

ระบบจัดการข้อมูลหนัง
Movie information management system

นายชัชฉันทน์ วงษ์แตร์ รหัส 6706022610331
นายธนวิษญ์ โอสธ รหัส 6706022610161
นางสาวเบญญภา เตชาธรรมนนท์ 6706022610195

โครงการนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอม
เกล้าพระนครเหนือ
ปีการศึกษา 2567
ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

คำนำ

การจัดทำโครงการ “ระบบจัดการข้อมูลหนังสือ” นี้เป็นส่วนหนึ่งของวิชา COMPUTER PROGRAMMING ของหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศคณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ เพื่อให้นักศึกษาได้นำความรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถทำงานได้จริง โดยเน้นการออกแบบและเขียนโปรแกรมในภาษา Python ซึ่งเป็นภาษาที่เรียนมาในวิชา COMPUTER PROGRAMMING โดยโครงการนี้จะช่วย การคิดวิเคราะห์ และแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรมสารสนเทศและเครือข่ายในอนาคต หากมีข้อผิดพลาดประการใด คณะผู้จัดทำต้องขออภัยไว้ ณ ที่นี้ด้วย

สารบัญ

หน้า

คำนำ.....	ข
สารบัญ	ค
สารบัญภาพ	ง
บทที่ 1 บทนำ.....	1
บทที่ 2 ระบบจัดการข้อมูลหนังสือ	2

สารบัญภาพ

	หน้าที่
ภาพที่ 2-1	2
ภาพที่ 2-2	2
ภาพที่ 2-3	3
ภาพที่ 2-4	4
ภาพที่ 2-5	4
ภาพที่ 2-6	5
ภาพที่ 2-7	6
ภาพที่ 2-8	6
ภาพที่ 2-9	6
ภาพที่ 2-10	6
ภาพที่ 2-11	7
ภาพที่ 2-12	7
ภาพที่ 2-13	7
ภาพที่ 2-14	8
ภาพที่ 2-15	8
ภาพที่ 2-16	8
ภาพที่ 2-17	9
ภาพที่ 2-18	9
ภาพที่ 2-19	9
ภาพที่ 2-20	10
ภาพที่ 2-21	10
ภาพที่ 2-22	11
ภาพที่ 2-23	11
ภาพที่ 2-24	Error! Bookmark not defined.
ภาพที่ 2-25	13
ภาพที่ 2-26	13
ภาพที่ 2-27	13
ภาพที่ 2-28	14

ภาพที่ 2-29	14
ภาพที่ 2-30	14
ภาพที่ 2-31	14
ภาพที่ 2-32	14
ภาพที่ 2-33	15
ภาพที่ 2-34	16
ภาพที่ 2-35	17

บทที่ 1

บทนำ

1.1 วัตถุประสงค์ของโครงการ

- 1.1.1 เพื่อพัฒนาระบบที่สามารถจัดการหนังสือได้อย่างมีประสิทธิภาพ
- 1.1.2 เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วย Python
- 1.1.3 เพื่อเรียนรู้การจัดการข้อมูลและไฟล์
- 1.1.4 เพื่อเรียนรู้การทำงานร่วมกันเป็นทีม

1.2 ขอบเขตของโครงการ

- 1.2.1 ระบบจัดการข้อมูลหนังสือจะมีฟังก์ชันพื้นฐาน 9 ฟังก์ชัน 1. สร้างไฟล์บันทึกใหม่ 2. เพิ่มบันทึกใหม่ 3. แสดงบันทึก 4. แก้ไขบันทึก 5. ค้นหาบันทึก 6. รายงานสรุป 7. ลบบันทึก 8. ลบไฟล์บันทึก 9. ออก

- 1.2.2 ระบบจัดการข้อมูลหนังสือ ประกอบด้วย 5 필ด์ ได้แก่ ID, Name, Year, Type, Price

- 1.2.3 ระบบจัดการข้อมูลหนังสือจะมีเมนูเพื่อให้ผู้ใช้สามารถเลือกดำเนินการได้

1.3 ประโยชน์ที่ได้รับ

- 1.3.1 พัฒนาระบบที่สามารถจัดการหนังสือได้อย่างมีประสิทธิภาพ
- 1.3.2 พัฒนาทักษะการเขียนโปรแกรม
- 1.3.3 เรียนรู้การจัดการข้อมูลและไฟล์
- 1.3.4 เรียนรู้การทำงานร่วมกันเป็นทีม

