orders.py

```python
# Inbuilt Modules
import smtplib
from string import capwords
from time import sleep
import datetime

# Project Modules
import mysql.connector as mycam
import config


# ------------------------------------------------------------------------------------------------ #
# |                            Below Functions are Common                                        | #
# ------------------------------------------------------------------------------------------------ #

# ---------------------------------------------------------------------
''' This function will allow every user in database to place order for any book if it exists in database.
    User is prompted every time to place another order if he/she wants.
    When user says 'n' or 'no' function are called from other modules to generate invoice and invoice number.'''

def Place_ord(username):
    ord_id_ls = []
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SET SQL_SAFE_UPDATES = 0")

    def work():
        print()
        book_name = str(input('Enter the name of the book ' + ': ')).lower()
        cursor.execute("SELECT * FROM books WHERE title='{}'".format(book_name))
        data = cursor.fetchone()
        if data == None:
```

```
data = cursor.fetchone()
if data == None:
    print('Sorry, No book named', book_name, 'found.')
else:
    print(f"   Book named , {capwords(book_name)},  found details are as follows:-")
    print("    Author's name:           ", capwords(data[2]) + ' ' + capwords(data[3]))
    print("    Released Year:           ", data[4])
    print("    Number of Pages:         ", data[8])
    print("    Words Per Page (Approx): ", data[9])
    print("    Publication:             ", capwords(data[6]))
    print("    Category of Book:        ", capwords(data[5]))
    print("    Available quantity:      ", data[7])
    print("    Price of the book:       ", data[10])
    choice = str(input("Do you want to order this Book ? "))

    if choice in ['yes', 'Yes', 'YES', 'y', 'Y']:
        quantity = int(input('Enter the QUANTITY you want to order: '))
        print('              Checking Stock, please wait...')
        sleep(2)

        if quantity <= data[7]:    # Checking if pieces specified to be bought are present or not for the specified book.
            print('              Enough quantity found')
            print('Please fill up your details below: ')

            name = str(input('     Enter Name: '))
            address = str(input('     Enter Address: '))
            dist = str(input('     Enter District: '))
            city = str(input('     Enter City: '))
            stu = str(input('     Enter State/UT: '))
            pin = int(input("     Enter Pincode: "))
            country = str(input('     Enter Country: '))
            phone = int(input('     Enter Phone number: '))
```

```
63
64          # Email Address is fetched from the accounts table. This email address will be used to send email
65          cursor.execute(f"SELECT email FROM accounts where username = '{username}'")
66          email = cursor.fetchone()
67          amount = data[10] * quantity
68          cursor.execute(f"SELECT ID FROM accounts WHERE username = '{username}'")
69          custom_id = cursor.fetchone()
70          print('  Now we came at the final step of the process...')
71          print("             Total Amount to be paid is: ", amount)
72          print("             No Delivery Charges Applicable, i.e. Delivery Free")
73          confirm = str(input('  Are you sure you want to confirm this order? '))
74
75          if confirm in ['yes', 'Yes', 'YES', 'y', 'Y']:
76              details = [name, address, dist, city, stu, pin, country, phone, custom_id]
77              cursor.execute(
78                  f"UPDATE books SET stock_quantity = stock_quantity - {quantity} WHERE title = '{book_name}'")
79              cam.commit()
80              cursor.execute(
81                  f"INSERT INTO orders "
82                  f"(customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, "
83                  f"user_name, order_type) "
84                  f"VALUES ('{name}', '{address}', '{dist}', '{city}', '{stu}', {pin}, '{country}', '{email[0]}', {phone}, {quantity}, "
85                  f"'{book_name}', '{username}', 'new')")
86              cam.commit()
87              cursor.execute(
88                  f"SELECT ord_no, order_dt FROM orders WHERE customer_name = '{name}' and phone_no = {phone} ORDER BY ord_no DESC")
89              data2 = cursor.fetchone()
90              ord_id_ls.append(data2[0])
91              print("Processing your order....")
92              sleep(2)
93              print()
94              print("               Order Placed Successfully.")
