข้อมูลรถ

3.4.2 เมื่อกรอกหมายเลข 3 จะปรากฏหัวข้อการ การลบข้อมูลของรถ

```
ลบรถ (Delete Car: soft-delete)
Car ID: 0
ลบแล้ว (active=0)
```

ภาพที่ 3-22 การลบข้อมูลรถ

3.4.3 กรอกหมายเลข 3 เพื่อลบข้อมูลลูกค้า

```

--- Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: 3

```

ภาพที่ 3-23 การเลือกใช้งานฟังก์ชันการลบข้อมูลลูกค้า

3.4.4 เมื่อกรอกหมายเลข 3 จะปรากฏหัวข้อการ การลบข้อมูลของลูกค้า

```

ลบลูกค้า (Delete Customer: soft-delete)
Customer ID: 0
ลบแล้ว (active=0)

```

ภาพที่ 3-24 การลบข้อมูลลูกค้า

3.4.5 กรอกหมายเลข 3 เพื่อลบข้อมูลสัญญาเช่า

```

--- Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
5) View Filters (ดูแบบกรอง)
0) Back (กลับ)
เลือก: 3

```

ภาพที่ 3-25 การเลือกใช้งานฟังก์ชันการลบข้อมูลสัญญาเช่า

3.4.6 เมื่อกรอกหมายเลข 3 จะปรากฏหัวข้อการ การลบข้อมูลของสัญญาเช่า

```

ลบสัญญา (Delete Contract: soft-delete)
Contract ID: 0
ลบแล้ว (active=0)

```

ภาพที่ 3-26 การลบข้อมูลสัญญาเช่า

3.5 การใช้งานโปรแกรมแสดงข้อมูล

3.5.1 กรอกรหัสหมายเลข 4 เพื่อแสดงข้อมูลรถทั้งหมด

```

--- Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: 4

```

ภาพที่ 3-27 การเลือกใช้งานฟังก์ชันการแสดงข้อมูลรถทั้งหมด

3.5.2 เมื่อกรอกรหัสหมายเลข 4 จะปรากฏหัวข้อการแสดงข้อมูลรถทั้งหมด

```

รายการรถ (Cars)
[001] C0001 | honda civic | กข1234 | 2,000.00/day
[002] C0002 | toyota crmry | พท1234 | 2,400.00/day

```

ภาพที่ 3-28 การแสดงข้อมูลรถทั้งหมด

3.5.3 กรอกรหัสหมายเลข 4 เพื่อแสดงข้อมูลลูกค้าทั้งหมด

```

--- Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: 4

```

ภาพที่ 3-29 การเลือกใช้งานฟังก์ชันการแสดงข้อมูลลูกค้าทั้งหมด

3.5.4 เมื่อกรอกรหัสหมายเลข 4 จะปรากฏหัวข้อการแสดงข้อมูลลูกค้าทั้งหมด

```

รายการลูกค้า (Customers)
[000] CT0001 | johnny nct | 1234567890000 | 0987654321

```

ภาพที่ 3-30 การแสดงข้อมูลลูกค้าทั้งหมด

3.5.5 กรอกหมายเลข 4 เพื่อแสดงข้อมูลสัญญาเช่าทั้งหมด

```

--- Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
5) View Filters (ดูแบบกรอง)
0) Back (กลับ)
เลือก: 4

```

ภาพที่ 3-31 การเลือกใช้งานฟังก์ชันการแสดงผลข้อมูลสัญญาเช่าทั้งหมด

3.5.6 เมื่อกรอกหมายเลข 4 จะปรากฏหัวข้อการแสดงผลข้อมูลสัญญาเช่าทั้งหมด

```

รายการสัญญาเช่า (Contracts)
[000] 680001 | cust CT0001 | car C0001 | 2025-1-1..2025-12-31 | 364d x1 | 728,000.00
[002] 680002 | cust CT0001 | car C0002 | 25-2-21..2025-6-25 | 0d x1 | 0.00

```

ภาพที่ 3-32 การแสดงผลข้อมูลสัญญาเช่าทั้งหมด

3.6 เรียกดูสัญญาเช่าแบบกรอง

3.6.1 กรอกหมายเลข 5 เพื่อแสดงข้อมูลสัญญาเช่าแบบกรอง เมื่อกรอกแล้วจะปรากฏหัวข้อให้เลือก แต่ละหัวข้อจะแสดงข้อมูลที่ต่างกันตามสิ่งที่ต้องการจะดู

```

--- Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
5) View Filters (ดูแบบกรอง)
0) Back (กลับ)
เลือก: 5

```

ภาพที่ 3-33 การเลือกใช้งานฟังก์ชันการแสดงผลข้อมูลสัญญาเช่าแบบกรอง

3.6.2 เลือกเลข 1 จะดูตามปีและเดือน

```

คู่มือกรอง (Filters)
1) ตามเดือน (YYYY-MM) 2) ตามสถานะ (active/closed) 3) ตามรหัสลูกค้า 4) ตามรหัสรถ
เลือก: 1
เดือน (YYYY-MM): 2
เดือน (YYYY-MM): 2025-2
พบ 1 รายการ
680001 2025-1-1..2025-12-31 CT0001->C0001 x1 active total 728,000.00

```

ภาพที่ 3-34 ปีและเดือน

3.6.3 เลือกเลข 2 ดูตามสถานะ

```

คู่มือกรอง (Filters)
1) ตามเดือน (YYYY-MM) 2) ตามสถานะ (active/closed) 3) ตามรหัสลูกค้า 4) ตามรหัสรถ
เลือก: 2
สถานะ: active
สถานะ: active
พบ 2 รายการ
680001 2025-1-1..2025-12-31 CT0001->C0001 x1 active total 728,000.00
680002 25-2-21..2025-6-25 CT0001->C0002 x1 active total 0.00

```

ภาพที่ 3-35 สถานะ

3.6.4 เลือกเลข 3 ดูตามรหัสลูกค้าที่เช่ารถ

```

คู่มือกรอง (Filters)
1) ตามเดือน (YYYY-MM) 2) ตามสถานะ (active/closed) 3) ตามรหัสลูกค้า 4) ตามรหัสรถ
เลือก: 3
Customer ID: CT0001
Customer ID: CT0002
พบ 1 รายการ
680001 2025-1-1..2025-12-31 CT0001->C0001 x1 active total 728,000.00

```

ภาพที่ 3-36 รหัสลูกค้าที่เช่ารถ

3.6.5 เลือกเลข 4 ดูตามรหัสรถที่ถูกเช่า

```

คู่มือกรอง (Filters)
1) ตามเดือน (YYYY-MM) 2) ตามสถานะ (active/closed) 3) ตามรหัสลูกค้า 4) ตามรหัสรถ
เลือก: 4
Car ID: C0002
Car ID: C0002
พบ 1 รายการ
680002 25-2-21..2025-6-25 CT0001->C0002 x1 active total 0.00

```

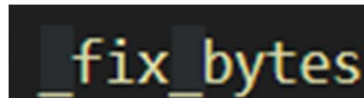
ภาพที่ 3-37 รหัสรถที่ถูกเช่า

บทที่ 4

อธิบายการทำงานของ Code

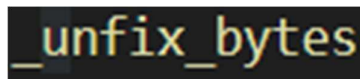
4.1 ฟังก์ชันไบนารีพื้นฐานโปรแกรมระบบเช่ารถ (Car Rental System)

4.1.1 ฟังก์ชันยูทิลิตี้ (Utility Function) ที่ทำหน้าที่ ผกผัน (Inversion) การดำเนินการของหรือฟังก์ชันการทำความสะอาดข้อมูลไบต์อื่น ๆ ที่อาจมีการนำอักขระพิเศษหรืออักขระควบคุมบางอย่างออกไป วัตถุประสงค์หลักของ คือ การกู้คืน (Restoration) ชุดข้อมูลไบต์ให้อยู่ในรูปแบบที่เหมาะสมสำหรับการใช้งานภายนอกระบบ หรือเพื่อการแสดงผลขั้นสุดท้ายต่อผู้ใช้งาน โดยเฉพาะอย่างยิ่งในสถานการณ์ที่ข้อมูลถูกนำออกมาจากฐานข้อมูลหรือพื้นที่จัดเก็บไฟล์ (File Storage) ที่มีการทำความสะอาดอย่างเข้มงวดแล้ว

A screenshot of a code editor showing the text `_fix_bytes` in a yellow monospace font on a dark background.

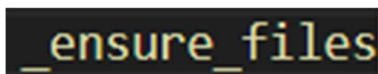
ภาพที่ 4-1 `_fix_bytes`

4.1.2 ฟังก์ชัน `_unfix_bytes` ในภาษา Python ก็ไม่ใช่ฟังก์ชันมาตรฐานที่มีมาพร้อมกับภาษาแต่เป็นฟังก์ชันยูทิลิตี้ที่กำหนดเอง (Custom Utility Function) ซึ่งมักถูกสร้างขึ้นให้ทำหน้าที่ ผกผัน (Inverse) หรือ กู้คืน (Restore) การทำงานของ `def _fix_bytes` หาก ใช้ทำความสะอาดข้อมูลก่อนจัดเก็บ ก็ใช้ เตรียมข้อมูลที่ถูกล้างทำความสะอาดแล้วให้กลับมาอยู่ในรูปแบบที่พร้อมแสดงผล หรือส่งออกต่อ

A screenshot of a code editor showing the text `_unfix_bytes` in a yellow monospace font on a dark background.

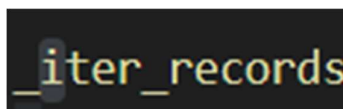
ภาพที่ 4-2 `_unfix_bytes`

4.1.3 ฟังก์ชัน `_ensure_files()` ในภาษา Python ไม่ใช่ฟังก์ชันมาตรฐาน แต่เป็น ฟังก์ชันยูทิลิตีที่ผู้พัฒนาสร้างขึ้นเอง (Custom Utility Function) ซึ่งบทบาทหลักของมันคือการ ตรวจสอบและเตรียมความพร้อมของไฟล์หรือไดเรกทอรีที่จำเป็นต่อระบบ ก่อนที่จะมีการดำเนินการอ่านหรือเขียนข้อมูลใด ๆ



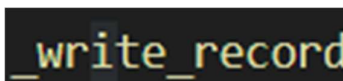
ภาพที่ 4-3 `_ensure_files()`

4.1.4 ฟังก์ชัน `_iter_records()` ในภาษา Python ไม่ใช่ฟังก์ชันมาตรฐาน แต่เป็น ฟังก์ชันยูทิลิตีที่กำหนดเอง (Custom Utility Function) ซึ่งบทบาทหลักคือการทำหน้าที่เป็น ตัววนซ้ำ (Iterator) เพื่อดึงหรือประมวลผลข้อมูล ทีละเรคคอร์ด (Record by Record) จากแหล่งข้อมูลภายใน



ภาพที่ 4-4 `_iter_records()`

4.1.5 ฟังก์ชัน `_write_record()` ในภาษา Python ไม่ใช่ฟังก์ชันมาตรฐาน แต่เป็น ฟังก์ชันยูทิลิตีที่ผู้พัฒนาสร้างขึ้นเอง (Custom Utility Function) ซึ่งมีบทบาทหลักคือการ บันทึก (Write) หรือ จัดเก็บ (Store) ข้อมูลที่ถูกประมวลผลแล้ว ทีละเรคคอร์ด ลงในแหล่งข้อมูลปลายทาง