1.4 เครื่องมือที่คาดว่าจะต้องใช้

- 1.4.1 ภาษา Python
- 1.4.2 Visual Studio Code

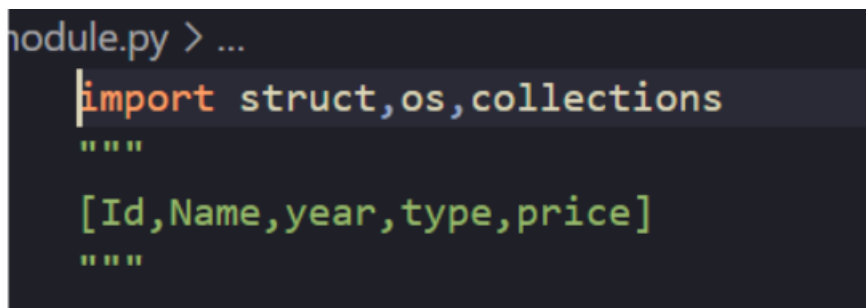
บทที่ 2

ระบบจัดการข้อมูลหนัง

2.1 Library

เริ่มต้นทำงานด้วยการ import library โดยประกอบไปด้วย

- struct เพื่อใช้ในการทำงานกับไฟล์ binary
- os ใช้การการทำงานกับไฟล์ต่างๆได้อย่างสะดวก
- collection เพื่อความง่ายในการทำงานกับ dictionary



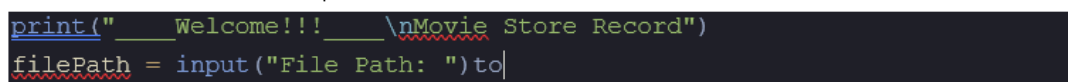
```
module.py > ...
import struct,os,collections
"""
[Id,Name,year,type,price]
"""
```

ภาพที่ 2-1

2.2 main.py

ไฟล์หลักที่ใช้รันโปรแกรม โดยประกอบไปด้วย

ส่วนยินดีต้อนรับ และถาม filepath



```
print("____Welcome!!!____\nMovie Store Record")
filePath = input("File Path: ")to|
```

ภาพที่ 2-2

จากนั้นก็จะเป็น loop ให้ user ทำการเลือก action

```

while runing:
    choice = input("Type your action (1 - 9) (0 for action list): ")
    while True:
        if choice not in ["0","1","2","3","4","5","6","7","8","9"]:
            print(Red + "Error: ValueError { Please Enter 1 to 9 }!!" + Reset)
            break
        else:
            match choice:
                case "0":
                    print("_____")
                    print(action)
                    print("_____")
                    print()
                    break
                case "1":
                    md.save_records(filePath)
                    break
                case "2":
                    md.add_records(filePath)
                    break
                case "3":
                    md.read_records(filePath)
                    break
                case "4":
                    md.edit_record(filePath)
                    break
                case "5":
                    md.find_records(filePath)
                    break
                case "6":
                    md.summary_Report(filePath)

```

ภาพที่ 2-3

*ตัวอย่างหน้าแสดงผล

```

____Welcome!!!____
Movie Store Record
_____
1.Create new File record
2.Add new records
3.Show All records
4.Edit record
5.Find records
6.Summary Report
7.Remove record
8.Delete file record
9.Quit
_____

Type your action (1 - 9) (0 for action list): 

```

ภาพที่ 2-4

2.3 save_records()

เป็น fuction ที่สร้างไฟล์recordใหม่ หรือสร้างทับไฟล์เดิมโดยการเริ่มต้นด้วยการเปิดไฟล์ด้วย เป็นฟังก์ชันสำหรับสร้างไฟล์ record ใหม่ โดยเริ่มทำงานด้วยเปิดไฟล์ขึ้นมาในโหมด "wb" จากนั้น ใช้ try เพื่อถาม user ว่าต้องการกี่ record ถ้า error ให้หยุดการทำงานและแจ้งเตือน ถ้าไม่ให้ทำงานต่อ

```

def save_records(filePath):

    with open(filePath,"wb") as file:
        r_count = 0
        try:
            count = int(input("How many record you want to create?: "))
        except ValueError as e:
            print(f"Error: {e}")
        except Exception as e:
            print(f"Error: {e}")
        else:

