- 4 CRUD functions for Payment Method class.

PaymentMethod.py

```python
def create(self, paymentMethodCode, paymentMethodName):

    data, columns = self.db.fetch ("SELECT * FROM payment_method WHERE payment_method_code = '{}' ".format(paymentMethodCode))
    if len(data) > 0:
        return {'Is Error': True, 'Error Message': "Payment Method Code '{}' already exists. Cannot Create. ".format(paymentMethodCode)}
    else:
        self.db.execute ("INSERT INTO payment_method (payment_method_code,payment_method_name) VALUES ('{}' ,'{}')".format(paymentMethodCode, paymentMethodName))
    return {'Is Error': False, 'Error Message': ""}

def read(self, paymentMethodCode):
    data, columns = self.db.fetch ("SELECT payment_method_code,payment_method_name FROM payment_method WHERE payment_method_code = '{}' ".format(paymentMethodCode))
    if len(data) > 0:
        retPaymentMethod = row_as_dict(data, columns)
    else:
        return ({'Is Error': True, 'Error Message': "Payment Method Code'{}' not found. Cannot Read.".format(paymentMethodCode)},{})

    return ({'Is Error': False, 'Error Message': ""},retPaymentMethod)

def update(self, paymentMethodCode, newPaymentMethodName):
    data, columns = self.db.fetch ("SELECT * FROM payment_method WHERE payment_method_code = '{}' ".format(paymentMethodCode))
    if len(data) > 0:
        self.db.execute ("UPDATE payment_method SET payment_method_name='{}' WHERE payment_method_code='{}' ".format(newPaymentMethodName, paymentMethodCode))
    else:
        return {'Is Error': True, 'Error Message': "Payment Method Code '{}' not found. Cannot Update.".format(paymentMethodCode)}

    return {'Is Error': False, 'Error Message': ""}

def delete(self, paymentMethodCode):
    data, columns = self.db.fetch ("SELECT * FROM payment_method WHERE payment_method_code = '{}' ".format(paymentMethodCode))
    if len(data) > 0:
        self.db.execute ("DELETE FROM payment_method WHERE payment_method_code ='{}' ".format(paymentMethodCode))
    else:
        return {'Is Error': True, 'Error Message': "Payment Method Code '{}' not found. Cannot Delete".format(paymentMethodCode)}
    return {'Is Error': False, 'Error Message': ""}

def dump(self):
    data, columns = self.db.fetch ('SELECT payment_method_code as "Payment Method Code", payment_method_name as "Payment Method Name" FROM payment_method ')
    return row_as_dict(data, columns)
```

create payment Method

read payment Method

update payment Method

delet payment Method

lab3_paymentMethod.py

```python
def main():
#main function begins here
    try:

        #Payment Method
        print("")
        print("*"*50,"create Payment Method","*"*50)
        PaymentMethods = PaymentMethod()
        create_PaymentMethod(PaymentMethods, "CC", "Credit Card")
        create_PaymentMethod(PaymentMethods, "DC", "Debit Card")
        create_PaymentMethod(PaymentMethods, "PP", "Prompt Pay")
        create_PaymentMethod(PaymentMethods, "TM", "Ture Money")
        report_list_payment_methods(PaymentMethods)
        waitKeyPress("Above are results for creating.")

        print("")
        print("*"*50,"Read Payment Method","*"*50)
        read_PaymentMethod(PaymentMethods, "CC")
        read_PaymentMethod(PaymentMethods, "DD") #error

        print("")
        print("*"*50,"update Payment Method","*"*50)
        update_PaymentMethod(PaymentMethods, "CC", "Credit")
        update_PaymentMethod(PaymentMethods, "DC", "Debit")
        report_list_payment_methods(PaymentMethods)
        waitKeyPress("Results after 2 reads, 2 updates to correct Intle spelling.")

        print("")
        print("*"*50,"delete Payment Method","*"*50)
        delete_PaymentMethod(PaymentMethods, "DD") #error
        delete_PaymentMethod(PaymentMethods, "TM")
        report_list_payment_methods(PaymentMethods)
        waitKeyPress("Results after deleting TT (not exist error).")


    except: #this traps for unexpected system errors
        print ("Unexpected error:", sys.exc_info()[0])
        raise # this line can be erased. It is here to raise another error so you can see which line to debug.
    else:
        print("Normal Termination.   Goodbye!")
#main function ends

#this is so that when called externally via a command line, main is executed.
if __name__ == "__main__":
    main()
```

แสดงผลการรัน lab3_paymentMethod.py

```
D:\KMUTT\year3\term1\CPE231_DATA\lab3>python lab3_paymentMethod.py

********************************************** create Payment Method **********************************************
Payment Method Create Success.
Payment Method Create Success.
Payment Method Create Success.
Payment Method Create Success.
{'CC': {'Payment Method Name': 'Credit Card'}, 'DC': {'Payment Method Name': 'Debit Card'}, 'PP': {'Payment Method Name': 'Prompt Pay'}, 'TM': {'Payment Method Name': 'Ture Money'}}
Above are results for creating. (Press a key to continue).

********************************************** Read Payment Method **********************************************
{'CC': {'payment_method_name': 'Credit Card'}}
Payment Method Code'DD' not found. Cannot Read.

********************************************** update Payment Method **********************************************
Payment Method Update Success.
Payment Method Update Success.
{'PP': {'Payment Method Name': 'Prompt Pay'}, 'TM': {'Payment Method Name': 'Ture Money'}, 'CC': {'Payment Method Name': 'Credit'}, 'DC': {'Payment Method Name': 'Debit'}}
Results after 2 reads, 2 updates to correct Intle spelling. (Press a key to continue).

********************************************** delete Payment Method **********************************************
Payment Method Code 'DD' not found. Cannot Delete
Payment Method Delete Success.
{'PP': {'Payment Method Name': 'Prompt Pay'}, 'CC': {'Payment Method Name': 'Credit'}, 'DC': {'Payment Method Name': 'Debit'}}
Results after deleting TT (not exist error). (Press a key to continue).
Normal Termination.   Goodbye!
```

ตารางใน PgAdmin 4

	payment_method_code [PK] character varying (10)	payment_method_name character varying (20)
1	CC	Credit
2	DC	Debit
3	PP	Prompt Pay

- def report_list_all_receipts(receipts, invoices, customers). Please show customer name in the header and also the invoice date as part of invoice information in line item.
- 4 CRUD functions for Receipt class plus an Update and a Delete function for the receipt line items.