```

```
                        print()
                        print("              Order Placed Successfully.")
                    else:
                        print('Your order has been discarded')

                else:
                    print('Sorry', data[7], 'pieces are left')
                    print('Thanks for visiting, keep shopping :)')
            elif choice in ['no', 'No', 'NO', 'n', 'N']:
                print('Okay.')
            else:
                print('Invalid Input')

        print()
        ch = str(input("Want to place more orders? (y/n): ")).lower()
        if ch in ['y', 'yes']:
            work()
        else:
            details.append(email[0])
            details.append(username)
            inv = config.Inv_no_gen(username)
            config.Pdf_generate(details, ord_id_ls, inv[0], inv[1])
            print("Thanks for visiting us :)")
            print()
    work()
    return

# ---------------------------------------------------------------------
'''This function will be used only if the order is specified for return/replace or order is cancelled
   For that, message and subject or email is defined in their respective functions.
   Sender's Email Address and Password is given in config module of the project
   An email will be sent to the email address of the user with the details.'''
```

```
125
126 def Send_email(email, subject, msg):
127     try:
128         server = smtplib.SMTP('smtp.gmail.com:587')
129         server.ehlo()
130         server.starttls()
131         server.login(config.EMAIL_ADDRESS, config.PASSWORD)
132         final_message = 'Subject: {}\n\n{}'.format(subject, msg)
133         server.sendmail(config.EMAIL_ADDRESS, email, final_message)
134         server.quit()
135         print(f"An email with all details is sent to you on your E-mail Address {email}")
136     except:
137         print("Email failed to send.")
138     return
139
140 # -----------------------------------------------------------------
141 '''This function will allow the user to return or replace the order within 7 days after the delivery date.
142    Else a message will be displayed that the order is not eligible for passing the Return/Replace Policy.
143    By default, the delivery date is the date 7 days after the date of placing the order'''
144
145 def Return_replace(username):
146     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
147     cursor = cam.cursor(buffered=True)
148     cursor.execute(f"SELECT ord_no FROM orders WHERE user_name = '{username}'")
149     record = cursor.fetchall()
150     L = []
151     if not record:
152         print('No order to return or replace')
153     else:
154         cursor.execute(
155             f"SELECT * from orders where user_name = '{username}' and DATEDIFF(NOW(), order_dt) >= 7 and DATEDIFF(NOW(), order_dt) <= 14 "
156             f"ORDER by ord_no DESC")
```

```
data = cursor.fetchall()
if not data:
    print('None of your order is eligible to return or replace.')
else:
    print('We have found', len(data), 'orders have been delivered to you')
    print('List of the orders that have been delivered to you are as follows:-')
    for i in data:
        L.append(i[0])
        print('   Book Name      : ', i[11])
        print('   Order Date     : ', i[12])
        print('   Order ID       : ', i[0])
        print('   Customer Name  : ', i[1])
        print(f"   Location       :  {i[2]}, {i[3]}, {i[4]}, {i[5]}, {i[6]}, {i[7]}")
        print('   Email          : ', i[8])
        print('   Phone no       : ', i[9])
        print('   No of Books    : ', i[10])
        print()

    choice = str(input('Do you want to return or replace any of the above orders (y/n) ')).lower()

    def work(c):
        if c == 'y':
            ID = int(input('Enter the order ID for which you want to return or replace: '))
            if ID not in L:
                print('No order ID ', ID, ' matched from the orders that have been delivered to you')
            else:
                print('You have chosen order ID: ', ID)
                print('Few details of your order are as follows: ')
                print()
                cursor.execute(f"SELECT * FROM orders WHERE ord_no = {ID}")
                data2 = cursor.fetchone()
                print('   Book name: ', data2[11])
```

```
        print('   Book Name      : ', i[11])
        print('   Order Date     : ', i[12])
        print('   Order ID       : ', i[0])
        print('   Customer Name  : ', i[1])
        print(f"   Location       :  {i[2]}, {i[3]}, {i[4]}, {i[5]}, {i[6]}, {i[7]}")
        print('   Email          : ', i[8])
        print('   Phone no       : ', i[9])
        print('   No of Books    : ', i[10])
        print()

choice = str(input('Do you want to return or replace any of the above orders (y/n) ')).lower()

def work(c):
    if c == 'y':
        ID = int(input('Enter the order ID for which you want to return or replace: '))