```

ภาพที่ 2-5

หลังจากได้จำนวน record ที่ user ต้องการ ให้ทำการ loop ให้ใส่ข้อมูล id, ชื่อ, ect.. และทำการเขียนข้อมูลนั้นไปยังไฟล์ในรูปแบบ binary ด้วยโครงสร้างที่กำหนด จนกว่าจะครบตามจำนวนที่ user ระบุ เมื่อเสร็จสิ้นให้แจ้งเตือน

```

else:
    for i in range(count):
        try:
            print(f"Recod Number #{r_count +1}")
            id = int(input("ID: "))
            name = input("Name: ")
            year = int(input("Year: "))
            cdType = input("Type: ")
            price = float(input("Price: "))

            except ValueError as e:
                print(f"Error: {e}")
                break
            except Exception as e:
                print(f"Error: {e}")
                break

            else:
                data = struct.pack("i20si20sf",id,name.encode(),year,cdType.encode(),price)
                file.write(data)
                r_count += 1
                print()

    print(Green + f"Done!! {r_count} record has been created" + Reset)

```

ภาพที่ 2-6

2.4 add_records()

เริ่มต้นด้วยการทำการเช็คค่าไฟล์ที่ user ถามหานั้นมีอยู่จริง

ถ้ามีอยู่จริงอยู่จริงให้ทำงานต่อ

การทำงานโดยส่วนใหญ่ของฟังก์ชันนี้นั้นเหมือนกับsave_records() ต่างกันที่ใช้ ("ab") โหมดแทน try เพื่อถาม user ว่าต้องการกี่ record ถ้า error ให้หยุดการทำงานและแจ้งเตือน ถ้าไม่ให้ทำงานต่อ

```

check = os.path.exists(filePath)
if check != True:
    print("Error: File Not Found!!")

else:
    with open(filePath,"ab") as file:
        r_count = 0
        try:
            count = int(input("How many record you want to create?: "))
        except ValueError as e:
            print(f"Error: {e}")
        except Exception as e:
            print(f"Error: {e}")
        else:

```

ภาพที่ 2-7

หลังจากได้จำนวน record ที่ user ต้องการ ให้ทำการ loop ให้ใส่ข้อมูล id, ชื่อ, ect.. และทำการเขียนข้อมูลนั้นไปยังไฟล์ในรูปแบบ binary ด้วยโครงสร้างที่กำหนด จนกว่าจะครบตามจำนวนที่ user ระบุ เมื่อเสร็จสิ้นให้แจ้งเตือน

```

else:
    for i in range(count):
        try:
            print(f"Recod Number #{r_count + 1}")
            id = int(input("ID: "))
            name = input("Name: ")
            year = int(input("Year: "))
            cdType = input("Type: ")
            price = float(input("Price: "))

            except ValueError as e:
                print(f"Error: {e}")
                break
            except Exception as e:
                print(f"Error: {e}")
                break

            else:
                data = struct.pack("i20si20sf", id, name.encode(), year, cdType.encode(), price)
                file.write(data)
                r_count += 1
                print()

    print(Green + f"Done!! {r_count} record has been created" + Reset)

```

ภาพที่ 2-8

2.5 class ProductTable

เป็น class สำหรับไว้สร้าง ตารางสำหรับแสดงค่าผลลัพธ์ใน

รูปแบบที่ดูง่ายและสวยงาม

class นั้นต้องการ 3 argument ข้อมูลที่จะแสดง, หัวข้อ, และขนาดของแต่ละช่อง

```

# Table
class ProductTable:
    def __init__(self, data, header, col_widths):
        self.data = data
        self.header = header
        self.col_widths = col_widths

```

ภาพที่ 2-9

ตามด้วย fuction ในการช่วยจัดรูปแบบของ table

```

def format_row(self, row):
    return ''.join(f'{str(cell):<{width}}' for cell, width in zip(row, self.col_widths))

def print_separator(self):
    print('-' * sum(self.col_widths))