Receipt.py

```python
from DBHelper import DBHelper
from helper_functions import *
from Product import *
from Customer import *

class Receipt:
    def __init__(self):
        self.db = DBHelper()

    def __updateReceiptTotal (self, receiptNo):

        sql = ("UPDATE receipt SET "
               "total_receipt = new_total_receipt"
               " FROM (SELECT rli.receipt_no , SUM(rli.amount_paid_here) As new_total_receipt From receipt_line_item rli GROUP BY rli.receipt_no) rli "
               " Where receipt.receipt_no = rli.receipt_no "
               "AND receipt.receipt_no = '{}' ".format(receiptNo))
        self.db.execute (sql)


    def __updateLineItem (self, receiptNo, receiptLineItemList):
        self.db.execute ("DELETE FROM receipt_line_item WHERE receipt_no = '{}' ".format(receiptNo))
        for lineItem in receiptLineItemList:
            self.db.execute ("INSERT INTO receipt_line_item (receipt_no, invoice_no, amount_paid_here) VALUES ('{}' ,'{}','{}')".format(receiptNo,lineItem["Invoice No"],lineItem["Amount Paid Here"]))
        self.__updateReceiptTotal(receiptNo)


    def create(self, receiptNo, receiptDate, customerCode, paymenMethod, paymenReference, remark, receiptLineItemList):
        data, columns = self.db.fetch ("SELECT * FROM receipt WHERE receipt_no = '{}' ".format(receiptNo))
        if len(data) > 0:
            return {'Is Error': True, 'Error Message': "Receipt No '{}' already exists. Cannot Create. ".format(receiptNo)}
        else:
            self.db.execute ("INSERT INTO receipt (receipt_no, receipt_date, customer_code, payment_method, payment_reference, remark) VALUES ('{}' ,{} ,'{}' ,'{}' ,'{}' ,'{}')".format(receiptNo,receiptDate,customerCode,paymenMethod,paymenR
            self.__updateLineItem(receiptNo, receiptLineItemList)

        return {'Is Error': False, 'Error Message': ""}

    def read(self, receiptNo):
        data, columns = self.db.fetch ("SELECT receipt_no, receipt_date, customer_code, payment_method, payment_reference, remark FROM receipt WHERE receipt_no = '{}' ".format(receiptNo))
        if len(data) > 0:
            retReceipt = row_as_dict(data, columns)
        else:
            return ({'Is Error': True, 'Error Message': "Receipt No '{}' not found. Cannot Read.".format(receiptNo)},{})

        return ({'Is Error': False, 'Error Message': ""},retReceipt)

    def update(self, receiptNo, newReceiptDate, newCustomerCode, newPaymenMethod, newPaymenReference, newRemark ,newReceiptLineItemList):
        # Finds the invoice number in invoices object and then changes the values to the new ones.
        # Returns dictionary {'Is Error': ___, 'Error Message': _____}.
        data, columns = self.db.fetch ("SELECT * FROM receipt WHERE receipt_no = '{}' ".format(receiptNo))
        if len(data) > 0:
            self.db.execute ("UPDATE receipt SET receipt_date = {}, customer_code = '{}', payment_method = '{}', payment_reference = '{}', remark= '{}' WHERE receipt_no = '{}' ".format(newReceiptDate, newCustomerCode, newPaymenMethod, newPaymen
            self.__updateLineItem(receiptNo, newReceiptLineItemList)
        else:
            return {'Is Error': True, 'Error Message': "Receipt No '{}' not found. Cannot Update.".format(receiptNo)}

        return {'Is Error': False, 'Error Message': ""}

    def delete(self, receiptNo):
        # Finds the invoice number invoices object and removes it from the dictionary.
        # Returns dictionary {'Is Error': ___, 'Error Message': _____}.
        data, columns = self.db.fetch ("SELECT * FROM receipt WHERE receipt_no = '{}' ".format(receiptNo))
        if len(data) > 0:
            self.db.execute ("DELETE FROM receipt WHERE receipt_no = '{}' ".format(receiptNo))
            self.db.execute ("DELETE FROM receipt_line_item WHERE receipt_no = '{}' ".format(receiptNo))
        else:
            return {'Is Error': True, 'Error Message': "Receipt No '{}' not found. Cannot Delete".format(receiptNo)}
        return {'Is Error': False, 'Error Message': ""}


    def dump(self):
        # Will dump all invoice data by returning 1 dictionary as output.

        data, columns = db.fetch ('SELECT r.receipt_no as "Receipt No", r.receipt_date as "Receipt Date", r.customer_code as "Customer Code", r.payment_method as "Payment Method", r.payment_reference as "Payment Reference", r.remark as "Remark"

        return row_as_dict(data, columns)


    def update_receipt_line(self, receiptNo, invoiceNo, newAmountPaid):
        data, columns = self.db.fetch ("SELECT * FROM receipt_line_item WHERE receipt_no = '{}' AND invoice_no = '{}' ".format(receiptNo, invoiceNo))
        if len(data) > 0:
            self.db.execute ("UPDATE receipt_line_item SET amount_paid_here = {} WHERE receipt_no = '{}' AND invoice_no = '{}' ".format(newAmountPaid, receiptNo, invoiceNo))
            self.__updateReceiptTotal(receiptNo)
        else:
            return {'Is Error': True, 'Error Message': "Invoice Code '{}' not found in Receipt No '{}'. Cannot Update.".format(invoiceNo, receiptNo)}

        return {'Is Error': False, 'Error Message': ""}

    def delete_receipt_line(self, receiptNo, invoiceNo):
        data, columns = self.db.fetch ("SELECT * FROM receipt_line_item WHERE receipt_no = '{}' AND invoice_no = '{}' ".format(receiptNo, invoiceNo))
        if len(data) > 0:
            self.db.execute ("DELETE FROM receipt_line_item WHERE receipt_no = '{}' AND invoice_no = '{}' ".format(receiptNo, invoiceNo))
            self.__updateReceiptTotal(receiptNo)

        else:
            return {'Is Error': True, 'Error Message': "Invoice Code '{}' not found in Receipt No '{}'. Cannot Delete.".format(invoiceNo, receiptNo)}

        return {'Is Error': False, 'Error Message': ""}
```