        if ID not in L:
            print('No order ID ', ID, ' matched from the orders that have been delivered to you')
        else:
            print('You have chosen order ID: ', ID)
            print('Few details of your order are as follows: ')
            print()
            cursor.execute(f"SELECT * FROM orders WHERE ord_no = {ID}")
            data2 = cursor.fetchone()
            print('   Book name: ', data2[11])
            print('   Quantity: ', data2[10])
            print('   Order date: ', data2[12])
            print()
            print('By SNT.bookshop Return and Replace policies, you have below two options')
            print('1. Return')
            print('2. Replace')

            opt = int(input('Enter the option number that you want to select: '))
```

```
197
198                 if opt == 1:
199                     print('You have entered under our Return policy')
200                     confirm = str(input('Are you sure you want to return this order? (y/n) ')).lower()
201                     if confirm in ['y', 'yes']:
202                         print('   Processing the Return')
203                         msg = f"Your request for returning your order has been accepted. Some details for it are as follows: \n" \
204                               f"Book Name: {capwords(data2[11])} \n" \
205                               f"Order ID: {data2[0]} \n" \
206                               f"Name: {data2[1]} \n" \
207                               f"Address: {data2[2]} \n" \
208                               f"District: {data2[3]} \n" \
209                               f"City: {data2[4]} \n" \
210                               f"State: {data2[5]} \n" \
211                               f"Pincode: {data2[6]} \n" \
212                               f"Country: {data2[7]} \n" \
213                               f"Phone: {data2[9]} \n" \
214                               f"Pieces Ordered: {data2[10]} \n" \
215                               f"Order Date & Time: {data2[12]} \n"
216                         sub = f"Information of Returning Order on S & T Bookshop"
217                         Send_email(data2[8], sub, msg)  # Calling Function to send email
218                         # Real Email is sent here.
219                         # Email is sent actually to the mailbox if the given email really exists
220                         cursor.execute(
221                             f"INSERT INTO orders "
222                             f"(customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, "
223                             f"user_name, order_type) VALUES ('{data2[1]}', '{data2[2]}', '{data2[3]}', '{data2[4]}', '{data2[5]}', "
224                             f"{data2[6]}, '{data2[7]}', '{data2[8]}', {data2[9]}, {data2[10]}, "
225                             f"'{data2[11]}', '{username}', 'return')")
226                         cam.commit()
227                         cursor.execute(f"DELETE FROM orders WHERE ord_no = {ID}")
228                         cam.commit()
```

```
                    print('   Successfully requested for return to your order.')
                    print(
                        '   Our courier boy will come within the next day to collect the package after verifying its condition,
                        '   be ready with the packed book')
                    print('   Money will be refund to your account within 2-3 days')
                    return
                else:
                    print('   Return process cancelled')

            elif opt == 2:
                print('You have entered under our Replace policy.')
                confirm = str(input('Are you sure you want to replace for this order? (y/n) ')).lower()

                if confirm in ['y', 'yes']:
                    msg = f"Your request for replacing your order has been accepted. Some details for it are as follows: \n" \
                          f"Book Name: {capwords(data2[11])} \n" \
                          f"Order ID: {data2[0]} \n" \
                          f"Name: {data2[1]} \n" \
                          f"Address: {data2[2]} \n" \
                          f"District: {data2[3]} \n" \
                          f"City: {data2[4]} \n" \
                          f"State: {data2[5]} \n" \
                          f"Pincode: {data2[6]} \n" \
                          f"Country: {data2[7]} \n" \
                          f"Phone: {data2[9]} \n" \
                          f"Pieces Ordered: {data2[10]} \n" \
                          f"Order Date & Time: {data2[12]} \n" \
                          f"Delivery will be provided within 7 working days. \n" \
                          f"Sorry for the inconvenience :)"
                    sub = f"Information of Replacing Order on S & T Bookshop"
                    print("   Successfully requested for replacement of your order.")