ภาพที่ 4-5 `_write_record()`

4.1.6 ฟังก์ชัน `def _detect_contract_rec_size()` ทำหน้าที่ตรวจสอบขนาดเรกคอร์ดของไฟล์สัญญา (`contracts.bin`) เพื่อระบุว่าไฟล์ดังกล่าวอยู่ในรูปแบบ “รุ่นเก่า” หรือ “รุ่นใหม่” โดยอ้างอิงจากขนาดของข้อมูลที่เก็บประโยชน์ ช่วยให้ระบบสามารถอ่านไฟล์ข้อมูลที่สร้างจากโปรแกรมรุ่นก่อนหน้าได้โดยไม่เกิดข้อผิดพลาด

```
def _detect_contract_rec_size()
```

ภาพที่ 4-6 `def _detect_contract_rec_size()`

4.1.7 ฟังก์ชัน `def migrate_contracts_if_old()` ทำหน้าที่อัปเดตไฟล์ข้อมูลสัญญาเข้ารุ่นเก่าให้กลายเป็นรุ่นใหม่โดยอัตโนมัติ เพื่อให้ระบบสามารถทำงานร่วมกับไฟล์ข้อมูลที่เคยถูกสร้างไว้ก่อนหน้าได้

```
def migrate_contracts_if_old()
```

ภาพที่ 4-7 `def _detect_contract_rec_size()`

4.2 ฟังก์ชันจัดการข้อมูลระบบเช่ารถ

4.2.1 ฟังก์ชัน add_car() ฟังก์ชัน add_car() มีหน้าที่เพิ่มข้อมูลรถยนต์ใหม่เข้าสู่ระบบเช่ารถ โดยรับข้อมูลจากผู้ใช้งาน เช่น รหัสรถ ยี่ห้อ รุ่น ทะเบียน และราคาเช่าต่อวัน จากนั้นตรวจสอบความถูกต้องของข้อมูลก่อนบันทึกลงในไฟล์ไบนารี cars.bin ขั้นตอนการทำงานเริ่มจากการแสดงข้อความ “เพิ่มรถ (Add Car)” เพื่อแจ้งให้ผู้ใช้ทราบ จากนั้นรับรหัสรถ (car_id) และตรวจสอบว่ารหัสว่างหรือซ้ำกับข้อมูลเดิมหรือไม่ หากไม่ผ่านเงื่อนไข ระบบจะหยุดการทำงานเพื่อป้องกันความผิดพลาด เมื่อรหัสผ่านการตรวจสอบแล้ว โปรแกรมจะรับข้อมูลยี่ห้อ รุ่น และทะเบียนรถ พร้อมจำกัดความยาวของแต่ละข้อมูลให้ตรงกับโครงสร้างของไฟล์ ถัดมาจะรับราคาเช่าต่อวันและตรวจสอบให้แน่ใจว่าผู้ใช้กรอกตัวเลขถูกต้อง โดยแปลงค่าราคาจากหน่วยบาทเป็นสตางค์เพื่อความแม่นยำในการเก็บข้อมูล เมื่อได้รับข้อมูลครบถ้วน ระบบจะสร้างออบเจกต์ของคลาส Car แล้วเรียกใช้เมธอด pack() เพื่อแปลงข้อมูลให้อยู่ในรูปแบบไบนารี (bytes) ก่อนเรียกฟังก์ชัน _write_record() เพื่อบันทึกข้อมูลลงในไฟล์ cars.bin เมื่อบันทึกเสร็จ ระบบจะแสดงข้อความ “บันทึกแล้ว” เพื่อยืนยันว่าการเพิ่มข้อมูลสำเร็จ ฟังก์ชันนี้จึงเป็นส่วนสำคัญของกระบวนการเพิ่มข้อมูล (Create) ในระบบเช่ารถยนต์ ช่วยให้ข้อมูลรถถูกจัดเก็บอย่างเป็นระเบียบ ถูกต้อง และพร้อมสำหรับการใช้งานในส่วนอื่นของโปรแกรม เช่น การทำสัญญาเช่าหรือการออกรายงาน

```
def add_car():
    print("\nเพิ่มรถ (Add Car)")
    car_id = input("Car ID (เช่น C0002): ").strip()[:6]
    if not car_id: print("รหัสห้ามว่าง"); return
    if _find_car_by_id(car_id): print("มีรหัสนี้แล้ว"); return
    brand = input("ยี่ห้อ (Brand): ").strip()[:16]
    model = input("รุ่น (Model): ").strip()[:18]
    plate = input("ทะเบียน (Plate): ").strip()[:12]
    while True:
        rate = input("ราคา/วัน (เช่น 1500 หรือ 1500.50): ").strip()
        try:
            rate_cents = int(round(float(rate)*100)); break
        except: print("กรอกตัวเลขอีกครั้ง")
    rec = Car(1,car_id,brand,model,plate,rate_cents)
    _write_record(CAR_FILE, CAR_SIZE, None, rec.pack())
    print("บันทึกแล้ว")
```

ภาพที่ 4-8 ฟังก์ชัน add_car()

4.2.2 ฟังก์ชัน `update_car()` ทำหน้าที่แก้ไขข้อมูลรถยนต์ที่มีอยู่แล้วในระบบเช่ารถ โดยให้ผู้ใช้ระบุรหัสรถ (Car ID) ที่ต้องการแก้ไข จากนั้นโปรแกรมจะค้นหาข้อมูลเดิมในไฟล์ `cars.bin` และแสดงข้อมูลเก่าเพื่อให้ผู้ใช้สามารถกรอกข้อมูลใหม่แทนที่เดิมได้ เช่น ยี่ห้อ รุ่น ทะเบียน และราคาเช่าต่อวัน กระบวนการทำงานเริ่มจากการแสดงข้อความ “แก้ไขรถ (Update Car)” เพื่อแจ้งให้ผู้ใช้ทราบ แล้วรับรหัสรถยนต์จากคีย์บอร์ด จากนั้นระบบจะเรียกใช้ฟังก์ชัน `_find_car_by_id(car_id)` เพื่อค้นหาข้อมูลรถตามรหัสที่กรอก หากไม่พบข้อมูล ระบบจะแสดงข้อความ “ไม่พบ” และหยุดการทำงานของฟังก์ชันโดยทันที เพื่อป้องกันการแก้ไขข้อมูลที่ไม่มีอยู่ในระบบ หากพบข้อมูลรถ ระบบจะดึงค่าข้อมูลเดิม เช่น ยี่ห้อ (brand), รุ่น (model), ทะเบียน (plate) และราคาต่อวัน (rate_cents) มาแสดงในวงเล็บ เพื่อให้ผู้ใช้เห็นข้อมูลเก่าและกรอกข้อมูลใหม่ได้สะดวก หากผู้ใช้ไม่กรอกข้อมูลใหม่ ระบบจะคงค่าข้อมูลเดิมไว้ แต่ถ้ามีการกรอกใหม่ โปรแกรมจะอัปเดตเฉพาะฟิลด์นั้น ๆ โดยใช้การตรวจสอบแบบมีเงื่อนไข เช่น

1. ถ้ามีการกรอกยี่ห้อใหม่ จะอัปเดตค่าของ `c.brand`

2. ถ้ามีการกรอกรุ่นใหม่ จะอัปเดตค่าของ `c.model`

3. ถ้ามีการกรอกทะเบียนใหม่ จะอัปเดตค่าของ `c.plate` และในกรณีที่มีการกรอกราคาเช่าใหม่ ระบบจะพยายามแปลงค่าที่กรอกให้เป็นตัวเลขทศนิยม (float) คูณด้วย 100 เพื่อแปลงหน่วยเป็นสตางค์ จากนั้นเก็บเป็นจำนวนเต็ม (int) หากรูปแบบราคาที่กรอกไม่ถูกต้อง เช่น ใส่ตัวอักษรแทนตัวเลข ระบบจะแสดงข้อความ “รูปแบบราคาไม่ถูกต้อง (ข้าม)” และจะไม่แก้ไขราคาส่วนนั้น

เมื่อปรับปรุงข้อมูลเสร็จเรียบร้อยแล้ว ฟังก์ชันจะเรียกใช้ `_write_record(CAR_FILE, CAR_SIZE, idx, c.pack())` เพื่อเขียนข้อมูลที่แก้ไขแล้วกลับไปยังตำแหน่งเดิมของไฟล์ (`cars.bin`) โดยใช้ตัวแปร `idx` ซึ่งเป็นตำแหน่งของข้อมูลรถคันนั้นในไฟล์ และใช้เมธอด `pack()` เพื่อแปลงข้อมูลออบเจกต์ให้เป็นรูปแบบไบนารี (bytes) ก่อนบันทึกจริง สุดท้าย โปรแกรมจะแสดงข้อความ “อัปเดตแล้ว” เพื่อแจ้งให้ผู้ใช้ทราบว่า การแก้ไขข้อมูลรถยนต์เสร็จสมบูรณ์ ฟังก์ชันนี้จึงมีความสำคัญในการปรับปรุงข้อมูลรถยนต์ให้เป็นปัจจุบันอยู่เสมอ โดยไม่ต้องลบและเพิ่มข้อมูลใหม่ ช่วยลดความซ้ำซ้อนและคงความถูกต้องของข้อมูลในระบบเช่ารถยนต์

```
def update_car():
    print("\nแก้ไขรถ (Update Car)")
    car_id = input("Car ID: ").strip()[:6]
    f = _find_car_by_id(car_id)
    if not f: print("ไม่พบ"); return
    idx, c = f
    brand = input(f"ยี่ห้อใหม่ (เดิม {c.brand}, วาง=ข้าม): ").strip()
    model = input(f"รุ่นใหม่ (เดิม {c.model}, วาง=ข้าม): ").strip()
    plate = input(f"ทะเบียนใหม่ (เดิม {c.plate}, วาง=ข้าม): ").strip()
    rate = input(f"ราคา/วันใหม่ (เดิม {c.rate_cents/100:.2f}, วาง=ข้าม): ").strip()
    if brand: c.brand = brand[:16]
    if model: c.model = model[:18]
    if plate: c.plate = plate[:12]
    if rate:
        try: c.rate_cents = int(round(float(rate)*100))
        except: print("รูปแบบราคาไม่ถูกต้อง (ข้าม)")
    _write_record(CAR_FILE, CAR_SIZE, idx, c.pack())
    print("อัปเดตแล้ว")
```

ภาพที่ 4-9 ฟังก์ชัน update_car()

4.2.3 ฟังก์ชัน delete_car() ทำหน้าที่ลบข้อมูลรถยนต์ออกจากระบบเช่ารถ โดยใช้รูปแบบการลบแบบซอฟต์ดีลิต (Soft Delete) ซึ่งหมายถึง การไม่ลบข้อมูลออกจากไฟล์จริง แต่เปลี่ยนสถานะของรถคันนั้นให้ไม่สามารถใช้งานได้แทน การทำงานเริ่มต้นด้วยการแสดงข้อความ “ลบรถ (Delete Car)” เพื่อแจ้งผู้ใช้ จากนั้นระบบจะให้กรอกรหัสรถ (Car ID) ที่ต้องการลบ และเรียกใช้ฟังก์ชัน _find_car_by_id() เพื่อค้นหาข้อมูลรถในไฟล์ cars.bin หากไม่พบข้อมูล ระบบจะแสดงข้อความ “ไม่พบ” และยุติการทำงานทันที หากพบข้อมูลรถ ระบบจะดึงตำแหน่ง (idx) และออบเจกต์ของรถ (c) ที่ค้นเจอ แล้วทำการเปลี่ยนค่าฟิลด์ active ของรถคันนั้นจากค่าเดิมให้เป็น 0 เพื่อระบุว่ารถคันนี้ถูกปิดการใช้งานหรือไม่พร้อมให้เช่า หลังจากนั้นจะเรียกใช้ฟังก์ชัน _write_record() เพื่อเขียนข้อมูลที่แก้ไขกลับไปยังตำแหน่งเดิมในไฟล์ cars.bin โดยใช้คำสั่ง c.pack() เพื่อแปลงข้อมูลให้อยู่ในรูปแบบไบนารีก่อนบันทึกจริง เมื่อดำเนินการเสร็จ ระบบจะแสดงข้อความ “ลบแล้ว (active=0)” เพื่อยืนยันว่าการเปลี่ยนสถานะสำเร็จ ฟังก์ชันนี้ช่วยให้สามารถจัดการการลบข้อมูลได้อย่างปลอดภัย โดยไม่ทำให้ข้อมูลเดิมสูญหาย และยังสามารถกู้คืนหรืออ้างอิงข้อมูลย้อนหลังได้ในภายหลัง

```
def delete_car():
    print("\nลบรถ (Delete Car: soft-delete)")
    car_id = input("Car ID: ").strip()[:6]
    f = _find_car_by_id(car_id)
    if not f: print("ไม่พบ"); return
    idx, c = f
    c.active = 0
    _write_record(CAR_FILE, CAR_SIZE, idx, c.pack())
    print("ลบแล้ว (active=0)")
```