Create Receipt

read Receipt

upder Receipt

delet Receipt

update receipt line item

delete receipt line item

ตาราง receipt ใน pgAdmin4

Data Output | Explain | Messages | Notifications

	receipt_no [PK] character (10)	receipt_date date	customer_code character (10)	payment_method character (20)	payment_reference character (50)	remark character (50)	total_receipt integer
1	RCT1001/20	2020-02-04	CP	Cash	Nothing	Paid all invoices pa...	200
2	RCT1002/20	2020-02-06	Sam	Debit Card	VISA card	Partially paid on IN...	9400

ตาราง receipt line item ใน pgAdmin4

Data Output | Explain | Messages | Notifications

	receipt_no [PK] character (10)	invoice_no [PK] character (10)	amount_paid_here integer
1	RCT1001/20	INT100/20	200
2	RCT1002/20	INT100/20	8000
3	RCT1002/20	INT101/20	1400

นางสาวรพีพรรณ มาสถิตทรัพย์ 61070507218

- def report_unpaid_invoices (invoices, customers, receipts). Returns 2 dictionaries of 1) a dictionary list of invoices with amount remaining with these fields: Invoice Number, Invoice Date, Customer Name, Invoice Amount Due, Invoice Amount Received , Invoice Amount Not Paid. And 2) a footer dictionary that shows: number of Invoices not paid and total of Invoice Amount Not Paid. The receipts object will be used to calculate Invoice Amount Received for each invoice.

ข้อ (1)

ข้อ (2)

ข้อ 1 ใน pgAdmin 4

lab3/postgres@PostgreSQL 12

Query Editor | Query History

```
1  select i.invoice_no AS "Invoice No" , i.date AS "Invoice Date" , c.name AS "Customer Name" , i.amount_due AS "Invoice Amount Due" ,
2         sum(rli.amount_paid_here) AS "Invoice Amount Received" , (i.amount_due - sum(rli.amount_paid_here)) As "Unpaid"
3  FROM receipt r JOIN receipt_line_item rli ON rli.receipt_no = r.receipt_no
4  JOIN invoice i ON i.invoice_no = rli.invoice_no
5  JOIN customer c ON c.customer_code = i.customer_code
6  Group by i.invoice_no, c.name , i.amount_due;
```

Data Output | Explain | Messages | Notifications

	Invoice No character varying (10)	Invoice Date date	Customer Name character varying (100)	Invoice Amount Due numeric (18,2)	Invoice Amount Received bigint	Unpaid numeric
1	INT100/20	2020-01-03	Sam Co., Ltd.	14980.00	8200	6780.00
2	INT101/20	2020-01-04	CPALL	2140.00	1400	740.00

ข้อ 2 ใน pgAdmin 4

lab3/postgres@PostgreSQL 12

Query Editor | Query History

```
1  select count(unpaid) as "Number of invoices not paid", sum(unpaid) as "Total unpaid" , sum("Amount Paid Here") as "Total Receipt"
2  from (SELECT rli."invoice_no" as "Invoice No", i.date as "Invoice Date", c.name as "Customer Name" ,
3        i."amount_due" as "Amount Received", SUM(rli.amount_paid_here) as "Amount Paid Here",
4       (i.amount_due - sum(rli.amount_paid_here)) as "unpaid"
5        FROM receipt r JOIN receipt_line_item rli ON r."receipt_no" = rli."receipt_no"
6        JOIN invoice i ON i."invoice_no" = rli."invoice_no"
7        JOIN customer c ON c."customer_code" = i."customer_code"
8        GROUP BY rli."invoice_no" ,i."date", c."name",i."amount_due") as total_un_re;
```

Data Output | Explain | Messages | Notifications

	Number of invoices not paid bigint	Total unpaid numeric	Total Receipt numeric
1	2	7520.00	9600