                    Send_email(data2[8], sub, msg)  # Calling Function to send email
```

```
                    send_email(data2[0], sub, msg)  # calling function to send email
                    # Real Email is sent here.
                    # Email is sent actually to the mailbox if the given email really exists
                    cursor.execute(
                        f"INSERT INTO orders "
                        f"(customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, "
                        f"user_name, order_type) VALUES ('{data2[1]}', '{data2[2]}', '{data2[3]}', '{data2[4]}', '{data2[5]}', {data2[6]}, "
                        f"'{data2[7]}', '{data2[8]}', {data2[9]}, {data2[10]}, "
                        f"'{data2[11]}', '{username}', 'replace')")
                    cam.commit()
                    cursor.execute(f"DELETE FROM orders WHERE ord_no = {ID}")
                    cam.commit()
                    print(
                        '   Our courier boy will come within the next day to collect the package after verifying its condition, '
                        '   be ready with the packed book')
                    print(
                        "   Replacement will be completed within 7 working days until your order has been delivered. ")
                elif confirm in ['n', 'no']:
                    print("Replacement Cancelled")
                    print("Exiting")
                    return
                else:
                    print('Invalid option choosen.')
            elif c in ['n', 'no']:
                print("Exiting")
                return
            else:
                print("Wrong Input")
                again_t = str(input("Do you want to return or replace any of the above orders? (y/n) ")).lower()
                if again_t in ['yes', 'y']:
                    work(again_t)
```

```
                    elif again_t in ['n', 'no']:
                        print("Exiting")
                        return
                    else:
                        print("Invalid Input.")

            work(choice)
        cam.close()
        return


# ------------------------------------------------------------------------------------------------ #
# |                            Below Functions are for Administrator                               |
# ------------------------------------------------------------------------------------------------ #

# ---------------------------------------------------------------------
'''This function will allow the administrator to cancel as many orders he wants by specifying the Order ID
   An email will be sent to the user with details. Order can be cancelled only within 7 days after placing order'''

def Cancel_ord():  # Admin can cancel any user's order
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SELECT ord_no FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7")
    rec = cursor.fetchall()
    if len(rec) == 0:
        print("No orders to cancel.")
        print("Sorry")
    else:
        print(f"Only {len(rec)} order(s) is/are eligible to cancel.")
        num = int(input(f"Enter number of orders you want to cancel (not more than {len(rec)}): "))
        if num <= len(rec):
```

```
321         if num <= len(rec):
322             for i in range(num):
323                 order_id = int(input('  Enter the Order ID: '))
324                 cursor.execute(f"SELECT ord_no FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {order_id}")
325                 data3 = cursor.fetchone()
326                 if len(data3) == 1:
327                     cursor.execute(f"SELECT * FROM orders WHERE ord_no = {order_id}")
328                     data = cursor.fetchone()
329                     if data == None:
330                         print(f"Order ID {order_id} does not exist.")
331                     else:
332                         cursor.execute("SET SQL_SAFE_UPDATES = 0")
333                         cursor.execute(
334                             f"UPDATE books SET stock_quantity = stock_quantity + {data[10]} WHERE title = '{data[11]}'")
335                         cam.commit()
336                         cursor.execute(f"SELECT publication, author_fname, author_lname FROM books WHERE title = '{data[11]}'")
337                         data2 = cursor.fetchone()
338                         msg = f"Your Order has been cancelled as per your request. Please review the product on our website. \n" \
339                               f"Book Name: {capwords(data[11])} \n" \
340                               f"Publication: {capwords(data2[0])} \n" \
341                               f"Author's Name: {capwords(data2[1])} {capwords(data2[2])} \n" \
342                               f"Order ID: {data[0]} \n" \
343                               f"Name: {capwords(data[1])} \n" \
344                               f"Address: {data[2]} \n" \
345                               f"District: {data[3]} \n" \
346                               f"City: {data[4]} \n" \
347                               f"State: {data[5]} \n" \
348                               f"Pincode: {data[6]} \n" \
349                               f"Country: {data[7]} \n" \
350                               f"Phone: {data[9]} \n" \
351                               f"Pieces Ordered: {data[10]} \n" \
352                               f"Order Date & Time: {data[12]}"
```

```
                                f"Pieces Ordered: {data[10]} \n" \
                                f"Order Date & Time: {data[12]}"
                            email = data[8]
                            sub = 'Regarding Cancellation of Order on S & T Bookshop'
                            Send_email(email, sub, msg)
                            cursor.execute(f"DELETE FROM orders WHERE ord_no = {order_id}")
                            cam.commit()
                            print("Successfully cancelled your order.")