ภาพที่ 4-10 ฟังก์ชัน delete_car()

4.2.4 ฟังก์ชัน add_customer() มีหน้าที่เพิ่มข้อมูลลูกค้าใหม่เข้าสู่ระบบเช่ารถ โดยรับข้อมูลจากผู้ใช้งาน เช่น รหัสลูกค้า ชื่อ-นามสกุล หมายเลขบัตรประชาชน และหมายเลขโทรศัพท์ จากนั้นตรวจสอบว่ารหัสลูกค้าว่างหรือซ้ำกับข้อมูลเดิมหรือไม่ หากรหัสไม่ซ้ำและถูกต้อง ระบบจะดำเนินการบันทึกข้อมูลลงในไฟล์ไบนารี customers.bin ก่อนบันทึก โปรแกรมจะสร้างออบเจกต์ของคลาส Customer และใช้เมธอด pack() เพื่อแปลงข้อมูลให้อยู่ในรูปแบบไบต์ (bytes) จากนั้นเรียกใช้ฟังก์ชัน _write_record() เพื่อเขียนข้อมูลใหม่ต่อท้ายไฟล์ เมื่อบันทึกเสร็จสิ้น ระบบจะแสดงข้อความ “บันทึกแล้ว” เพื่อยืนยันผลการทำงาน ฟังก์ชันนี้ช่วยให้ระบบสามารถเพิ่มข้อมูลลูกค้าได้อย่างถูกต้อง ป้องกันข้อมูลซ้ำซ้อน และเตรียมข้อมูลสำหรับการเช่ารถหรือการทำสัญญาในขั้นตอนต่อไป

```
def add_customer():
    print("\nเพิ่มลูกค้า (Add Customer)")
    cid = input("Customer ID (เช่น C0001): ").strip()[:6]
    if not cid: print("รหัสห้ามว่าง"); return
    if _find_customer_by_id(cid): print("มีรหัสนี้แล้ว"); return
    name = input("ชื่อ-นามสกุล: ").strip()[:48]
    idcard = input("หมายเลขบัตร: ").strip()[:24]
    phone = input("เบอร์โทร: ").strip()[:16]
    rec = Customer(1, cid, name, idcard, phone)
    _write_record(CUSTOMER_FILE, CUS_SIZE, None, rec.pack())
    print("บันทึกแล้ว")
```

ภาพที่ 4-11 ฟังก์ชัน add_customer()

4.2.5 ฟังก์ชัน `update_customer()` มีหน้าที่แก้ไขข้อมูลลูกค้าที่มีอยู่แล้วในระบบเช่ารถ โดยให้ผู้ระบุรหัสลูกค้า (Customer ID) ที่ต้องการแก้ไข จากนั้นโปรแกรมจะค้นหาข้อมูลลูกค้าจากไฟล์ `customers.bin` ผ่านฟังก์ชัน `_find_customer_by_id()` หากไม่พบข้อมูล ระบบจะแสดงข้อความ “ไม่พบ” และหยุดการทำงานทันทีเพื่อป้องกันข้อผิดพลาด เมื่อพบข้อมูลลูกค้า โปรแกรมจะแสดงรายละเอียดเดิมของลูกค้าคนนั้น เช่น ชื่อ-นามสกุล หมายเลขบัตรประชาชน และหมายเลขโทรศัพท์ เพื่อให้ผู้ใช้สามารถกรอกข้อมูลใหม่แทนที่ได้ โดยผู้ใช้สามารถเลือกแก้ไขเฉพาะส่วนที่ต้องการ ส่วนใดที่ไม่กรอกข้อมูลใหม่ ระบบจะคงค่าข้อมูลเดิมไว้ จากนั้นโปรแกรมจะตรวจสอบความยาวของข้อมูลแต่ละส่วน เช่น ชื่อไม่เกิน 48 ตัวอักษร หมายเลขบัตรไม่เกิน 24 ตัวอักษร และเบอร์โทรศัพท์ไม่เกิน 16 ตัวอักษร เพื่อให้สอดคล้องกับขนาดของเรกคอร์ดในไฟล์ไบนารี

เมื่อผู้ใช้กรอกข้อมูลเรียบร้อยแล้ว ระบบจะทำการอัปเดตค่าข้อมูลในออบเจกต์ของคลาส `Customer` และใช้เมธอด `pack()` เพื่อแปลงข้อมูลให้อยู่ในรูปแบบของไบต์ (bytes) ก่อนเรียกฟังก์ชัน `_write_record()` เพื่อเขียนข้อมูลใหม่ทับข้อมูลเดิมลงในไฟล์ `customers.bin` โดยอ้างอิงจากตำแหน่ง (idx) ของลูกค้าที่ถูกแก้ไข เมื่อกระบวนการอัปเดตเสร็จสิ้น โปรแกรมจะแสดงข้อความ “อัปเดตแล้ว” เพื่อยืนยันผลการแก้ไข ฟังก์ชันนี้จึงมีบทบาทสำคัญในการปรับปรุงข้อมูลลูกค้าให้เป็นปัจจุบัน ช่วยลดความผิดพลาดและรักษาความถูกต้องของข้อมูลในระบบเช่ารถยนต์

```
def update_customer():
    print("\nแก้ไขลูกค้า (Update Customer)")
    cid = input("Customer ID: ").strip()[:6]
    f = _find_customer_by_id(cid)
    if not f: print("ไม่พบ"); return
    idx, c = f
    name = input(f"ชื่อใหม่ (เดิม {c.name}, วาง=ข้าม): ").strip()
    idc = input(f"บัตรใหม่ (เดิม {c.idcard}, วาง=ข้าม): ").strip()
    tel = input(f"โทรใหม่ (เดิม {c.phone}, วาง=ข้าม): ").strip()
    if name: c.name = name[:48]
    if idc: c.idcard = idc[:24]
    if tel: c.phone = tel[:16]
    _write_record(CUSTOMER_FILE, CUS_SIZE, idx, c.pack())
    print("อัปเดตแล้ว")
```

ภาพที่ 4-12 ฟังก์ชัน `update_customer()`

4.2.6 ฟังก์ชัน `delete_customer()` มีหน้าที่ลบข้อมูลลูกค้าออกจากระบบเช่ารถ โดยใช้วิธี “การลบแบบซอฟต์ดีลิต (Soft Delete)” คือไม่ลบข้อมูลออกจากไฟล์โดยตรง แต่เปลี่ยนสถานะของลูกค้าคนนั้นให้ไม่สามารถใช้งานได้แทน การทำงานเริ่มจากการแสดงข้อความ “ลบลูกค้า (Delete Customer)” เพื่อแจ้งให้ผู้ใช้ทราบว่ากำลังเข้าสู่โหมดการลบข้อมูล จากนั้นระบบจะรับรหัสลูกค้า (Customer ID) ที่ต้องการลบ และใช้ฟังก์ชัน `_find_customer_by_id()` เพื่อตรวจสอบว่ารหัสดังกล่าวมีอยู่ในไฟล์ `customers.bin` หรือไม่ หากไม่พบข้อมูล ระบบจะแสดงข้อความ “ไม่พบ” และยุติการทำงานทันที แต่หากพบข้อมูล ระบบจะดึงตำแหน่ง (`idx`) และออฟเจ็กต์ลูกค้า (`c`) ที่ตรงกับรหัสนั้นออกมา จากนั้นทำการเปลี่ยนค่าฟิลด์ `active` ของลูกค้าคนนั้นให้เป็นค่า 0 เพื่อระบุว่าบัญชีลูกค้าถูกปิดการใช้งานแล้ว หลังจากแก้ไขค่าเรียบร้อยแล้ว โปรแกรมจะเรียกใช้เมธอด `pack()` เพื่อแปลงข้อมูลให้อยู่ในรูปแบบของไบต์ (bytes) และเรียกใช้ฟังก์ชัน `_write_record()` เพื่อเขียนข้อมูลที่แก้ไขกลับไปยังตำแหน่งเดิมในไฟล์

เมื่อระบบดำเนินการเสร็จสิ้น โปรแกรมจะแสดงข้อความ “ลบแล้ว (`active=0`)” เพื่อยืนยันว่าการลบข้อมูลสำเร็จ ฟังก์ชันนี้ช่วยให้ระบบสามารถจัดการข้อมูลลูกค้าได้อย่างปลอดภัย โดยไม่สูญเสียข้อมูลเดิม ทำให้สามารถกู้คืนหรืออ้างอิงข้อมูลลูกค้าในภายหลังได้ ซึ่งเป็นแนวทางที่เหมาะสมสำหรับระบบที่ต้องการรักษาความถูกต้องของประวัติการเช่ารถในระยะยาว

```
def delete_customer():
    print("\nลบลูกค้า (Delete Customer: soft-delete)")
    cid = input("Customer ID: ").strip()[:6]
    f = _find_customer_by_id(cid)
    if not f: print("ไม่พบ"); return
    idx, c = f
    c.active = 0
    _write_record(CUSTOMER_FILE, CUS_SIZE, idx, c.pack())
    print("ลบแล้ว (active=0)")
```

ภาพที่ 4-13 ฟังก์ชัน `delete_customer()`

4.2.7 ฟังก์ชัน `add_contract()` มีหน้าที่เพิ่มข้อมูลสัญญาเช่ารถใหม่เข้าสู่ระบบ โดยเชื่อมโยงข้อมูลระหว่างลูกค้าและรถยนต์ที่มีอยู่ในระบบ เพื่อจัดเก็บรายละเอียดของการเช่าในไฟล์ไบนารี `contracts.bin` การทำงานเริ่มจากการเรียกใช้ฟังก์ชัน `migrate_contracts_if_old()` เพื่อปรับรูปแบบไฟล์สัญญาให้เป็นรุ่นล่าสุด จากนั้นจะแสดงข้อความ “เพิ่มสัญญาเช่า (Add Contract)” เพื่อแจ้งผู้ใช้ให้ทราบก่อนเริ่มกรอกข้อมูล โปรแกรมจะรับรหัสสัญญา (`contract_id`) จากผู้ใช้ เช่น “CNT001” และตรวจสอบว่ารหัสนี้ว่างหรือซ้ำกับข้อมูลเดิมในไฟล์หรือไม่ หากซ้ำระบบจะแสดงข้อความ “รหัสนี้มีอยู่แล้ว” และหยุดการทำงานเพื่อป้องกันความซ้ำซ้อน ต่อมาจะให้ผู้กรอกข้อมูลรหัสลูกค้า (`cust_id`) และรหัสรถ (`car_id`) เพื่อเชื่อมโยงข้อมูลกับลูกค้าและรถในระบบ โดยฟังก์ชัน `_find_car_by_id()` จะตรวจสอบว่ามีรถคันนั้นอยู่หรือไม่ หากไม่พบจะขึ้นข้อความ “ไม่พบรถ” และหยุดการทำงานทันที

