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 \_\_updateReceiptAmountUnpaid (self, receiptNo):

        sql = ("UPDATE receipt\_line\_item SET "

                " amount\_unpaid = new\_amount\_unpaid"

                " 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,paymenReference,remark))

            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, newPaymenReference, newRemark,receiptNo))

            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" FROM receipt r JOIN customer c ON r.customer\_code = c.customer\_code')

        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': ""}

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, ('Invoice No',), ('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

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 (customer code, product code) will be unique.

    # 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

# 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

#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:

        print('Receipts Create Success.')

    return result #send result for caller program to use

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

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', 'Invoice Amount Received'))

     #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

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

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 "Total Receipt"'

                            ' 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