                        else:
                            print("Order not eligible to cancel.")
                    else:
                        print(f"Not possible to cancel more than {len(rec)} orders.")
                        Cancel_ord()
            cam.close()
            return

# ---------------------------------------------------------------------
'''This function will allow administrator to update the order address of multiple orders'''

def Update_ord_addr():  # Admin can update any user's order
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SELECT * FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7")
    lim = cursor.fetchall()
    if not lim:
        print("No orders are eligible for update of address.")
    else:
        print(f"Only {len(lim)} order(s) is/are eligible for update of address.")

        def work(data):
            num = int(input('Enter the number of orders\'s addresses you want to update: '))
            lst = [(data[i][0]) for i in range(len(data))]
```

```
383         if num <= len(data):
384             for i in range(num):
385                 order_id = int(input('Enter the Order ID to update the order: '))
386                 if order_id in lst:
387                     new_addr = str(input('     Enter the new address for order ID ' + str(order_id) + ': '))
388                     new_city = str(input('     Enter the new city for order ID    ' + str(order_id) + ': '))
389                     new_pinc = int(input('     Enter the new pincode for order ID ' + str(order_id) + ': '))
390                     new_dist = str(input('     Enter the new district for Order ID' + str(order_id) + ': '))
391                     new_stat = str(input('     Enter the new State/UT for Order ID' + str(order_id) + ': '))
392                     cursor.execute("SET SQL_SAFE_UPDATES = 0")
393                     cursor.execute("UPDATE orders SET address = '{}' WHERE ord_no = {}".format(new_addr, order_id))
394                     cam.commit()
395                     cursor.execute("UPDATE orders SET city = '{}'  WHERE ord_no = {}".format(new_city, order_id))
396                     cam.commit()
397                     cursor.execute("UPDATE orders SET pincode = {}  WHERE ord_no = {}".format(new_pinc, order_id))
398                     cam.commit()
399                     cursor.execute("UPDATE orders SET district = '{}'  WHERE ord_no = {}".format(new_dist, order_id))
400                     cam.commit()
401                     cursor.execute("UPDATE orders SET state_ut = '{}'  WHERE ord_no = {}".format(new_stat, order_id))
402                     cam.commit()
403                     cursor.execute(f"SELECT e_mail FROM orders WHERE ord_no = {order_id}")
404                     email = cursor.fetchone()
405                     msg = f"Your Order has been updated successfully \n. " \
406                           f"New Location of the order is {new_addr}, {new_city}, {new_pinc}, {new_city}, {new_stat} :)"
407                     sub = "Information of Order Update on S & T Book Shop Management System"
408                     Send_email(email[0], sub, msg)
409                     print('Successfully Updated')
410                     print(f"regarding order update for new location on email address {email[0]}")
411                 else:
412                     print("This order ID is not eligible for update of order address.")
413                     continue
414         else:
```

```
                continue
            else:
                print("Invalid Input. Try Again")
                work(data)
        work(lim)
    cam.close()
    return

# ---------------------------------------------------------------------
'''This function will allow administrator to view all orders in database of every user.'''