เมื่อผ่านขั้นตอนตรวจสอบ ระบบจะให้ผู้กรอกวันเริ่มต้น (`start`) และวันสิ้นสุด (`end`) ของการเช่าในรูปแบบ `YYYY-MM-DD` จากนั้นคำนวณจำนวนวันเช่าด้วยการนำวันสิ้นสุดลบวันเริ่มต้น หากกรอกวันที่ไม่ถูกต้อง ระบบจะตั้งค่าเริ่มต้นเป็น 0 วัน ต่อมาจะให้ผู้กรอกจำนวนรถที่เช่า (`qty`) ซึ่งต้องอยู่ในช่วง 1–7 คัน โดยระบบจะตรวจสอบความถูกต้องของค่าที่กรอกในลูป `while` เพื่อให้แน่ใจว่าข้อมูลเป็นตัวเลขที่อยู่ในช่วงที่กำหนด หลังจากได้รับข้อมูลครบ ระบบจะนำราคาค่าเช่าต่อวัน (`rate_cents`) มาคำนวณเป็นราคารวม (`total`) โดยใช้สูตร **`total = days × price × qty`** จากนั้นให้ผู้ระบุสถานะของสัญญา (`status`) ว่าเป็น “active” หรือ “closed” แล้วสร้างออบเจกต์ของคลาส `Contract` เพื่อจัดเก็บข้อมูลทั้งหมด ก่อนแปลงข้อมูลให้อยู่ในรูปแบบไบต์ด้วยเมธอด `pack()` และบันทึกลงไฟล์ `contracts.bin` ผ่านฟังก์ชัน `_write_record()`

เมื่อการบันทึกเสร็จสิ้น โปรแกรมจะแสดงข้อความ “บันทึกแล้ว” เพื่อยืนยันผลการทำงาน ฟังก์ชันนี้จึงเป็นฟังก์ชันสำคัญที่ใช้ในการบันทึกข้อมูลการเช่ารถอย่างครบถ้วน ช่วยให้ระบบสามารถติดตาม ตรวจสอบ และจัดการสัญญาเช่ารถได้อย่างถูกต้องและมีประสิทธิภาพ

```

def add_contract():
    migrate_contracts_if_old()
    print("\nเพิ่มสัญญาเช่า (Add Contract)")
    contract_id = input("Contract ID (เช่น CNT001): ").strip()[:8]
    if not contract_id: print("รหัสห้ามว่าง"); return
    if _find_contract_by_id(contract_id): print("รหัสนี้มีอยู่แล้ว"); return
    cust_id = input("Customer ID (ลูกค้าต้องมีอยู่แล้ว): ").strip()[:6]
    car_id = input("Car ID (รถต้องมีอยู่แล้ว): ").strip()[:6]
    car = _find_car_by_id(car_id)
    if not car: print("ไม่พบรถ"); return
    start = input("วันเริ่ม (YYYY-MM-DD): ").strip()[:10]
    end = input("วันสิ้นสุด (YYYY-MM-DD): ").strip()[:10]
    # days
    try:
        sd = datetime.strptime(start, "%Y-%m-%d").date()
        ed = datetime.strptime(end, "%Y-%m-%d").date()
        days = max(0, (ed-sd).days)
    except: days = 0
    # qty 1..7
    while True:
        q = input("Cars Rente (จำนวนรถ 1-7): ").strip()
        try:
            qty = int(q)
            if 1 <= qty <= 7: break
            else: print("ต้องอยู่ในช่วง 1..7")
        except: print("กรอกตัวเลข 1..7")
    _, car_obj = car
    price = car_obj.rate_cents
    total = days * price * qty
    status = (input("สถานะ (active/closed): ").strip() or "active")[:8]
    rec = Contract(1, contract_id, cust_id, car_id, start, end, days, price, total, qty, status)
    _write_record(CONTRACT_FILE, CON_SIZE_NEW, None, rec.pack())
    print("บันทึกแล้ว")

```

ภาพที่ 4-14 ฟังก์ชัน add_contract()

4.2.8 ฟังก์ชัน update_contract() มีหน้าที่แก้ไขข้อมูลของสัญญาเช่ารถที่มีอยู่แล้วในระบบ โดยเริ่มต้นจากการเรียกใช้ฟังก์ชัน migrate_contracts_if_old() เพื่อให้แน่ใจว่าไฟล์ข้อมูลสัญญา (contracts.bin) อยู่ในรูปแบบใหม่ที่สามารถใช้งานได้ จากนั้นโปรแกรมจะแสดงข้อความ “แก้ไขสัญญาเช่า (Update Contract)” เพื่อแจ้งให้ผู้ใช้ทราบ และรับรหัสสัญญา (contract_id) จากผู้ใช้เพื่อตรวจสอบว่ามีข้อมูลอยู่ในระบบหรือไม่ หากไม่พบข้อมูล ระบบจะแสดงข้อความ “ไม่พบ” และยุติการทำงานทันที เมื่อพบข้อมูลสัญญา ระบบจะดึงข้อมูลเก่ามาแสดง เช่น วันที่สิ้นสุด (end), สถานะ (status) และจำนวนรถที่เช่า (qty) จากนั้นเปิดให้ผู้ใช้กรอกข้อมูลใหม่แทนที่เดิม หากผู้ใช้ไม่กรอกข้อมูลใหม่ในบางส่วน ระบบจะคงค่าข้อมูลเดิมไว้ การแก้ไขข้อมูลวันที่สิ้นสุดจะถูกตรวจสอบด้วยคำสั่ง try-except เพื่อป้องกันรูปแบบวันที่ผิดพลาด โดยระบบจะคำนวณจำนวนวันเช่า (days) ใหม่อัตโนมัติจากส่วนต่างของวันเริ่มต้นและวันสิ้นสุด ในส่วนของสถานะสัญญา (status) หากผู้ใช้กรอกข้อมูลใหม่ ระบบจะอัปเดตค่าฟิลด์ดังกล่าวทันที เช่น เปลี่ยนจาก “active” เป็น “closed” สำหรับกรณีที่สัญญาสิ้นสุดแล้ว ส่วนการแก้ไขจำนวนรถที่เช่า (qty) จะตรวจสอบให้แน่ใจว่าค่าที่กรอกเป็นตัวเลขและอยู่ในช่วง 1-7 คัน หากไม่ตรงตามเงื่อนไข โปรแกรมจะแสดงข้อความเตือน

เพื่อให้กรอกใหม่ เมื่อการแก้ไขข้อมูลเสร็จสิ้น ระบบจะคำนวณราคารวม (total_cents) ใหม่ โดยนำจำนวนวันเข้ามาคูณกับราคาเช่าต่อวันและจำนวนรถที่เช่า จากนั้นใช้เมธอด pack() เพื่อแปลงข้อมูลให้อยู่ในรูปแบบไบต์ (bytes) และเรียกฟังก์ชัน _write_record() เพื่อเขียนข้อมูลใหม่ทับตำแหน่งเดิมในไฟล์ contracts.bin สุดท้ายโปรแกรมจะแสดงข้อความ “อัปเดตแล้ว” เพื่อยืนยันว่าการแก้ไขข้อมูลสำเร็จ ฟังก์ชันนี้จึงเป็นส่วนสำคัญที่ช่วยให้ระบบสามารถปรับปรุงข้อมูลสัญญาเช่าได้อย่างถูกต้อง รักษาความสอดคล้องของข้อมูล และเพิ่มความยืดหยุ่นในการจัดการข้อมูลในระบบเช่ารถยนต์

```
def update_contract():
    migrate_contracts_if_old()
    print("\nแก้ไขสัญญาเช่า (Update Contract)")
    contract_id = input("Contract ID: ").strip()[:8]
    f = _find_contract_by_id(contract_id)
    if not f: print("ไม่พบ"); return
    idx, c = f
    new_end = input(f"วันสิ้นสุดใหม่ (เดิม {c.end}, วาง=ข้าม): ").strip()
    new_status = input(f"สถานะใหม่ (เดิม {c.status}, วาง=ข้าม): ").strip()
    new_qty = input(f"จำนวนรถใหม่ 1-7 (เดิม {c.qty}, วาง=ข้าม): ").strip()
    if new_end:
        try:
            sd = datetime.strptime(c.start, "%Y-%m-%d").date()
            ed = datetime.strptime(new_end, "%Y-%m-%d").date()
            c.end = new_end[:10]
            c.days = max(0, (ed-sd).days)
        except: print("รูปแบบวันที่ไม่ถูกต้อง (ข้าม)")
    if new_status: c.status = new_status[:8]
    if new_qty:
        try:
            q = int(new_qty)
            if 1 <= q <= 7:
                c.qty = q
            else:
                print("จำนวนรถต้องอยู่ในช่วง 1..7 (ข้าม)")
        except:
            print("รูปแบบจำนวนรถไม่ถูกต้อง (ข้าม)")
    # recompute total
    c.total_cents = c.days * c.price_per_day_cents * max(1, int(c.qty))
    _write_record(CONTRACT_FILE, CON_SIZE_NEW, idx, c.pack())
    print("อัปเดตแล้ว")
```

ภาพที่ 4-15 ฟังก์ชัน update_contract()

4.2.9 ฟังก์ชัน `delete_contract()` มีหน้าที่ลบข้อมูลสัญญาเช่ารถออกจากระบบ โดยใช้รูปแบบ “การลบแบบซอฟต์ดีลิต (Soft Delete)” ซึ่งหมายถึงการเปลี่ยนสถานะของข้อมูลให้ไม่ใช้งาน โดยไม่ลบข้อมูลจริงออกจากไฟล์ เพื่อให้ยังสามารถเก็บประวัติของสัญญาเช่าไว้ใช้อ้างอิงภายหลังได้ การทำงานเริ่มต้นด้วยการเรียกใช้ฟังก์ชัน `migrate_contracts_if_old()` เพื่อปรับรูปแบบไฟล์สัญญา (`contracts.bin`) ให้เป็นรุ่นล่าสุด จากนั้นโปรแกรมจะแสดงข้อความ “ลบสัญญา (Delete Contract)” เพื่อแจ้งผู้ใช้และรับรหัสสัญญา (`contract_id`) ที่ต้องการลบ เมื่อได้รับรหัสแล้ว ระบบจะตรวจสอบว่ารหัสที่กรอกมีอยู่จริงในระบบหรือไม่ โดยเรียกใช้ฟังก์ชัน `_find_contract_by_id()` หากไม่พบข้อมูล ระบบจะแสดงข้อความ “ไม่พบ” และหยุดการทำงานทันทีเพื่อป้องกันข้อผิดพลาด แต่หากพบข้อมูลสัญญา ระบบจะนำตำแหน่งของข้อมูล (`idx`) และออบเจกต์สัญญา (`c`) ที่ตรงกับรหัสดังกล่าวออกมาใช้งาน จากนั้นทำการเปลี่ยนสถานะของสัญญา (`active`) ให้เป็นค่า 0 ซึ่งหมายถึง “ไม่ใช้งาน” หรือ “ถูกลบ” หลังจากเปลี่ยนสถานะเรียบร้อยแล้ว โปรแกรมจะใช้เมธอด `pack()` เพื่อแปลงข้อมูลให้อยู่ในรูปแบบไบนารี (bytes) และเรียกใช้ฟังก์ชัน `_write_record()` เพื่อบันทึกการเปลี่ยนแปลงกลับลงในไฟล์ `contracts.bin` ที่ตำแหน่งเดิมของข้อมูลนั้น เมื่อบันทึกเสร็จสิ้น ระบบจะแสดงข้อความ “ลบแล้ว (active=0)” เพื่อยืนยันว่าการลบข้อมูลสำเร็จ ฟังก์ชันนี้ช่วยให้ระบบสามารถจัดการการลบข้อมูลสัญญาได้อย่างปลอดภัย โดยไม่กระทบต่อประวัติการเช่ารถที่บันทึกไว้เดิม อีกทั้งยังสามารถกู้คืนข้อมูลกลับมาได้หากจำเป็นในอนาคต จึงเป็นวิธีที่เหมาะสมสำหรับระบบที่ต้องการความถูกต้องและต่อเนื่องของข้อมูลในระยะยาว

```
def delete_contract():
    migrate_contracts_if_old()
    print("\nลบสัญญา (Delete Contract: soft-delete)")
    contract_id = input("Contract ID: ").strip()[:8]
    f = _find_contract_by_id(contract_id)
    if not f: print("ไม่พบ"); return
    idx, c = f
    c.active = 0
    _write_record(CONTRACT_FILE, CON_SIZE_NEW, idx, c.pack())
    print("ลบแล้ว (active=0)")
```