***

- code ใส่ไว้ใช้ในหน้ารวม API ๆ Lab3 main function
- ผลลัพธ์ ใส่ไว้ในผลลัพธ์รวม

นางสาวรพีพรรณ มาสถิตทรัพย์ 61070507218

API.PY

①

```python
# API.py
from DBHelper import DBHelper
from helper_functions import *
#this file will contain all API functions calls exposed to outside world for users to use

# function about Product
def create_product(products, code, name, units):
    result = products.create(code, name, units)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Product Create Success.')
    return result #send result for caller program to use

def read_product(products, code):
    result = products.read(code) #returns tuple of (error dict, data dict)
    if result[0]['Is Error']: #in case error
        print(result[0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_product(products, code, newName, newUnits):
    result = products.update(code, newName, newUnits) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Product Update Success.')
    return result #send result for caller program to use

def delete_product(products, code):
    result = products.delete(code)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Product Delete Success.')
    return result #send result for caller program to use

def report_list_products(products):
    result = products.dump()
    #printDictInCSVFormat(result, ('Code',), ('Name', 'Units'))
    print (result)
    return result #send result for caller program to use

# function about Customer
def create_customer(customers, customerCode, customerName, address, creditLimit, country):
    result = customers.create(customerCode, customerName, address, creditLimit, country)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Customer Create Success.')
    return result #send result for caller program to use

def read_customer(customers, customerCode):
    result = customers.read(customerCode) #returns tuple of (error dict, data dict)
    if result[0]['Is Error']: #in case error
        print(result[0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_customer(customers, customerCode, newCustomerName, newAddress, newCreditLimit, newCountry):
    result = customers.update(customerCode, newCustomerName, newAddress, newCreditLimit, newCountry) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Customer Update Success.')
    return result #send result for caller program to use

def delete_customer(customers, customerCode):
    result = customers.delete(customerCode)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Customer Delete Success.')
    return result #send result for caller program to use

def report_list_all_customers(customers):
    result = customers.dump()
    printDictInCSVFormat(result, ('Customer Code',), ('Name', 'Address','Credit Limit', 'Country'))
    return result #send result for caller program to use

# function about Invoice
def create_invoice(invoices, invoiceNo, invoiceDate, customerCode, dueDate, invoiceLineTuplesList):
    if invoiceDate == None:
        invoiceDate = 'null'
    else:
        invoiceDate = "'" + invoiceDate + "'"
    if dueDate == None:
        dueDate = 'null'
    else:
        dueDate = "'" + dueDate + "'"
    result = invoices.create(invoiceNo, invoiceDate, customerCode, dueDate, invoiceLineTuplesList)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Invoice Create Success.')
    return result #send result for caller program to use

def read_invoice(invoices, invoiceNo):
    result = invoices.read(invoiceNo) #returns tuple of (error dict, data dict)
    if result[0]['Is Error']: #in case error
        print(result[0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_invoice(invoices, invoiceNo, newInvoiceDate, newCustomerCode, newDueDate, newInvoiceLineTuplesList):
    if newInvoiceDate == None:
        newInvoiceDate = 'null'
    else:
        newInvoiceDate = "'" + newInvoiceDate + "'"
    if newDueDate == None:
        newDueDate = 'null'
    else:
        newDueDate = "'" + newDueDate + "'"
    result = invoices.update(invoiceNo, newInvoiceDate, newCustomerCode, newDueDate, newInvoiceLineTuplesList) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Invoice Update Success.')
    return result #send result for caller program to use

def delete_invoice(invoices, invoiceNo):
    result = invoices.delete(invoiceNo)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Invoice Delete Success.')
    return result #send result for caller program to use

def update_invoice_line(invoices, invoiceNo, productCode, newQuantity, newUnitPrice):
    result = invoices.update_invoice_line(invoiceNo, productCode, newQuantity, newUnitPrice) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Invoice Line Item Update Success.')
    return result #send result for caller program to use

def delete_invoice_line(invoices, invoiceNo, productCode):
    result = invoices.delete_invoice_line(invoiceNo, productCode) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Invoice Line Item Delete Success.')
    return result #send result for caller program to use

def report_list_all_invoices(invoices, customers, products):
    # will dump all invoices data and return 1 dictionary as a result (with header and line item joined).
    # Please show the customer name and product name also.
    # a helper function such as def print_tabular_dictionary(tabularDictionary) can then be called to print this in a tabular (table-like) form with column headings and data.

    db = DBHelper()
    data, columns = db.fetch ('SELECT i.invoice_no as "Invoice No", i.date as "Date" '
                              ' , i.customer_code as "Customer Code", c.name as "Customer Name" '
                              ' , i.due_date as "Due Date", i.total as "Total", i.vat as "VAT", i.amount_due as "Amount Due" '
                              ' , ili.product_code as "Product Code", p.name as "Product Name" '
                              ' , ili.quantity as "Quantity", ili.unit_price as "Unit Price", ili.extended_price as "Extended Price" '
                              ' FROM invoice i JOIN customer c ON i.customer_code = c.customer_code '
                              '  JOIN invoice_line_item ili ON i.invoice_no = ili.invoice_no '
                              '  JOIN product p ON ili.product_code = p.code '
                              ' ')
    #print (result)
    result = row_as_dict(data, columns)
    printDictInCSVFormat(result, ('Date', 'Customer Code', 'Customer Name','Due Date','Total','VAT','Amount Due'
                                  , 'Product Code', 'Product Name', 'Quantity', 'Unit Price', 'Extended Price'))
    return result #send result for caller program to use

def report_products_sold(invoices, products, dateStart, dateEnd):
    db = DBHelper()
    data, columns = db.fetch ('SELECT p.code as "Code", ili.product_code as "Product Code", p.name as "Product Name" '
                              ' , SUM(ili.quantity) as "Total Quantity Sold", SUM(ili.extended_price) as "Total Value Sold" '
                              ' FROM invoice i JOIN invoice_line_item ili ON i.invoice_no = ili.invoice_no '
                              '  JOIN product p ON ili.product_code = p.code '
                              ' WHERE i.date between \'' + dateStart + '\' and \'' + dateEnd + '\' '
                              '  GROUP BY p.code, ili.product_code, p.name ')
    result = row_as_dict(data, columns)
    data, columns = db.fetch ('SELECT 0 as "Footer", SUM(ili.extended_price) as "Total Value Sold" '
                              '  FROM invoice i JOIN invoice_line_item ili ON i.invoice_no = ili.invoice_no '
                              '   JOIN product p ON ili.product_code = p.code '
                              ' WHERE i.date between \'' + dateStart + '\' and \'' + dateEnd + '\' '
                              ' ')
    result2 = row_as_dict(data, columns)

    printDictInCSVFormat(result, (None), ('Product Code','Product Name', 'Total Quantity Sold', 'Total Value Sold'))
    printDictInCSVFormat(result2, (None), ('Total Value Sold',))
    return result, result2
```