def View_all_ords():  # Admin can view all user's order
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    print("  1 -> View all New Orders Placed \n"
          "  2 -> View all Orders for Return \n"
          "  3 -> View all Orders for Replacement \n"
          "  4 -> View all Orders including all categories")
    ch = int(input("  Enter your choice (1 to 4): "))

    def start(choice):
        if choice == 1:
            cursor.execute("SELECT * FROM orders WHERE order_type = 'new' ORDER BY ord_no DESC")
            data1 = cursor.fetchall()
            if not data1:
                print("No orders.")
            else:
                for row in data1:
                    tp = row
                    print("    Order ID       : ", tp[0])
                    print("    Book Name      : ", capwords(tp[11]))
                    print("    Pieces         : ", tp[10])
```

```
445                     print("    Customer Name  : ", capwords(tp[1]))
446                     print("    Address        : ", tp[2])
447                     print("    District       : ", tp[3])
448                     print("    City           : ", tp[4])
449                     print("    State          : ", tp[5])
450                     print("    Pincode        : ", tp[6])
451                     print("    Country        : ", tp[7])
452                     print("    E-mail Address : ", tp[8])
453                     print("    Phone          : ", tp[9])
454                     print("    Order Date     : ", tp[12])
455                     print("    Username       : ", tp[13])
456                     print()
457         elif choice == 2:
458             cursor.execute("SELECT * FROM orders WHERE order_type = 'return' ORDER BY ord_no DESC")
459             data2 = cursor.fetchall()
460             if not data2:
461                 print("No orders.")
462             else:
463                 for row in data2:
464                     tp = row
465                     print("    Order ID       : ", tp[0])
466                     print("    Book Name      : ", capwords(tp[11]))
467                     print("    Pieces         : ", tp[10])
468                     print("    Customer Name  : ", capwords(tp[1]))
469                     print("    Address        : ", tp[2])
470                     print("    District       : ", tp[3])
471                     print("    City           : ", tp[4])
472                     print("    State          : ", tp[5])
473                     print("    Pincode        : ", tp[6])
474                     print("    Country        : ", tp[7])
475                     print("    E-mail Address : ", tp[8])
476                     print("    Phone          : ", tp[9])
```

```
                print("    Order Date     : ", tp[12])
                print("    Username       : ", tp[13])
                print()
        elif choice == 3:
            cursor.execute("SELECT * FROM orders WHERE order_type = 'replace' ORDER BY ord_no DESC")
            data1 = cursor.fetchall()
            if not data1:
                print("No orders.")
            else:
                for row in data1:
                    tp = row
                    print("    Order ID       : ", tp[0])
                    print("    Book Name      : ", capwords(tp[11]))
                    print("    Pieces         : ", tp[10])
                    print("    Customer Name  : ", capwords(tp[1]))
                    print("    Address        : ", tp[2])
                    print("    District       : ", tp[3])
                    print("    City           : ", tp[4])
                    print("    State          : ", tp[5])
                    print("    Pincode        : ", tp[6])
                    print("    Country        : ", tp[7])
                    print("    E-mail Address : ", tp[8])
                    print("    Phone          : ", tp[9])
                    print("    Order Date     : ", tp[12])
                    print("    Username       : ", tp[13])
                    print()
        elif choice == 4:
            cursor.execute("SELECT * FROM orders ORDER BY ord_no DESC")
            data = cursor.fetchall()
            if not data:
                print("No orders.")
            else:
```

```
            for row in data:
                tp = row
                print("    Order ID       : ", tp[0])
                print("    Book Name      : ", capwords(tp[11]))
                print("    Pieces         : ", tp[10])
                print("    Customer Name  : ", capwords(tp[1]))
                print("    Address        : ", tp[2])
                print("    District       : ", tp[3])
                print("    City           : ", tp[4])
                print("    State          : ", tp[5])
                print("    Pincode        : ", tp[6])
                print("    Country        : ", tp[7])
                print("    E-mail Address : ", tp[8])
                print("    Phone          : ", tp[9])
                print("    Order Date     : ", tp[12])
                print("    Username       : ", tp[13])
                print("    Order Type     : ", tp[14])
                if tp[14] not in ['return', 'replace']:
                    cursor.execute(
                        f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
                    data2 = cursor.fetchone()
                    if not data2:
                        cursor.execute(
                            f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
                        data3 = cursor.fetchone()
                        print(f"    Order Status   :  Delivered on {data3[0]}")
                    else:
                        print("    Order Status   :  In transit")
                print()
        else:
            ch2 = int(input("Wrong Input. Enter Again: "))
```

```
            ch2 = int(input("Wrong Input. Enter Again: "))
            start(ch2)

    start(ch)
    cam.close()
    return

# ---------------------------------------------------------------------
'''This function will allow administrator to view all orders of a user'''

def View_all_u():  # Admin can view any user's order particularly
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SELECT ord_no FROM orders")
    data = cursor.fetchall()
    if not data:
        print("No orders in database.")