ภาพที่ 4-16 ฟังก์ชัน `delete_contract()`

4.2.10 ฟังก์ชัน `view_filter()` มีหน้าที่ใช้สำหรับค้นหาและแสดงรายการสัญญาเช่ารถตามเงื่อนไขที่ผู้ใช้กำหนด โดยมีวัตถุประสงค์เพื่อช่วยให้ผู้ดูแลระบบสามารถตรวจสอบข้อมูลสัญญาได้อย่างสะดวกและรวดเร็ว การทำงานเริ่มจากการตรวจสอบขนาดของเรกคอร์ดในไฟล์ข้อมูลด้วยฟังก์ชัน `_detect_contract_rec_size()` เพื่อให้ทราบว่าข้อมูลอยู่ในรูปแบบใหม่หรือเก่า จากนั้นแสดงเมนูตัวเลือกให้ผู้ใช้เลือกวิธีการกรองข้อมูล เช่น กรองตามเดือน (YYYY-MM), ตามสถานะ (active/closed), ตามรหัสลูกค้า หรือรหัสรถยนต์ หลังจากผู้ใช้เลือกเงื่อนไขการกรอง ระบบจะเปิดไฟล์ `contracts.bin` ในโหมดอ่านข้อมูลแบบไบนารี (rb) แล้ววนลูปอ่านข้อมูลแต่ละเรกคอร์ดออกมาโดยใช้เมธอด `unpack_new()` หรือ `unpack_old()` จากคลาส `Contract` เพื่อแปลงข้อมูลให้อยู่ในรูปแบบที่สามารถนำไปประมวลผลได้ หากเรกคอร์ดใดมีสถานะ `active` ไม่เท่ากับ 1 หมายความว่าสัญญานั้นถูกปิดหรือถูกลบไปแล้ว ระบบจะข้ามการประมวลผลของข้อมูลนั้นทันที ระบบจะทำการตรวจสอบข้อมูลแต่ละเรกคอร์ดตามเงื่อนไขที่ผู้ใช้เลือก เช่น หากเลือกกรองตามเดือน ระบบจะเปรียบเทียบวันที่เริ่มต้น (start) ของสัญญากับเดือนที่ผู้ใช้กรอก หากตรงกันจะบันทึกไว้ในตัวแปร `results` ในกรณีกรองตามสถานะ ระบบจะตรวจสอบค่าฟิลด์ `status` ส่วนกรณีกรองตามรหัสลูกค้า หรือรหัสรถ ระบบจะเปรียบเทียบค่ากับฟิลด์ `cust_id` หรือ `car_id` ตามลำดับ

```
def view_filter():
    rec_size = _detect_contract_rec_size()
    print("\nดูแบบกรอง (Filters)")
    print("1) ตามเดือน (YYYY-MM) 2) ตามสถานะ (active/closed) 3) ตามรหัสลูกค้า 4) ตามรหัสรถ")
    ch = input("เลือก: ").strip()
    results: List[Contract] = []
    with open(CONTRACT_FILE, "rb") as f:
        while True:
            b = f.read(rec_size)
            if not b or len(b) < rec_size: break
            c = Contract.unpack_new(b) if rec_size == CON_SIZE_NEW else Contract.unpack_old(b)
            if c.active != 1: continue
            ok = False
            if ch == "1":
                key = input("เดือน (YYYY-MM): ").strip()[:7]
                ok = c.start.startswith(key)
            elif ch == "2":
                st = input("สถานะ: ").strip()[:8]
                ok = (c.status == st)
            elif ch == "3":
                who = input("Customer ID: ").strip()[:6]
                ok = (c.cust_id == who)
            elif ch == "4":
                which = input("Car ID: ").strip()[:6]
                ok = (c.car_id == which)
            else:
                print("ยกเลิก"); return
            if ok: results.append(c)
    print(f"พบ {len(results)} รายการ")
    for c in results:
        print(f"{c.contract_id} {c.start}..{c.end} {c.cust_id}->{c.car_id} x{c.qty} {c.status} total {c.total_cents/100:,.2f}")
```

ภาพที่ 4-17 ฟังก์ชัน `view_filter()`

4.3 ฟังก์ชันเมนูจัดการข้อมูลระบบเช่ารถ

4.3.1 ฟังก์ชัน `manage_cars_menu()` มีหน้าที่จัดการเมนูหลักสำหรับการดำเนินงานที่เกี่ยวข้องกับข้อมูลรถยนต์ภายในระบบเช่ารถ โดยใช้โครงสร้างการทำงานแบบวนลูป `while True:` เพื่อให้ผู้ใช้สามารถเลือกดำเนินการซ้ำได้หลายครั้งโดยไม่ต้องกลับไปหน้าแรกของโปรแกรมทุกครั้งที่การทำงานเริ่มต้นด้วยการแสดงเมนูคำสั่งหลักให้ผู้ใช้เลือก ซึ่งประกอบด้วยตัวเลือกหลัก 5 รายการ ได้แก่ “Add” สำหรับเพิ่มข้อมูลรถใหม่, “Update” สำหรับแก้ไขข้อมูลรถที่มีอยู่, “Delete” สำหรับลบข้อมูลรถ, “View All” สำหรับดูรายการรถทั้งหมด และ “Back” สำหรับกลับไปยังเมนูหลักของระบบ เมื่อผู้ใช้กรอกค่าตัวเลือก (ch) ระบบจะตรวจสอบค่าที่กรอกผ่านคำสั่งเงื่อนไข `if-elif-else` โดยแต่ละตัวเลือกจะเรียกใช้ฟังก์ชันที่เกี่ยวข้อง เช่น `add_car()` สำหรับเพิ่มข้อมูลรถใหม่, `update_car()` สำหรับแก้ไขข้อมูลเดิม, `delete_car()` สำหรับลบข้อมูลรถแบบซอฟต์แวร์ดีลิต และ `list_cars()` สำหรับแสดงรายการรถทั้งหมดที่มีในระบบ หากผู้ใช้เลือก “0” ระบบจะใช้คำสั่ง `return` เพื่อออกจากลูปและกลับไปยังเมนูก่อนหน้า หากผู้ใช้กรอกค่าที่ไม่ถูกต้อง ระบบจะแสดงข้อความ “เลือกไม่ถูกต้อง” เพื่อแจ้งเตือนและให้ผู้ใช้กรอกใหม่อีกครั้ง โดยไม่ทำให้โปรแกรมหยุดทำงาน ฟังก์ชันนี้จึงเป็นจุดเชื่อมโยงหลักในการเรียกใช้ฟังก์ชันการจัดการรถทั้งหมดในระบบ ช่วยให้งานของผู้ใช้เป็นระบบระเบียบ เข้าใจง่าย และสามารถควบคุมการดำเนินการกับข้อมูลรถยนต์ได้อย่างครบวงจรในทีเดียว

```
def manage_cars_menu():
    while True:
        ch = input("""
--- Manage Cars (จัดการรถ: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: """).strip()
        if ch=="1": add_car()
        elif ch=="2": update_car()
        elif ch=="3": delete_car()
        elif ch=="4": list_cars()
        elif ch=="0": return
        else: print("เลือกไม่ถูกต้อง")
```

ภาพที่ 4-18 ฟังก์ชัน `manage_cars_menu()`

4.3.2 ฟังก์ชัน `manage_customers_menu()` มีหน้าที่จัดการเมนูหลักของส่วนข้อมูลลูกค้าในระบบเช่ารถ โดยควบคุมการเพิ่ม แก้ไข ลบ และแสดงข้อมูลลูกค้าผ่านคำสั่งต่าง ๆ ที่ผู้ใช้เลือก ฟังก์ชันนี้ใช้โครงสร้างการทำงานแบบวนซ้ำ `while True:` เพื่อให้ผู้ใช้สามารถเลือกดำเนินการหลายครั้งได้โดยไม่ต้องกลับไปหน้าจอหลักทุกครั้ง การทำงานเริ่มจากการแสดงเมนูตัวเลือกบนหน้าจอ ซึ่งประกอบด้วย 5 รายการหลัก ได้แก่ “Add” สำหรับเพิ่มข้อมูลลูกค้าใหม่, “Update” สำหรับแก้ไขข้อมูลเดิม, “Delete” สำหรับลบข้อมูลลูกค้า, “View All” สำหรับแสดงรายชื่อลูกค้าทั้งหมด และ “Back” สำหรับกลับไปยังเมนูก่อนหน้า เมื่อผู้ใช้กรอกหมายเลขตัวเลือก (ch), โปรแกรมจะตรวจสอบค่าที่ได้รับผ่านคำสั่งเงื่อนไข `if-elif-else` เพื่อเรียกใช้ฟังก์ชันที่สอดคล้องกัน เช่น `add_customer()` สำหรับบันทึกข้อมูลลูกค้าใหม่, `update_customer()` สำหรับปรับปรุงข้อมูลที่มีอยู่, `delete_customer()` สำหรับเปลี่ยนสถานะลูกค้าเป็นไม่ใช้งาน และ `list_customers()` สำหรับแสดงข้อมูลลูกค้าทั้งหมดในระบบ หากผู้ใช้เลือก “0” ระบบจะใช้คำสั่ง `return` เพื่อออกจากลูปและกลับไปยังเมนูหลักของโปรแกรม

ในกรณีที่ผู้ใช้กรอกค่าที่ไม่ถูกต้อง เช่น ตัวอักษรหรือตัวเลขที่อยู่นอกช่วงที่กำหนด ระบบจะแสดงข้อความ “เลือกไม่ถูกต้อง” เพื่อแจ้งเตือนและให้ผู้ใช้ป้อนข้อมูลใหม่โดยไม่ทำให้โปรแกรมหยุดทำงาน ฟังก์ชันนี้จึงเป็นส่วนสำคัญในการควบคุมการเข้าถึงและจัดการข้อมูลลูกค้าอย่างเป็นระบบ มีความยืดหยุ่น และใช้งานได้ง่ายสำหรับผู้ดูแลระบบ

```
def manage_customers_menu():
    while True:
        ch = input("""
--- Manage Customers (จัดการลูกค้า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
0) Back (กลับ)
เลือก: """).strip()
        if ch=="1": add_customer()
        elif ch=="2": update_customer()
        elif ch=="3": delete_customer()
        elif ch=="4": list_customers()
        elif ch=="0": return
        else: print("เลือกไม่ถูกต้อง")
```