product

Customer

report_list_all_ Customers

invoice

update invoice line

delet invoice line

report list all invoice

report products sold

ต่อ

2

```python
def report_customer_products_sold_list(invoices, products, customers, dateStart, dateEnd):
    db = DBHelper()
    data, columns = db.fetch ('SELECT i.customer_code, c.customer_code as "Customer Code", c.name as "Customer Name" '
                              ' , ili.product_code as "Product Code", p.name as "Product Name" '
                              ' , i.invoice_no as "Invoice No" '
                              ' , SUM(ili.quantity) as "Quantity Sold", SUM(ili.extended_price) as "Value Sold" '
                              ' FROM invoice i JOIN invoice_line_item ili ON i.invoice_no = ili.invoice_no '
                              '   JOIN customer c ON i.customer_code = c.customer_code '
                              '   JOIN product p ON ili.product_code = p.code '
                              ' WHERE i.date between \'' + dateStart + '\' and \'' + dateEnd + '\' '
                              ' GROUP BY i.customer_code, c.customer_code, c.name, i.invoice_no, ili.product_code, p.name ')
    result = row_as_dict(data, columns)
    data, columns = db.fetch ('SELECT 0 as "Footer", SUM(ili.quantity) as "Quantity Sold", SUM(ili.extended_price) as "Value Sold" '
                              ' FROM invoice i JOIN invoice_line_item ili ON i.invoice_no = ili.invoice_no '
                              '   JOIN customer c ON i.customer_code = c.customer_code '
                              '   JOIN product p ON ili.product_code = p.code '
                              ' WHERE i.date between \'' + dateStart + '\' and \'' + dateEnd + '\' '
                              ' ')
    result2 = row_as_dict(data, columns)

    printDictInCSVFormat(result, (None), ('Customer Code','Customer Name', 'Product Code', 'Product Name', 'Invoice No', 'Quantity Sold', 'Value Sold'))
    printDictInCSVFormat(result2, (None), ('Quantity Sold','Value Sold'))
    return result.values(), result2

def report_customer_products_sold_total(invoices, products, customers, dateStart, dateEnd):
    # Will return 2 dictionaries:
    # 1) a dictionary as list customers and the total number and value of products sold to them in the given date range in this format:  Customer Code, Customer Name, Product Code,  Product Name, Total Quantity Sold, Total Value Sold. Here (c
    # And 2) a second footer dictionary showing: t the end also show the sum of Total Quantity Sold, sum of Total Value Sold.
    db = DBHelper()
    data, columns = db.fetch ('SELECT i.customer_code, c.customer_code as "Customer Code", c.name as "Customer Name" '
                              ' , ili.product_code as "Product Code", p.name as "Product Name" '
                              ' , SUM(ili.quantity) as "Total Quantity Sold", SUM(ili.extended_price) as "Total Value Sold" '
                              ' FROM invoice i JOIN invoice_line_item ili ON i.invoice_no = ili.invoice_no '
                              '   JOIN customer c ON i.customer_code = c.customer_code '
                              '   JOIN product p ON ili.product_code = p.code '
                              ' WHERE i.date between \'' + dateStart + '\' and \'' + dateEnd + '\' '
                              ' GROUP BY i.customer_code, c.customer_code, c.name, i.invoice_no, ili.product_code, p.name ')
    result = row_as_dict(data, columns)
    data, columns = db.fetch ('SELECT 0 as "Footer", SUM(ili.quantity) as "Total Quantity Sold", SUM(ili.extended_price) as "Total Value Sold" '
                              ' FROM invoice i JOIN invoice_line_item ili ON i.invoice_no = ili.invoice_no '
                              '   JOIN customer c ON i.customer_code = c.customer_code '
                              '   JOIN product p ON ili.product_code = p.code '
                              ' WHERE i.date between \'' + dateStart + '\' and \'' + dateEnd + '\' '
                              ' ')
    result2 = row_as_dict(data, columns)

    printDictInCSVFormat(result, (None), ('Customer Code','Customer Name', 'Product Code', 'Product Name', 'Total Quantity Sold', 'Total Value Sold'))
    printDictInCSVFormat(result2, (None), ('Total Quantity Sold','Total Value Sold'))
    return result.values(), result2
```

report customer products sold list

report customer products sold total

```python
# function about Payment method
def create_PaymentMethod(PaymentMethods, paymentMethodCode, paymentMethodName):
    result = PaymentMethods.create(paymentMethodCode, paymentMethodName)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Payment Method Create Success.')
    return result #send result for caller program to use

def read_PaymentMethod(PaymentMethods, paymentMethodCode):
    result = PaymentMethods.read(paymentMethodCode) #returns tuple of (error dict, data dict)
    if result[0]['Is Error']: #in case error
        print(result[0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_PaymentMethod(PaymentMethods, paymentMethodCode, newPaymentMethodName):
    result = PaymentMethods.update(paymentMethodCode, newPaymentMethodName) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Payment Method Update Success.')
    return result #send result for caller program to use

def delete_PaymentMethod(PaymentMethods, paymentMethodCode):
    result = PaymentMethods.delete(paymentMethodCode)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Payment Method Delete Success.')
    return result #send result for caller program to use

def report_list_payment_methods(PaymentMethods):
    result = PaymentMethods.dump()
    print(result)
    return result #send result for caller program to use
```