```

ภาพที่ 2-10

และทำ fuction สำหรับการ display โดยเริ่มจากการแสดง header และทำการขีดเส้นเว้นด้วย separator และ loop แสดง ข้อมูลที่มีอยู่ออกมา

```
def display(self):
    print(self.format_row(self.header))
    self.print_separator()
    for row in self.data:
        print(self.format_row(row))
```

ภาพที่ 2-11

2.6 clean_data()

การดึงข้อมูล binary ออกมาบางครั้งจะมี null (\0xx) ออกมาด้วย fuction นี้จึงมีไว้เพื่อลบ null ออกเพื่อความถูกต้องของข้อมูลที่จะนำไปแสดงผล โดยใช้ loop ในการดูข้อมูลและตาม หา null ถ้าพบ null ให้แทนที่ด้วย “ ”

```
def clean_data(raw_data):
    return [
        [str(cell).replace('\x00', ' ') for cell in row]
        for row in raw_data
    ]
```

ภาพที่ 2-12

2.7 read_records()

ใช้ในการแสดงผลข้อมูลทั้งหมดภายใน ไฟล์ binary ออกมาเป็น ตารางที่อ่านและเข้าใจได้ง่าย มีหลังการทำงานด้วยการเปิด ไฟล์ binary ขึ้นมาและแปลงข้อมูลตามโครงสร้าง โดยทำการใช้ while loop ในการดึงข้อมูลออกมาและนำไปเก็บไว้ในตัวแปร result

```
result = []

while True:
    record = file.read(struct.calcsize("i20si20sf"))
    if not record:
        break
    nn = []
    record = struct.unpack("i20si20sf", record)
    record = (
        record[0],           # ID (int)
        record[1].decode().strip(), # Name (str)
        record[2],           # Year (int)
        record[3].decode().strip(), # Type (str)
        record[4],           # Price (float)
    )
    for i in record:
        nn.append(i)
    result.append(nn)
```

ภาพที่ 2-13

จากนั้นนำข้อมูลทั้งหมดที่เก็บไว้ใน result มาทำการ clean เพื่อลบ null และนำไปแสดงผล ในรูปแบบตารางด้วย class ProductTable() พร้อมกำหนดหัวข้อ (ID,NAME<ect...) และ ขนาดของตาราง และสั่งให้แสดงผลด้วย Fuction display จาก class ProductTable()

```
result = clean_data(result)
print("_____")

header = ['ID', 'Name', 'Year', 'Type', 'Price($)']
col_widths = [10, 20, 10, 10, 10]
table = ProductTable(result, header, col_widths)
table.display()
print("_____")
```

ภาพที่ 2-14

*ตัวอย่างหน้าแสดงผล

Result:

ID	Name	Year	Type	Price(\$)
1	Jaw	1995	Cd	22.0
2	Jonhwig	2020	Dvd	33.0
3	Theshining	1995	Cd	20.0
4	1917	2021	Dvd	40.0
5	Shingodzilla	2016	Blueray	44.0

Type your action (1 - 9) (0 for action list):

ภาพที่ 2-15

2.8 find_record()

function ไว้สำหรับหา record ที่ต้องการด้วย ชื่อหรือid เริ่มต้นด้วยการให้ user ใช้ชื่อหรือไอดี ที่กำลังตามหามาและเก็บไว้ในตัวแปร find

```
se:
    find = input('Which record are you looking for? (Id,Name): ')
```

ภาพที่ 2-16

สร้างตัวแปร สำหรับเก็บข้อมูล n_record ไว้เก็บข้อมูลทั้งหมด Result ไว้รับข้อมูลที่ user กำลังตามหาอยู่ count ไว้เก็บค่าเวลาที่เจอไฟล์ที่ user ตามหาอยู่ ทำงานด้วยการเปิด ไฟล์ binary ขึ้นมาและแปลงข้อมูลตามโครงสร้าง ด้วยกการ ใช้ while loop และค่อยๆเก็บไว้ใน ตัวแปร n_record

```

n_record = []
result = []
oo = 0
with open(filePath,"rb") as file:
    while True:
        record = file.read(struct.calcsize("i20si20sf"))
        if not record:
            break
        else:
            record = struct.unpack("i20si20sf",record)
            record = record[0],record[1].decode(),record[2],record[3].decode(),record[4]
            n_record.append(list(record))