ภาพที่ 4-19 ฟังก์ชัน `manage_customers_menu()`

4.3.3 ฟังก์ชัน `manage_contracts_menu()` มีหน้าที่เป็นเมนูหลักในการจัดการข้อมูลสัญญาเช่ารถของระบบ โดยใช้โครงสร้างการทำงานแบบวนซ้ำ `while True`: เพื่อให้ผู้ใช้สามารถเลือกดำเนินการได้ต่อเนื่องโดยไม่ต้องออกจากโปรแกรม การทำงานเริ่มจากการแสดงเมนูคำสั่งให้ผู้ใช้เลือก ซึ่งประกอบด้วยตัวเลือกหลัก 6 รายการ ได้แก่ “Add” สำหรับเพิ่มข้อมูลสัญญาใหม่, “Update” สำหรับแก้ไขข้อมูลสัญญาเดิม, “Delete” สำหรับลบสัญญา, “View All” สำหรับแสดงรายการสัญญาทั้งหมด, “View Filters” สำหรับดูข้อมูลสัญญาแบบกรองตามเงื่อนไข และ “Back” สำหรับกลับไปยังเมนูหลักของระบบ หลังจากแสดงเมนู ระบบจะรอรับค่าจากผู้ใช้ผ่านตัวแปร `ch` และตรวจสอบคำตอบโดยใช้คำสั่งเงื่อนไข `if-elif-else` ซึ่งแต่ละตัวเลือกจะเรียกใช้ฟังก์ชันย่อยที่เกี่ยวข้อง เช่น `add_contract()` เพื่อเพิ่มข้อมูลสัญญาใหม่, `update_contract()` สำหรับแก้ไขข้อมูลสัญญาที่มีอยู่, `delete_contract()` สำหรับลบสัญญาออกจากระบบแบบซอฟต์แวร์, `list_contracts()` เพื่อดูข้อมูลสัญญาทั้งหมด และ `view_filter()` สำหรับแสดงรายการสัญญาตามเงื่อนไขที่กำหนด เช่น ตามเดือน สถานะ ลูกค้า หรือรหัสรถยนต์ หากผู้ใช้กรอกค่า “0” ระบบจะทำการ `return` เพื่อออกจากลูปและกลับไปยังเมนูหลัก แต่หากกรอกค่าที่ไม่ถูกต้อง โปรแกรมจะแสดงข้อความ “เลือกไม่ถูกต้อง” เพื่อแจ้งให้ทราบและกลับไปยังเมนูอีกครั้งโดยไม่หยุดการทำงาน ฟังก์ชันนี้จึงเป็นศูนย์กลางในการควบคุมและเรียกใช้งานคำสั่งย่อยทั้งหมดของระบบจัดการสัญญาเช่า ช่วยให้ผู้ใช้สามารถบริหารจัดการข้อมูลสัญญาได้ครบวงจรในจุดเดียว สะดวก รวดเร็ว และมีความเป็นระบบ

```
def manage_contracts_menu():
    while True:
        ch = input("""
--- Manage Contracts (จัดการสัญญาเช่า: เพิ่ม/แก้ไข/ลบ/ดู) ---
1) Add (เพิ่ม)
2) Update (แก้ไข)
3) Delete (ลบ)
4) View All (ดูทั้งหมด)
5) View Filters (ดูแบบกรอง)
0) Back (กลับ)
เลือก: """).strip()
        if ch=="1": add_contract()
        elif ch=="2": update_contract()
        elif ch=="3": delete_contract()
        elif ch=="4": list_contracts()
        elif ch=="5": view_filter()
        elif ch=="0": return
        else: print("เลือกไม่ถูกต้อง")
```

ภาพที่ 4-20 ฟังก์ชัน `manage_contracts_menu()`

4.4 ฟังก์ชันสร้างรายงาน (Report Generation)

4.4.1 ฟังก์ชัน `generate_report()` มีหน้าที่จัดเตรียมและรวบรวมข้อมูลเพื่อสร้างรายงานสรุปผลการดำเนินงานของระบบเช่ารถ โดยเฉพาะข้อมูลที่เกี่ยวข้องกับลูกค้า รถยนต์ และสัญญาเช่าที่มีอยู่ในระบบ การทำงานเริ่มจากการเรียกใช้ฟังก์ชัน `migrate_contracts_if_old()` เพื่อปรับปรุงรูปแบบไฟล์ข้อมูลสัญญาให้เป็นรุ่นใหม่ล่าสุด เพื่อให้มั่นใจว่าข้อมูลทั้งหมดสามารถอ่านและประมวลผลได้อย่างถูกต้อง จากนั้น โปรแกรมจะโหลดข้อมูลของลูกค้าและรถยนต์ที่ยังอยู่ในสถานะใช้งาน (`active=1`) เข้ามาเก็บในหน่วยความจำ โดยใช้โครงสร้างข้อมูลแบบพจนานุกรม (Dictionary) เพื่อให้เข้าถึงข้อมูลได้อย่างรวดเร็ว สำหรับข้อมูลลูกค้า โปรแกรมจะอ่านข้อมูลจากไฟล์ `CUSTOMER_FILE` ผ่านฟังก์ชัน `_iter_records()` และใช้เมธอด `Customer.unpack()` เพื่อแปลงข้อมูลจากรูปแบบไบนารีให้อยู่ในรูปแบบของออบเจกต์ลูกค้า (`Customer`) จากนั้นจะบันทึกเฉพาะลูกค้าที่มีสถานะเปิดใช้งาน (`active=1`) ลงในพจนานุกรม `customers` โดยใช้รหัสลูกค้า (`cid`) เป็นคีย์ ในส่วนของข้อมูลรถยนต์ โปรแกรมจะดำเนินการในลักษณะเดียวกัน โดยอ่านข้อมูลจากไฟล์ `CAR_FILE` และใช้เมธอด `Car.unpack()` เพื่อแปลงข้อมูล จากนั้นเก็บเฉพาะรถยนต์ที่ยังเปิดใช้งานไว้ในพจนานุกรม `cars` โดยใช้รหัสรถ (`car_id`) เป็นคีย์ การจัดเก็บข้อมูลในรูปแบบนี้ช่วยให้ระบบสามารถเข้าถึงและเชื่อมโยงข้อมูลระหว่างลูกค้า รถยนต์ และสัญญาเช่าได้สะดวก รวดเร็ว และมีประสิทธิภาพ ฟังก์ชันนี้จึงเป็นส่วนเริ่มต้นของกระบวนการสร้างรายงาน ซึ่งทำหน้าที่เตรียมข้อมูลพื้นฐานทั้งหมดให้พร้อมสำหรับการสรุปผลในขั้นตอนถัดไป เช่น การคำนวณยอดรวม รายได้ หรือการแสดงรายละเอียดการเช่าในรูปแบบรายงานสรุป เพื่อใช้ในการวิเคราะห์และตรวจสอบข้อมูลของระบบเช่ารถยนต์โดยรวม

```
def _fmt_money(cents: int) -> str: return f"{cents/100:,.2f}"

def generate_report():
    migrate_contracts_if_old()
    # Load active sets
    customers: Dict[str, Customer] = {}
    for _, b in _iter_records(CUSTOMER_FILE, CUS_SIZE):
        c = Customer.unpack(b)
        if c.active == 1: customers[c.cid] = c
    cars: Dict[str, Car] = {}
    for _, b in _iter_records(CAR_FILE, CAR_SIZE):
        c = Car.unpack(b)
        if c.active == 1: cars[c.car_id] = c
```

ภาพที่ 4-21 ฟังก์ชัน `generate_report()` ใช้รวบรวมข้อมูล

4.4.2 ในส่วนนี้ของฟังก์ชัน `generate_report()` มีหน้าที่หลักในการอ่านและโหลดข้อมูลสัญญาเช่ารถจากไฟล์เก็บข้อมูลเข้าสู่หน่วยความจำ เพื่อใช้ในการประมวลผลและจัดทำรายงานในขั้นตอนถัดไป โดยเริ่มจากการสร้างตัวแปร `contracts` ซึ่งเป็นลิสต์ (List) สำหรับเก็บออบเจกต์ของสัญญา (Contract) ที่อ่านได้จากไฟล์ข้อมูล จากนั้นใช้คำสั่ง `for` เพื่อวนลูปอ่านข้อมูลแต่ละระเบียนจากไฟล์ `CONTRACT_FILE` โดยอ้างอิงขนาดของเรคคอร์ดตามค่าคงที่ `CON_SIZE_NEW` ซึ่งเป็นรูปแบบข้อมูลสัญญาในรุ่นใหม่ของระบบในแต่ละรอบของการวนลูป โปรแกรมจะตรวจสอบว่าข้อมูลที่อ่าน (b) ว่างหรือมีขนาดน้อยกว่าที่กำหนดหรือไม่ หากเงื่อนไขใดเป็นจริง โปรแกรมจะหยุดการอ่านข้อมูลทันที (break) เพื่อป้องกันการประมวลผลข้อมูลที่ไม่สมบูรณ์หรือเกิดข้อผิดพลาด จากนั้นจะใช้เมธอด `Contract.unpack_new(b)` เพื่อแปลงข้อมูลจากรูปแบบไบนารี (bytes) ให้กลายเป็นออบเจกต์ของคลาส `Contract` ซึ่งสามารถเข้าถึงฟิลด์ต่าง ๆ เช่น รหัสสัญญา รหัสลูกค้า รหัสรถยนต์ วันที่เริ่มต้น วันที่สิ้นสุด สถานะการเช่า และยอดรวมค่าเช่า

หลังจากแปลงข้อมูลเรียบร้อยแล้ว โปรแกรมจะตรวจสอบสถานะของสัญญาด้วยเงื่อนไข `if c.active == 1:` เพื่อเลือกเฉพาะสัญญาที่อยู่ในสถานะเปิดใช้งาน (active) เท่านั้น แล้วจึงนำข้อมูลดังกล่าวไปเก็บในลิสต์ `contracts` ซึ่งจะใช้เป็นฐานข้อมูลสำคัญในการสร้างรายงานในภายหลัง เช่น รายการสัญญาที่ยังดำเนินการอยู่ หรือยอดรวมรายได้จากการเช่ารถที่ยังไม่ปิดสัญญา

ขั้นตอนนี้จึงถือเป็นกระบวนการสำคัญของการเตรียมข้อมูลในระบบรายงาน ทำให้ข้อมูลที่ถูกนำมาวิเคราะห์และแสดงผลมีความถูกต้อง ครบถ้วน และอยู่ในรูปแบบที่พร้อมสำหรับการสรุปผลทางสถิติในขั้นตอนสุดท้ายของการสร้างรายงานระบบเช่ารถยนต์

```
# Read contracts (new format after possible migration)
contracts: List[Contract] = []
for _,b in _iter_records(CONTRACT_FILE, CON_SIZE_NEW):
    if not b or len(b)<CON_SIZE_NEW: break
    c=Contract.unpack_new(b)
    if c.active==1: contracts.append(c)
```

ภาพที่ 4-22 ฟังก์ชัน `generate_report()` ใช้อ่านและโหลดข้อมูลสัญญาเช่ารถ

4.4.3 ส่วนนี้ของฟังก์ชัน `generate_report()` มีหน้าที่กำหนดรูปแบบและโครงสร้างของรายงานสรุปข้อมูลระบบเช่ารถ โดยเริ่มจากการกำหนดคอลัมน์ (cols) ที่ต้องการแสดงในรายงาน ซึ่งครอบคลุมข้อมูลสำคัญทั้งหมดของสัญญาเช่า เช่น รหัสสัญญา (Contract ID), ชื่อลูกค้า (Name), หมายเลขบัตรประชาชน (ID Card), หมายเลขโทรศัพท์ (Phone), หมายเลขทะเบียนรถ (Car Plate), รุ่นรถ (Car), วันที่เริ่มต้นและสิ้นสุดการเช่า (Start Date และ End Date), จำนวนรถที่เช่า (Cars Rented), ราคาต่อวัน (Price/Day), ราคารวม (Total Price) และสถานะ (Status) ในแต่ละคอลัมน์ จะมีการกำหนดความกว้าง (width) และรูปแบบการจัดตำแหน่งข้อความ (align) เพื่อให้ข้อมูลในรายงานจัดวางอย่างเป็นระเบียบและอ่านง่าย โดยจะถูกแยกเก็บไว้ในตัวแปร `headers`, `widths` และ `aligns` ตามลำดับ เพื่อใช้ในการสร้างตารางรายงานในส่วนถัดไป จากนั้นโปรแกรมจะเริ่มสร้างเนื้อหาของรายงานในลิสต์ `lines` โดยเพิ่มบรรทัดหัวข้อย่อยรายงาน เช่น “Car Rental System - Summary Report” เพื่อแสดงชื่อรายงานหลัก ตามด้วยวันและเวลาที่รายงานถูกสร้าง (Generated At) ซึ่งได้จากคำสั่ง `datetime.now().strftime()` รวมถึงการระบุเวอร์ชันของโปรแกรม (App Version), ลักษณะการจัดเก็บข้อมูล (Endianness), และการเข้ารหัสข้อความ (Encoding) เพื่อแสดงข้อมูลทางเทคนิคของระบบสำหรับอ้างอิงทางการจัดการข้อมูล