C

R

U

D

Payment method

report list payment methods

```python
#function about Receipt
def create_receipt(receipts, receiptNo, receiptDate, customerCode, paymenMethod, paymenReference, remark, receiptLineItemList):
    if receiptDate == None:
        receiptDate = 'null'
    else:
        receiptDate = "'" + receiptDate + "'"
    result = receipts.create(receiptNo, receiptDate, customerCode, paymenMethod, paymenReference, remark, receiptLineItemList)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
```

C

```python
def read_receipt(receipts, receiptNo):
    result = receipts.read(receiptNo) #returns tuple of (error dict, data dict)
    if result[0]['Is Error']: #in case error
        print(result[0]['Error Message'])
    else:
        print(result[1])
    return result #send result for caller program to use

def update_receipt(receipts, receiptNo, newReceiptDate, newCustomerCode, newPaymenMethod, newPaymenReference, newRemark, newReceiptLineItemList):
    if newReceiptDate == None:
        newReceiptDate = 'null'
    else:
        newReceiptDate = "'" + newReceiptDate + "'"

    result = receipts.update(receiptNo, newReceiptDate, newCustomerCode, newPaymenMethod, newPaymenReference, newRemark, newReceiptLineItemList) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Receipt Update Success.')
    return result #send result for caller program to use

def delete_receipt(receipts, receiptNo):
    result = receipts.delete(receiptNo)#returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Receipt Delete Success.')
    return result #send result for caller program to use
```

R

U

D

Receipt

```python
def report_list_all_receipts(receipts, invoices, customers):
    db = DBHelper()
    data, columns = db.fetch ('SELECT r.receipt_no as "Receipt No", r.receipt_date as "Receipt Date", r.customer_code as "Customer Code", c.name as "Customer Name" '
                              ', r.payment_method as "Payment Method", r.payment_reference as "Payment Reference", r.remark as "Remark", rli.amount_paid_here As "Total Receipt"'
                              ', i.invoice_no AS "Invoice No", i.Date AS "Invoice Date", i.amount_due AS "Invoice Amount Due" ,rli.amount_paid_here AS "Invoice Amount Received"'
                              'FROM receipt r JOIN receipt_line_item rli ON rli.receipt_no = r.receipt_no '
                              'JOIN customer c ON c.customer_code = r.customer_code '
                              'JOIN invoice i ON i.invoice_no = rli.invoice_no')

    #print (result)
    result = row_as_dict(data, columns)
    #printDictInCSVFormat(result, ('Receipt No',), ('Receipt Date', 'Customer Code', 'Customer Name', 'Payment Method','Payment Reference','Remark','Total Receipt','Invoice No','Invoice Date','Invoice Amount Due',
    #send result for caller program to use
    print(str(columns)[1:-1])
    for i in data:
        s = []
        for j in i:
            s.append(str(j))
        print(', '.join(s))
    return result
```

report list all receipts

```python
def update_receipt_line(receipts, receiptNo, invoiceNo, newAmountPaid):
    result = receipts.update_receipt_line(receiptNo, invoiceNo, newAmountPaid) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Receipt Line Item Update Success.')
    return result #send result for caller program to use


def delete_receipt_line(receipts, receiptNo, invoiceNo):
    result = receipts.delete_receipt_line(receiptNo, invoiceNo) #returns error dictionary
    if result['Is Error']: #if error
        print(result['Error Message'])
    else:
        print('Receipt Line Item Delete Success.')
    return result #send result for caller program to use
```

update receipt line

delet receipt line

```python
def report_unpaid_invoices(invoices,customers,receipts):
    db = DBHelper()
    data, columns = db.fetch (' select i.invoice_no AS "Invoice No" , i.date AS "Invoice Date" , c.name AS "Customer Name" , i.amount_due AS "Invoice Amount Due" ,'
                              'sum(rli.amount_paid_here) AS "Invoice Amount Received" , (i.amount_due - sum(rli.amount_paid_here)) As "Unpaid"'
                              ' FROM receipt r JOIN receipt_line_item rli ON rli.receipt_no = r.receipt_no '
                              'JOIN invoice i ON i.invoice_no = rli.invoice_no '
                              'JOIN customer c ON c.customer_code = i.customer_code '
                              'Group by i.invoice_no, c.name , i.amount_due;')

    #print (result)
    result = row_as_dict(data, columns)

    db = DBHelper()
    data, columns = db.fetch (' select 0 as "Footer", count(unpaid) as "Number of invoices not paid", sum(unpaid) as "Total unpaid" , sum("Amount Paid Here") as "T
                              ' from (SELECT rli."invoice_no" as "Invoice No", i.date as "Invoice Date", c.name as "Customer Name" , '
                              ' i."amount_due" as "Amount Received", SUM(rli.amount_paid_here) as "Amount Paid Here", '
                              ' (i.amount_due - sum(rli.amount_paid_here)) as "unpaid" '
                              ' FROM receipt r JOIN receipt_line_item rli ON r."receipt_no" = rli."receipt_no"'
                              ' JOIN invoice i ON i."invoice_no" = rli."invoice_no" '
                              ' JOIN customer c ON c."customer_code" = i."customer_code" '
                              ' GROUP BY rli."invoice_no" ,i."date", c."name",i."amount_due") as total_un_re;')

    #print (result)
    result2 = row_as_dict(data, columns)

    printDictInCSVFormat(result, ('Invoice No',), ('Invoice Date', 'Customer Name', 'Invoice Amount Due', 'Invoice Amount Received','Unpaid'))
    printDictInCSVFormat(result2, (None), ('Number of invoices not paid', 'Total unpaid', 'Total Receipt'))

    return result, result2
```