```

ภาพที่ 2-17

จากนั้นทำการเช็คว่ามีชื่อ หรือ id ที่ตรงกับ find ใหม่ใน n_record ด้วยการ ใช้ for loop ใน n_record ถ้าตรงตามเงื่อนไข ให้ append ไปยัง ตัวแปร result และ count +1

```

for i in n_record:
    if find == str(i[0]) or find in i[1]:
        result.append(i)
        count += 1

```

ภาพที่ 2-18

จากนั้นทำการเช็คว่ามี record ถูกเพิ่มไปใน result ใหม่ ด้วยการนับ count ถ้า count ไม่เท่ากับ 0

ให้นำข้อมูลทั้งหมดที่เก็บไว้ใน result มาทำการ clean เพื่อลบ null

และนำไปแสดงผล ในรูปแบบตารางด้วย class ProductTable()

พร้อมกำหนดหัวข้อ (ID,NAME,ect...) และ ขนาดของตาราง และสั่งให้แสดงผลด้วย

Fuction display จาก class ProductTable()

ถ้าเท่ากับ 0 ให้แจ้งเตือน user ว่าไม่พบข้อมูล

```

if count == 0:
    print(Red + f"Not found [{find}] in any records" + Reset)
else:
    result = clean_data(result)
    header = ['ID', 'Name', 'Year', 'Type', 'Price($)']
    col_widths = [10, 20, 10, 10, 10]
    table = ProductTable(result, header, col_widths)
    table.display()

```

ภาพที่ 2-19

*ตัวอย่างหน้าแสดงผล

Type your action (1 - 9) (0 for action list): 3
Which record are you looking for? (Id,Name): 2
Result:

ID	Name	Year	Type	Price(\$)
2	Jonhwig	2020	Dvd	33.0

ภาพที่ 2-20

2.9 edit_record()

เป็น fuction สำหรับการเปลี่ยนแปลงข้อมูลบางอย่างใน record

บางอันตามที่ user เลือก

โดย เริ่มต้นคล้ายๆกับ read_rescord() ด้วยการนำดึงข้อมูล binary ออกมาเก็บไว้ ในรูปแบบ dictionary

มีหลังการทำงานด้วยการเปิด ไฟล์ binary ขึ้นมาและแปลงข้อมูลตามโครงสร้าง โดยทำการใช้ while loop ในการดึงข้อมูลออกมาและนำไปเก็บไว้ในตัวแปร record

และทำการแสดงของมูลเหล่านั้นพร้อมกับเลข index ข้อมูลเพื่อให้ user เลือก record ที่จะแก้ไข และทำการจัดการกับ error ด้วย try except

```
records = []
record_size = struct.calcsize("i20si20sf")

with open(file, "rb") as file_obj:
    while True:
        record = file_obj.read(record_size)
        if not record:
            break
        records.append(struct.unpack("i20si20sf", record))

print(Green + "Current Records:" + Reset)
print("_____")
for idx, record in enumerate(records):
    name = record[1].decode().strip()
    id = record[0]
    year = record[2]
    _type = record[3].decode().strip()
    price = record[4]
    print(f"{idx + 1}: [Id: {id}, Name: {name}, Year: {year}, Type: {_type}, Price: {price:.2f}$]")
print("_____")
try:
    index = int(input("Enter the record number you want to edit: ")) - 1
    if index < 0 or index >= len(records):
        print("Error: Invalid record number.")
        return
except ValueError:
    print("Error: Invalid input. Please enter a number.")
    return
```

จากนั้นนำข้อมูลใน dictionary ไปเก็บไว้ในตัวแปรใหม่ เพื่อนำไปใช้งาน

หลังจาก user ทำการเลือก record user จะสามารถเปลี่ยนแปลงข้อมูลของ record นั้นได้ไม่ว่าจะเป็น id,name,year,type,price ถ้าไม่ต้องการเปลี่ยนส่วนไหนให้ปล่อยว่างไว้