ท้ายสุด โปรแกรมจะเพิ่มหัวข้อย่อย “Rental Report for [วันที่ปัจจุบัน]” เพื่อระบุช่วงเวลาที่ย่อยที่รายงานนี้ถูกสร้างขึ้น ซึ่งช่วยให้ผู้ดูแลระบบสามารถระบุได้ว่ารายงานฉบับนั้นเป็นของวันใด ฟังก์ชันในส่วนนี้จึงมีบทบาทสำคัญในการจัดเตรียมโครงสร้างและข้อมูลพื้นฐานของรายงานให้ครบถ้วนก่อนเข้าสู่ขั้นตอนการสรุปผลและแสดงรายละเอียดข้อมูลสัญญาเช่าในภายหลัง

```
# Columns per new requirement: add ID Card and Phone
cols = [
    ("Contract ID", 12, "<"),
    ("Name", 16, "<"),
    ("ID Card", 14, "<"),
    ("Phone", 14, "<"),
    ("Car Plate", 12, "<"),
    ("Car", 12, "<"),
    ("Start Date", 12, "<"),
    ("End Date", 12, "<"),
    ("Cars Rente", 12, ">"),
    ("Price/Day", 10, ">"),
    ("Total Price", 12, ">"),
    ("Status", 8, "<"),
]
headers = [c[0] for c in cols]
widths = [c[1] for c in cols]
aligns = [c[2] for c in cols]

lines = []
lines.append("Car Rental System - Summary Report")
lines.append(f"Generated At : {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
lines.append(f"App Version : {APP_VERSION}")
lines.append(f"Endianness : {ENDIANNESS}")
lines.append(f"Encoding : {ENCODING}")
lines.append("")
lines.append(f"Rental Report for {datetime.now().strftime('%d/%m/%Y')}")
lines.append("")
```

ภาพที่ 4-23 ฟังก์ชัน generate_report() ใช้กำหนดรูปแบบและโครงสร้างของรายงานสรุป

```
def border():
    return "+" + "+".join("-" * (w+2) for w in widths) + "+"

lines.append(border())
lines.append("|" + "|".join(f" {h:{a}{w}} " for h,w,a in zip(headers,widths,aligns)) + "|")
lines.append(border())

for con in contracts:
    cust = customers.get(con.cust_id)
    car = cars.get(con.car_id)
    name = cust.name if cust else "-"
    idc = cust.idcard if cust else "-"
    phone = cust.phone if cust else "-"
    plate = car.plate if car else "-"
    car_text = f"{car.brand} {car.model}" if car else con.car_id
    # status mapping
    raw = (con.status or "").strip()
    if not raw: raw = "Active" if con.active==1 else "Closed"
    s = raw.lower()
    if raw=="1" or s in ("active","open"): status_text="Active"
    elif raw=="0" or s in ("closed","close","inactive"): status_text="Closed"
    else: status_text = raw.title()[:12]
```

ภาพที่ 4-24 ฟังก์ชัน generate_report() กำหนดรูปแบบและโครงสร้างของรายงานสรุป

4.4.4 ในส่วนนี้ของฟังก์ชัน `generate_report()` โปรแกรมทำหน้าที่จัดรูปแบบการแสดงผลข้อมูลสัญญาเช่าให้อยู่ในรูปแบบตาราง โดยใช้ลูป `for` ร่วมกับข้อมูลความกว้างและการจัดตำแหน่งของคอลัมน์ เพื่อจัดเรียงข้อมูลให้สวยงามและอ่านง่าย หากข้อความยาวเกินที่กำหนด ระบบจะตัดข้อความให้พอดีกับตาราง จากนั้นบันทึกข้อมูลแต่ละบรรทัดลงในลิสต์ `lines` เพื่อใช้สร้างรายงานต่อไป หลังจากแสดงข้อมูลครบแล้ว โปรแกรมจะเพิ่มหัวข้อ “Monthly Income Summary” เพื่อสรุปรายได้รายเดือน โดยใช้ตัวแปร `msum` เก็บผลรวมของรายได้ (`total_cents`) แยกตามเดือนจากวันเริ่มต้นของสัญญาเช่า การทำงานส่วนนี้ช่วยให้รายงานมีทั้งข้อมูลเชิงรายละเอียดและข้อมูลสถิติทางการเงินที่สามารถใช้วิเคราะห์ผลประกอบการของระบบเช่ารถได้อย่างมีประสิทธิภาพ

```
cells=[]
for val,w,a in zip(row,widths,aligns):
    sval = str(val)
    if len(sval)>w: sval=sval[:w]
    cells.append(f" {sval:{a}{w}} ")
lines.append("|"+"".join(cells)+"|")

lines.append(border())
lines.append("")
lines.append("Monthly Income Summary")
lines.append("")
msum: Dict[str,int] = defaultdict(int)
for con in contracts:
    msum[con.start[:7]] += con.total_cents
```

ภาพที่ 4-25 ฟังก์ชัน `generate_report()` ใช้จัดรูปแบบการแสดงผล

4.4.5 ในส่วนนี้ของฟังก์ชัน `generate_report()` โปรแกรมทำหน้าที่สร้างรายงานสรุปผลการดำเนินงานในรูปแบบต่าง ๆ ได้แก่ รายได้รวมรายเดือน, รายงานตามลูกค้า และรายงานตามรุ่นรถ โดยเริ่มจากการกำหนดรายชื่อเดือนในตัวแปร `month_names` เพื่อใช้ในการแปลงค่าของเดือนจากตัวเลขเป็นชื่อเดือน เช่น “01” เป็น “January” จากนั้นโปรแกรมจะวนลูปอ่านค่าจากตัวแปร `msum` ซึ่งเก็บผลรวมรายได้ (`total_cents`) แยกตามเดือน เพื่อแสดงผลในรูปแบบ “เดือน-ปี” พร้อมจำนวนรายได้ทั้งหมดของเดือนนั้น ๆ โดยใช้ฟังก์ชัน `_fmt_money()` ในการจัดรูปแบบตัวเลขให้อยู่ในหน่วยเงินบาทอย่างเป็นระเบียบ

ถัดมาคือส่วน “Monthly Customer Summary” ซึ่งเป็นการสรุปข้อมูลลูกค้าที่มีการเช่ารถในแต่ละเดือน โปรแกรมจะใช้โครงสร้างข้อมูลแบบ `defaultdict(set)` เพื่อเก็บชื่อของลูกค้าที่เช่าในแต่ละเดือน

ละช่วงเดือน โดยเชื่อมโยงข้อมูลจากออบเจกต์ customers ผ่านรหัสลูกค้า (cust_id) หากไม่พบข้อมูลลูกค้า ระบบจะแสดงรหัสแทนชื่อเพื่อป้องกันความผิดพลาด จากนั้นจะจัดเรียงเดือนตามลำดับเวลา (sorted()) และพิมพ์รายชื่อกู้ในแต่ละเดือนออกมาในรายงาน

ส่วนสุดท้ายคือ “Monthly Car Rental Summary” ซึ่งมีหลักการทำงานคล้ายกันแต่เปลี่ยนจากการจัดกลุ่มลูกค้าเป็นการจัดกลุ่มตามรถยนต์ที่ถูกเช่าในแต่ละเดือน โดยใช้ข้อมูลจากออบเจกต์ cars และรหัสรถ (car_id) เพื่อดึงชื่อรุ่น (model) ของรถที่ถูกใช้งานในช่วงเวลานั้น ๆ แล้วเพิ่มลงในพจนานุกรม by_model เพื่อใช้สร้างรายงานสรุปในรูปแบบ “รุ่นรถ – เดือน – ปี” โค้ดในส่วนนี้ช่วยให้รายงานที่สร้างขึ้นมีมิติของข้อมูลที่หลากหลาย ทั้งในเชิงรายได้ ลูกค้า และรถยนต์ ซึ่งเป็นประโยชน์อย่างยิ่งต่อการวิเคราะห์ประสิทธิภาพการให้บริการและการวางแผนเชิงธุรกิจของระบบเช่ารถยนต์ในอนาคต

```
month_names = ["January", "February", "March", "April", "May", "June",
               "July", "August", "September", "October", "November", "December"]
for ym in sorted(msum.keys()):
    year, month = ym.split("-")
    (variable) formatted_date: str = f"{month_names[int(month)-1]} {year}"
    formatted_date = f"{month_names[int(month)-1]} {year}"
    lines.append(f"{formatted_date} total : {_fmt_money(msum[ym])}")
lines.append("")
lines.append("Monthly Customer Summary")
lines.append("-"*28)
by_cust: Dict[str,set] = defaultdict(set)
for con in contracts:
    name = customers.get(con.cust_id).name if customers.get(con.cust_id) else con.cust_id
    by_cust[con.start[:7]].add(name)
for ym in sorted(by_cust.keys()):
    year, month = ym.split("-")
    month_name = month_names[int(month)-1]
    formatted_date = f"{month_name} {year}"
    lines.append(formatted_date)
    for n in sorted(by_cust[ym]): lines.append(f"    {n}")
    lines.append("")
lines.append("Monthly Car Rental Summary")
lines.append("-"*28)
by_model: Dict[str,list] = defaultdict(list)
for con in contracts:
    label = (cars.get(con.car_id).model if cars.get(con.car_id) else con.car_id)
    by_model[con.start[:7]].append(label)
for ym in sorted(by_model.keys()):
    year, month = ym.split("-")
    month_name = month_names[int(month)-1]
    formatted_date = f"{month_name} {year}"
    lines.append(formatted_date)
    for m in sorted(set(by_model[ym])): lines.append(f"    {m}")
    lines.append("")
```

ภาพที่ 4-26 ฟังก์ชัน generate_report() ใช้สร้างรายงานสรุปผลการดำเนินงานในรูปแบบต่าง ๆ

4.4.6 ในส่วนสุดท้ายของฟังก์ชัน `generate_report()` โปรแกรมทำหน้าที่สรุปข้อมูลสำคัญของระบบและบันทึกผลออกเป็นรายงานสรุป โดยเริ่มจากการนับจำนวนข้อมูลที่ยังเปิดใช้งานในแต่ละประเภท ได้แก่ รถยนต์ ลูกค้า และสัญญาเช่า โดยใช้คำสั่ง `sum()` ร่วมกับฟังก์ชัน `_iter_records()` เพื่อวนลูปอ่านข้อมูลจากไฟล์หลักของแต่ละประเภท ได้แก่ `CAR_FILE`, `CUSTOMER_FILE` และ `CONTRACT_FILE` จากนั้นตรวจสอบสถานะของข้อมูลด้วยเงื่อนไข `.active == 1` เพื่อเลือกเฉพาะเรคคอร์ดที่ยังคงใช้งานอยู่