ข้อ ① ตามโจทย์

ข้อ ② ตามโจทย์

แสดงผลการรัน Lab3mainfunction.py

```
D:\KMUTT\year3\term1\CPE231_DATA\lab3>python Lab3mainfunction..py
Product Create Success.
Product Create Success.
Product Create Success.
{'HD01': {'Name': 'Seagate HDD 80 GB', 'Units': 'PCS'}, 'HD02': {'Name': 'IBM HDD 60 GB', 'Units': 'PCS'}, 'INT01': {'Name': 'Intel Pentium IV 3.6 GHz', 'Units': 'PCS'}}
These are 3 products in the database (Press a key to continue).
Customer Create Success.
Customer Create Success.

Customer Code, Name, Address, Credit Limit, Country
Sam, Sam Co., Ltd., 122 Bangkok, 500000, Thailand
CP, CPALL, 123 Bangkok, 100000, Thailand
These are 2 customers in the database (Press a key to continue).
Invoice Create Success.
Invoice Create Success.

Invoice No, Date, Customer Code, Customer Name, Due Date, Total, VAT, Amount Due, Product Code, Product Name, Quantity, Unit Price, Extended Price
INT100/20, 2020-01-03, Sam, Sam Co., Ltd., None, 14000.00, 980.00, 14980.00, HD02, IBM HDD 60 GB, 8, 1000.00, 8000.00
INT101/20, 2020-01-04, CP, CPALL, None, 2000.00, 140.00, 2140.00, HD02, IBM HDD 60 GB, 1, 2000.00, 2000.00
These are 2 invoices in the database (Press a key to continue).
Receipts Create Success.
Receipts Create Success.
Receipts Create Success.
'Receipt No', 'Receipt Date', 'Customer Code', 'Customer Name', 'Payment Method', 'Payment Reference', 'Remark', 'Total Receipt', 'Invoice No', 'Invoice Date', 'Invoice Amount Due', 'Invoice Amount
Received'
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 100, INT100/20, 2020-01-03, 1
4980.00, 100
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 200, INT101/20, 2020-01-04, 2
140.00, 200
RCT1002/20, 2020-02-05, Sam       , Sam Co., Ltd., Credit Card        , Master Card, Citibank                           , Partially paid on INT101/20                     , 8560, INT100/20, 2020
-01-03, 14980.00, 8560
RCT1002/20, 2020-02-05, Sam       , Sam Co., Ltd., Credit Card        , Master Card, Citibank                           , Partially paid on INT101/20                     , 1440, INT101/20, 2020
-01-04, 2140.00, 1440
RCT1003/20, 2020-02-06, CP        , CPALL, Debit Card         , Debit Card                                       , This will later be deleted                       , 10, INT100/20, 2020-01-03, 14
980.00, 10
RCT1003/20, 2020-02-06, CP        , CPALL, Debit Card         , Debit Card                                       , This will later be deleted                       , 20, INT101/20, 2020-01-04, 21
40.00, 20
Results of creating 3 receipts: RCT1001/20, RCT1002/20, and RCT1003/20 (Press a key to continue).
{'RCT1003/20': {'receipt_date': datetime.date(2020, 2, 6), 'customer_code': 'CP        ', 'payment_method': 'Debit Card         ', 'payment_reference': 'Debit Card
      ', 'remark': 'This will later be deleted                        '}}
Receipt No 'RCT1005/20' not found. Cannot Read.
'Receipt No', 'Receipt Date', 'Customer Code', 'Customer Name', 'Payment Method', 'Payment Reference', 'Remark', 'Total Receipt', 'Invoice No', 'Invoice Date', 'Invoice Amount Due', 'Invoice Amount
Received'
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 100, INT100/20, 2020-01-03, 1
4980.00, 100
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 200, INT101/20, 2020-01-04, 2
140.00, 200
RCT1002/20, 2020-02-05, Sam       , Sam Co., Ltd., Credit Card        , Master Card, Citibank                           , Partially paid on INT101/20                     , 8560, INT100/20, 2020
-01-03, 14980.00, 8560
RCT1002/20, 2020-02-05, Sam       , Sam Co., Ltd., Credit Card        , Master Card, Citibank                           , Partially paid on INT101/20                     , 1440, INT101/20, 2020
-01-04, 2140.00, 1440
RCT1003/20, 2020-02-06, CP        , CPALL, Debit Card         , Debit Card                                       , This will later be deleted                       , 10, INT100/20, 2020-01-03, 14
980.00, 10
RCT1003/20, 2020-02-06, CP        , CPALL, Debit Card         , Debit Card                                       , This will later be deleted                       , 20, INT101/20, 2020-01-04, 21
40.00, 20
Results of reading 2 receipts: RCT1003/20 (successfully), and RCT1005/20 (unsuccessfully) (Press a key to continue).
Receipt Update Success.
Receipt No 'RCT1004/20' not found. Cannot Update.
'Receipt No', 'Receipt Date', 'Customer Code', 'Customer Name', 'Payment Method', 'Payment Reference', 'Remark', 'Total Receipt', 'Invoice No', 'Invoice Date', 'Invoice Amount Due', 'Invoice Amount
Received'
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 100, INT100/20, 2020-01-03, 1
4980.00, 100
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 200, INT101/20, 2020-01-04, 2
140.00, 200
RCT1003/20, 2020-02-06, CP        , CPALL, Debit Card         , Debit Card                                       , This will later be deleted                       , 10, INT100/20, 2020-01-03, 14