โดยใช้ if else ในการเช็คว่ามีข้อมูลช่องไหนถูกเปลี่ยนบ้าง
ภาพที่ 2-21

```
record_to_edit = records[index]
id = record_to_edit[0]
name = record_to_edit[1].decode().strip()
year = record_to_edit[2]
_type = record_to_edit[3].decode().strip()
price = record_to_edit[4]

print(f"\nEditing record {index + 1}: [Id: {id}, Name: {name}, Year: {year}, Type: {_type}, Price: {price:.2f}$]")

new_id = input(f"New ID (leave blank to not change): ")
new_name = input(f"New Name (leave blank to not change): ")
new_year = input(f"New Year (leave blank to not change): ")
new_type = input(f"New Type (leave blank to not change): ")
new_price = input(f"New Price (leave blank to not change): ")

new_id = int(new_id) if new_id else id
new_name = new_name.encode() if new_name.strip() else record_to_edit[1]
new_year = int(new_year) if new_year else year
new_type = new_type.encode() if new_type.strip() else record_to_edit[3]
new_price = float(new_price) if new_price else price
```

ภาพที่ 2-22

แล้วก็ทำการเขียนข้อมูลใหม่ไปยัง dictionary ที่เก็บข้อมูลไว้และทำการนำข้อมูลใน dictionary ที่ถูกแก้ไขเขียนทับไปยังไฟล์ binary เดิม เป็นการสิ้นสุดการเปลี่ยนแปลงข้อมูล ภายใน record

```
records[index] = (new_id, new_name, new_year, new_type, new_price)

with open(file, "wb") as file_obj:
    for record in records:
        file_obj.write(struct.pack("i20si20sf", *record))

print(Green + "Record updated successfully!" + Reset)
```

ภาพที่ 2-23

2.10 remove_record()

เป็น function สำหรับการ ลบ record บาง record ภายในไฟล์

การทำงานส่วนใหญ่ของ function นี้คล้ายคลึงกับ edit_record ด้วยการนำดึงข้อมูล binary ออกมาเก็บไว้

ในรูปแบบ dictionary และทำการแสดงของมูลเหล่านั้นพร้อมกับเลข index ข้อมูล

มีหลังการทำงานด้วยการเปิด ไฟล์ binary ขึ้นมาและแปลงข้อมูลตามโครงสร้าง โดยการใช้ while loop ในการดึงข้อมูลออกมาและนำไปเก็บไว้ในตัวแปร record

และทำการแสดงของมูลเหล่านั้นพร้อมกับเลข index ข้อมูลเพื่อให้ user เลือก record ที่จะแก้ไข

และทำการจัดการกับ error ด้วย try except

```
records = []
record_size = struct.calcsize("i20si20sf")

with open(file, "rb") as file_obj:
    while True:
        record = file_obj.read(record_size)
        if not record:
            break
        records.append(struct.unpack("i20si20sf", record))

print(Green + "Current Records:" + Reset)
print("_____")
for idx, record in enumerate(records):
    name = record[1].decode().strip()
    id = record[0]
    year = record[2]
    _type = record[3].decode().strip()
    price = record[4]
    print(f"{idx + 1}: [Id: {id}, Name: {name}, Year: {year}, Type: {_type}, Price: {price:.2f}$]")
print("_____")
try:
    index = int(input("Enter the record number you want to edit: ")) - 1
    if index < 0 or index >= len(records):
        print("Error: Invalid record number.")
        return
except ValueError:
    print("Error: Invalid input. Please enter a number.")
    return
```

จากนั้นถามการยืนยันจาก user ถ้า yes ให้ทำการลบ record นั้นออกจาก dictionary และเขียนข้อมูลใหม่ทับไฟล์เดิมถ้า no หรือ error ให้แจ้งเตือน

```
confirm = input(f"Are you sure you want to remove the record? (yes/no): ").strip().lower()

if confirm == 'yes':
    del records[index]
    with open(file, "wb") as file_obj:
        for record in records:
            file_obj.write(struct.pack("i20si20sf", *record))
    print(Green + "Record removed successfully!" + Reset)
    print()

elif confirm == 'no':
    print(Red + "Record removal canceled." + Reset)
    print()

else:
    print("Invalid input. Please respond with 'yes' or 'no'.")
    print()
```

ภาพที่ 2-24

2.11 report_summary()

เป็น fuction สำหรับแสดงข้อมูล โดยสรุป ประกอยไปด้วย

จำนวน product/record ทั้งหมด ราคาทั้งหมดของทุก product ประเภทของ product ทั้งหมด ปีของproduct ทั้งหมด หา ค่าเฉลี่ยของราคา product