ผลลัพธ์ที่ได้จะถูกเก็บไว้ในตัวแปร `cars_active`, `cust_active` และ `con_active` ตามลำดับ แล้วนำไปเพิ่มในลิสต์ `lines` เพื่อใช้ในการสร้างรายงานในรูปแบบข้อความ เช่น “Total cars”, “Total customers” และ “Total contracts” ซึ่งแสดงจำนวนรวมของข้อมูลแต่ละประเภทในระบบปัจจุบัน

เมื่อรวบรวมข้อมูลครบถ้วน โปรแกรมจะทำการบันทึกข้อมูลทั้งหมดลงในไฟล์รายงานที่กำหนดไว้ในตัวแปร `REPORT_FILE` โดยเปิดไฟล์ในโหมดเขียน (“w”) และเข้ารหัสแบบ UTF-8 เพื่อรองรับภาษาไทย จากนั้นใช้คำสั่ง `fp.write("\n".join(lines))` เพื่อรวมบรรทัดทั้งหมดในลิสต์ `lines` แล้วเขียนลงในไฟล์รายงาน

สุดท้าย โปรแกรมจะแสดงข้อความ “Report generated → [REPORT_FILE]” บนหน้าจอ เพื่อแจ้งให้ผู้ใช้ทราบว่า การสร้างรายงานเสร็จสิ้นและไฟล์รายงานถูกสร้างเรียบร้อยแล้ว ส่วนนี้ถือเป็นขั้นตอนสุดท้ายของกระบวนการสร้างรายงานในระบบเช่ารถ ช่วยให้ผู้ใช้ดูแลสามารถตรวจสอบสถิติและข้อมูลสรุปของระบบได้อย่างครบถ้วนและเป็นทางการ

```
cars_active = sum(1 for _,b in _iter_records(CAR_FILE,CAR_SIZE) if Car.unpack(b).active==1)
cust_active = sum(1 for _,b in _iter_records(CUSTOMER_FILE,CUS_SIZE) if Customer.unpack(b).active==1)
con_active = len(contracts)
lines.append(f"Total cars      : {cars_active}")
lines.append(f"Total customers : {cust_active}")
lines.append(f"Total contracts : {con_active}")
lines.append("")

with open(REPORT_FILE,"w",encoding="utf-8") as fp:
    fp.write("\n".join(lines))

print(f"Report generated -> {REPORT_FILE}")
```

ภาพที่ 4-27 ฟังก์ชัน `generate_report()` ใช้ทำหน้าที่สรุปข้อมูลสำคัญของระบบและบันทึกผลออก

4.5 ฟังก์ชันเมนูหลักของระบบ (Main Program)

4.5.1 ฟังก์ชัน `main()` เป็นจุดเริ่มต้นหลักของการทำงานในโปรแกรมระบบเช่ารถ โดยมีหน้าที่ควบคุมลำดับขั้นตอนการทำงานทั้งหมดของระบบ ตั้งแต่การตรวจสอบไฟล์ข้อมูล การตั้งค่าเริ่มต้น การรับคำสั่งจากผู้ใช้ ไปจนถึงการเรียกใช้เมนูย่อยต่าง ๆ ภายในโปรแกรม การทำงานเริ่มจากการเรียกใช้ฟังก์ชัน `_ensure_files()` เพื่อสร้างไฟล์ข้อมูลที่จำเป็นหากยังไม่มีอยู่ และฟังก์ชัน `migrate_contracts_if_old()` เพื่อปรับปรุงรูปแบบไฟล์สัญญาเข้าให้เป็นเวอร์ชันล่าสุด

ต่อมาโปรแกรมจะกำหนดการรับอาร์กิวเมนต์ผ่านโมดูล `argparse` เพื่อให้สามารถรันโปรแกรมด้วยคำสั่งเพิ่มเติม เช่น `--report` สำหรับสร้างรายงานทันที และ `--seed-sample` สำหรับเพิ่มข้อมูลตัวอย่างเข้าสู่ระบบ ในกรณีที่มีการระบุอาร์กิวเมนต์ `--report` โปรแกรมจะเรียกใช้ฟังก์ชัน `generate_report()` เพื่อสร้างรายงานสรุปผลโดยอัตโนมัติ แล้วสิ้นสุดการทำงานทันที

หากไม่มีการส่งอาร์กิวเมนต์เข้ามา โปรแกรมจะเข้าสู่ลูปหลัก (`while True:`) ซึ่งใช้ในการแสดงเมนูหลัก (MENU) เพื่อให้ผู้ใช้เลือกการทำงาน โดยมีตัวเลือกหลักได้แก่ การจัดการข้อมูลรถ (`manage_cars_menu()`), การจัดการลูกค้า (`manage_customers_menu()`), การจัดการสัญญาเช่า (`manage_contracts_menu()`), และการสร้างรายงาน (`generate_report()`) นอกจากนี้ยังมีตัวเลือก “0” สำหรับออกจากโปรแกรมอย่างปลอดภัย

ในกรณีที่ผู้ใช้เลือกสร้างรายงาน ระบบจะมีการดักจับข้อผิดพลาด (`try-except`) เพื่อป้องกันไม่ให้โปรแกรมหยุดทำงานอย่างกะทันหัน พร้อมแสดงข้อความแจ้งเตือนในกรณีเกิดข้อผิดพลาด เมื่อผู้ใช้เลือกออกจากโปรแกรม โปรแกรมจะพิมพ์ข้อความ “Goodbye” แสดงการสิ้นสุดการทำงาน ส่วนคำสั่ง `if __name__ == "__main__":` จะตรวจสอบให้แน่ใจว่าฟังก์ชัน `main()` จะถูกเรียกใช้ก็ต่อเมื่อไฟล์นี้ถูกรันโดยตรง ซึ่งช่วยให้ระบบทำงานได้อย่างถูกต้องและเป็นอิสระจากโมดูลอื่น ฟังก์ชันนี้จึงถือเป็นหัวใจหลักของโปรแกรมที่ควบคุมทุกส่วนให้ทำงานร่วมกันอย่างมีประสิทธิภาพ

```

def main(argv=None):
    _ensure_files()
    migrate_contracts_if_old()
    parser = argparse.ArgumentParser(description="Car Rental System")
    parser.add_argument("--report", action="store_true")
    parser.add_argument("--seed-sample", action="store_true")
    args, _ = parser.parse_known_args(argv)
    if args.seed_sample:
        seed_sample()
    if args.report:
        generate_report()
        return
    while True:
        ch = input(MENU).strip()
        if ch=="1": manage_cars_menu()
        elif ch=="2": manage_customers_menu()
        elif ch=="3": manage_contracts_menu()
        elif ch=="4": generate_report()
        elif ch=="0":
            try: generate_report()
            except Exception as e: print("ผิดพลาด (สร้างรายงานไม่สำเร็จ แต่ออกจากโปรแกรมได้):", e)
            print("Goodbye"); break
        else:
            print("โปรดเลือก 0-4")

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print("\nผู้ใช้ยกเลิกโปรแกรม")

```

ภาพที่ 4-28 ฟังก์ชัน main()

บทที่ 5

สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

จากการดำเนินโครงการเรื่อง ระบบจัดการข้อมูลการเช่ารถ ผู้จัดทำสามารถพัฒนาโปรแกรมต้นแบบที่ช่วยในการจัดการข้อมูลรถ ลูกค้า และสัญญาเช่าได้สำเร็จ โดยใช้ภาษา Python ในการพัฒนา ระบบที่ได้สามารถเพิ่ม แก้ไข ลบ และค้นหาข้อมูลได้ รวมถึงสามารถสร้างและปิดสัญญาเช่าได้อย่างถูกต้องและสะดวก อีกทั้งยังสามารถสรุปรายงานการเช่าออกมาได้ ทำให้เห็นว่าโครงการนี้สามารถตอบโจทย์ความต้องการพื้นฐานของธุรกิจเช่ารถได้ในระดับหนึ่ง

5.2 อภิปรายผล

จากการทดลองใช้งานพบว่าระบบมีความสะดวกและช่วยลดความผิดพลาดจากการจัดเก็บข้อมูลแบบเอกสาร แต่ก็ยังมีข้อจำกัดบางประการ เช่น การจัดเก็บข้อมูลยังอยู่ในรูปแบบไฟล์ข้อความ ไม่ได้ใช้ฐานข้อมูลที่เหมาะสมสำหรับข้อมูลจำนวนมาก และรูปแบบรายงานยังคงเป็นเพียงข้อความพื้นฐาน อย่างไรก็ตาม ระบบนี้ถือว่าการฝึกทักษะด้านการออกแบบและเขียนโปรแกรมของผู้จัดทำได้เป็นอย่างดี

5.3 ข้อเสนอแนะ

สำหรับแนวทางในอนาคต ควรมีการพัฒนาระบบให้รองรับการใช้งานบนเครือข่ายหรือรูปแบบเว็บแอปพลิเคชัน รวมถึงการปรับปรุงการเก็บข้อมูลให้อยู่ในฐานข้อมูลจริงเพื่อความเร็วและความถูกต้องที่มากขึ้น นอกจากนี้ยังสามารถเพิ่มเติมฟังก์ชันการทำงาน เช่น

5.3.2 ควรปรับปรุงการจัดเก็บข้อมูลให้เป็นฐานข้อมูลเชิงสัมพันธ์ (เช่น MySQL หรือSQLite) เพื่อรองรับข้อมูลจำนวนมากและค้นหาได้รวดเร็ว

5.3.3 เพิ่มระบบเข้าสู่ระบบ (Login) และกำหนดสิทธิ์การใช้งานตามบทบาทผู้ใช้ เช่น พนักงาน ผู้จัดการ เจ้าหน้าที่บัญชี

5.3.4 พัฒนาระบบให้สามารถทำงานบนเว็บหรือเครือข่ายได้ เพื่อรองรับผู้ใช้หลายคนพร้อมกัน

5.3.5 ขยายความสามารถของรายงาน เช่น รายงานเป็นกราฟ แผนภูมิ หรือระบบ Dashboard เพื่อช่วยผู้จัดการวิเคราะห์ข้อมูลได้ง่ายขึ้น

5.3.6 พัฒนาระบบการชำระเงินออนไลน์ หรือการเชื่อมต่อกับระบบติดตามรถ (GPS) เพื่อให้ใกล้เคียงกับระบบเช่ารถจริงในเชิงพาณิชย์

5.4 สิ่งที่ได้จัดทำได้รับในการพัฒนาโครงการ

ในการพัฒนาโครงการเรื่อง ระบบจัดการข้อมูลการเช่ารถ ผู้จัดทำได้รับความรู้และประสบการณ์หลายด้าน ทั้งในเชิงวิชาการและการปฏิบัติจริง ด้านวิชาการได้เรียนรู้การนำความรู้จากรายวิชาต่าง ๆ มาประยุกต์ใช้ เช่น การเขียนโปรแกรมด้วยภาษา Python การออกแบบโครงสร้างข้อมูล และการพัฒนาระบบให้สามารถทำงานได้ตามความต้องการจริง นอกจากนี้ยังได้ฝึกการวิเคราะห์ปัญหา การออกแบบระบบ การวางแผนการทำงาน รวมถึงการแก้ไขข้อผิดพลาดที่เกิดขึ้นระหว่างการพัฒนา

ด้านการทำงานร่วมกัน ผู้จัดทำได้เรียนรู้การทำงานเป็นทีม การแบ่งหน้าที่รับผิดชอบ การสื่อสาร และการแลกเปลี่ยนความคิดเห็นกับเพื่อนร่วมกลุ่ม ทำให้สามารถพัฒนาโครงการจนสำเร็จได้ตามที่วางไว้ อีกทั้งยังได้ฝึกความรับผิดชอบ ความอดทน และการแก้ปัญหาเฉพาะหน้า ซึ่งเป็นทักษะที่สำคัญสำหรับการทำงานจริงในอนาคต