980.00, 10
RCT1003/20, 2020-02-06, CP        , CPALL, Debit Card         , Debit Card                                       , This will later be deleted                       , 20, INT101/20, 2020-01-04, 21
40.00, 20
RCT1002/20, 2020-02-06, Sam       , Sam Co., Ltd., Debit Card         , VISA card                                       , Partially paid on INT100/20 and INT101/20        , 8000, INT100/20, 2020
-01-03, 14980.00, 8000
RCT1002/20, 2020-02-06, Sam       , Sam Co., Ltd., Debit Card         , VISA card                                       , Partially paid on INT100/20 and INT101/20        , 1400, INT101/20, 2020
-01-04, 2140.00, 1400
Results of updating 2 receipts: RCT1002/20 (successfully) and RCT1004/20 (unsuccessfully) (Press a key to continue).
Receipt Delete Success.
Receipt No 'RCT1005/20' not found. Cannot Delete
'Receipt No', 'Receipt Date', 'Customer Code', 'Customer Name', 'Payment Method', 'Payment Reference', 'Remark', 'Total Receipt', 'Invoice No', 'Invoice Date', 'Invoice Amount Due', 'Invoice Amount
Received'
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 100, INT100/20, 2020-01-03, 1
4980.00, 100
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 200, INT101/20, 2020-01-04, 2
140.00, 200
RCT1002/20, 2020-02-06, Sam       , Sam Co., Ltd., Debit Card         , VISA card                                       , Partially paid on INT100/20 and INT101/20        , 8000, INT100/20, 2020
-01-03, 14980.00, 8000
RCT1002/20, 2020-02-06, Sam       , Sam Co., Ltd., Debit Card         , VISA card                                       , Partially paid on INT100/20 and INT101/20        , 1400, INT101/20, 2020
-01-04, 2140.00, 1400
Results of deleting 2 receipts: RCT1003/20 (successfully) and RCT1005/20 (unsuccessfully) (Press a key to continue).

Invoice No, Invoice Date, Customer Name, Invoice Amount Due, Invoice Amount Received, Unpaid
INT100/20, 2020-01-03, Sam Co., Ltd., 14980.00, 8100, 6880.00
INT101/20, 2020-01-04, CPALL, 2140.00, 1600, 540.00

Number of invoices not paid, Total unpaid, Total Receipt
2, 7420.00, 9700
Report unpaid invoice 1 (Press a key to continue).
Receipt Line Item Update Success.
Invoice Code 'INT100/20' not found in Receipt No 'RCT1005/20'. Cannot Update.
Invoice Code 'INT103/20' not found in Receipt No 'RCT1001/20'. Cannot Update.
'Receipt No', 'Receipt Date', 'Customer Code', 'Customer Name', 'Payment Method', 'Payment Reference', 'Remark', 'Total Receipt', 'Invoice No', 'Invoice Date', 'Invoice Amount Due', 'Invoice Amount
Received'
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 200, INT101/20, 2020-01-04, 2
140.00, 200
RCT1002/20, 2020-02-06, Sam       , Sam Co., Ltd., Debit Card         , VISA card                                       , Partially paid on INT100/20 and INT101/20        , 8000, INT100/20, 2020
-01-03, 14980.00, 8000
RCT1002/20, 2020-02-06, Sam       , Sam Co., Ltd., Debit Card         , VISA card                                       , Partially paid on INT100/20 and INT101/20        , 1400, INT101/20, 2020
-01-04, 2140.00, 1400
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 200, INT100/20, 2020-01-03, 1
4980.00, 200
Result of updating 3 receipt line, first successfully, others unsuccessfully (Press a key to continue).
Receipt Line Item Delete Success.
Invoice Code 'INT100/20' not found in Receipt No 'RCT1003/20'. Cannot Delete.
Invoice Code 'INT103/20' not found in Receipt No 'RCT1001/20'. Cannot Delete.
'Receipt No', 'Receipt Date', 'Customer Code', 'Customer Name', 'Payment Method', 'Payment Reference', 'Remark', 'Total Receipt', 'Invoice No', 'Invoice Date', 'Invoice Amount Due', 'Invoice Amount
Received'
RCT1002/20, 2020-02-06, Sam       , Sam Co., Ltd., Debit Card         , VISA card                                       , Partially paid on INT100/20 and INT101/20        , 8000, INT100/20, 2020
-01-03, 14980.00, 8000
RCT1002/20, 2020-02-06, Sam       , Sam Co., Ltd., Debit Card         , VISA card                                       , Partially paid on INT100/20 and INT101/20        , 1400, INT101/20, 2020
-01-04, 2140.00, 1400
RCT1001/20, 2020-02-04, CP        , CPALL, Cash               , Nothing                                          , Paid all invoices partially                      , 200, INT100/20, 2020-01-03, 1
4980.00, 200
Result of deleting 3 receipt line, first successfully, others unsuccessfully (Press a key to continue).

Invoice No, Invoice Date, Customer Name, Invoice Amount Due, Invoice Amount Received, Unpaid
INT100/20, 2020-01-03, Sam Co., Ltd., 14980.00, 8200, 6780.00
INT101/20, 2020-01-04, CPALL, 2140.00, 1400, 740.00

Number of invoices not paid, Total unpaid, Total Receipt
2, 7520.00, 9600
Normal Termination.   Goodbye!

D:\KMUTT\year3\term1\CPE231_DATA\lab3>
```