โดยการ ดึงข้อมูล binary ออกมาเก็บไว้และทำการ clean ข้อมูลเพื่อ ลบ null

```
else:
    with open(filePath, "rb") as file:
        print(Green + "Result:" + Reset)
        result = []

        while True:
            record = file.read(struct.calcsize("i20si20sf"))
            if not record:
                break
            nn = []
            record = struct.unpack("i20si20sf", record)
            record = (
                record[0],                # ID (int)
                record[1].decode().strip(), # Name (str)
                record[2],                # Year (int)
                record[3].decode().strip(), # Type (str)
                record[4]                 # Price (float)
            )
            for i in record:
                nn.append(i)

            result.append(nn)

        result = clean_data(result)
```

ภาพที่ 2-25

จากนั้น สร้างตัวแปรสำหรับรับข้อมูลที่จะนำไปแสดงผล

```
totalCount = len(result)
totalRevenue = 0
typeCount = []
yearCount = []
averagePrice = 0
```

ภาพที่ 2-26

- ทำการหา จำนวน product/record ทั้งหมด

```
totalCount = len(result)
```

ภาพที่ 2-27

- หา ราคาทั้งหมดของทุก product และนำไปเก็บไว้ใน ตัวแปรที่เตรียมไว้

```
for i in result:
    totalRevenue += float(i[4])
```

ภาพที่ 2-28

- ประเภทของ product ทั้งหมดและนำไปเก็บไว้ใน ตัวแปรที่เตรียมไว้

```
for i in result:
    typeCount.append(i[3])
```

ภาพที่ 2-29

- หา ปีทั้งหมด ของ product ทั้งหมดและนำไปเก็บไว้ใน ตัวแปรที่เตรียมไว้

```
for i in result:
    yearCount.append(i[2])
```

ภาพที่ 2-30

- หา ค่าเฉลี่ยของราคา product และนำไปเก็บไว้ใน ตัวแปรที่เตรียมไว้

```
averagePrice += totalRevenue / totalCount
```

ภาพที่ 2-31

จากนั้นนำข้อมูลทั้งหมดออกมาแสดงผลในรูปแบบที่ดูง่าย

```
print("-" * 50)
print(f"{'Summary Report'}")
print("-" * 50)
print(f"{'Total Products: '} {totalCount}")
print(f"{'Total Revenue: '} ${totalRevenue}")
print("\nProducts Count by Type:")
print("-" * 50)
for p_type, count in (dict(collections.Counter(typeCount))).items():
    print(f"{p_type}: {count}")

print("\nYearly Breakdown:")
print("-" * 50)
for year, count in (dict(collections.Counter(yearCount))).items():
    print(f"{year}: {count}")

print(f"\n{'Average Price: '} {averagePrice:.2f}$")
print("-" * 50)
```

ภาพที่ 2-32

*ตัวอย่างหน้าแสดงผล

```
Type your action (1 - 5) (0 for action list): 0
Result:

Summary Report

Total Products: 5
Total Revenue: $159.0

Products Count by Type:

Cd: 2
Dvd: 2
Blueray: 1

Yearly Breakdown:

1995: 2
2020: 1
2021: 1
2016: 1

Average Price: 31.80$
```

ภาพที่ 2-33

2.12 del_file_reocrd

fuction สำหรับการ ลบไฟล์ record ที่

ทำงานด้วยการใช้ os.path.exists() ในการหาว่าไฟล์นั้นมีอยู่ไหม ถ้ามี return true

if ไม่ true ให้แจ้งว่าไม่พบไฟล์ ถ้า True ให้ทำการถามการยืนยันยังจาก user ถ้า yes

ให้ทำการลบไฟล์ทิ้ง และแจ้งเสร็จสิ้น ถ้า user ใส่อย่างอื่นให้แจ้งเตือน

```
def del_file_record(filePath):
    check = os.path.exists(filePath)
    if check != True:
        print("Error: File Not Found!!")
    else:
        q = input("Are you sure you want to delete file record? (yes/no)")
        if q.lower() == "yes":
            os.remove(filePath)
            print(Green + f"{filePath} Removed!!" + Reset)
        else:
            print(Red + "Removing has been Stopped!!" + Reset)
    print()
```

ภาพที่ 2-34