    else:
        user = str(input("Enter the customer's username: "))
        cursor.execute(f"SELECT * FROM orders WHERE user_name = '{user}' ORDER BY order_dt DESC")
        data2 = cursor.fetchall()
        if data2 == None:
            print(f"No orders of username '{user}'.")
        else:
            for row in data2:
                tp = row
                print("    Order ID       : ", tp[0])
                print("    Book Name      : ", capwords(tp[11]))
                print("    Pieces         : ", tp[10])
                print("    Customer Name  : ", capwords(tp[1]))
                print("    Address        : ", tp[2])
```

```
                print("    Address        : ", tp[2])
                print("    District       : ", tp[3])
                print("    City           : ", tp[4])
                print("    State          : ", tp[5])
                print("    Pincode        : ", tp[6])
                print("    Country        : ", tp[7])
                print("    E-mail Address : ", tp[8])
                print("    Phone          : ", tp[9])
                print("    Order Date     : ", tp[12])
                print("    Order Type     : ", capwords(tp[14]))
                if tp[14] not in ['return', 'replace']:
                    cursor.execute(
                        f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
                    data2 = cursor.fetchone()
                    if not data2:
                        cursor.execute(
                            f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
                        data3 = cursor.fetchone()
                        print(f"    Order Status   :  Delivered on {data3[0]}")
                    else:
                        print("    Order Status   :  In transit")
                print()
    cam.close()
    return

# ---------------------------------------------------------------------
'''This function will allow administrator to view all orders of a particular date'''

def View_ord_dt():  # Admin can view any user's order by date
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SELECT ord_no FROM orders")
```

```
601     data = cursor.fetchall()
602     if not data:
603         print("No orders in database.")
604     else:
605         dt = str(input("Enter date (YYYY-MM-DD): "))
606         cursor.execute(f"SELECT * FROM orders WHERE DATE(order_dt) = '{dt}'")
607         data = cursor.fetchall()
608         for row in data:
609             tp = row
610             print("    Order ID       : ", tp[0])
611             print("    Book Name      : ", capwords(tp[11]))
612             print("    Pieces         : ", tp[10])
613             print("    Customer Name  : ", capwords(tp[1]))
614             print("    Address        : ", tp[2])
615             print("    District       : ", tp[3])
616             print("    City           : ", tp[4])
617             print("    State          : ", tp[5])
618             print("    Pincode        : ", tp[6])
619             print("    Country        : ", tp[7])
620             print("    E-mail Address : ", tp[8])
621             print("    Phone          : ", tp[9])
622             print("    Order Date     : ", tp[12])
623             print("    Username       : ", tp[13])
624             print("    Order Type     : ", capwords(tp[14]))
625             if tp[14] not in ['return', 'replace']:
626                 cursor.execute(f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
627                 data2 = cursor.fetchone()
628                 if not data2:
629                     cursor.execute(f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
630                     data3 = cursor.fetchone()
631                     print(f"    Order Status   :  Delivered on {data3[0]}")
632                 else:
```

```
                print("    Order Status    :  In transit")
            print()
        cam.close()
        return

# ------------------------------------------------------------------------------------------------------------
# |                                  Below Functions are only for non-admin users                             |
# ------------------------------------------------------------------------------------------------------------

# ---------------------------------------------------------------------
'''This function will allow users to cancel their order only if they cancel it within 7 days of the date since
   order is placed. An email will be sent to the user with the details.'''

def Cancel_ord_u(username):
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute(
        f"SELECT * from orders where user_name = '{username}' and DATEDIFF(NOW(), order_dt) <= 7 ORDER by ord_no DESC")
    data = cursor.fetchall()
    list1 = []
    if not data:
        print("None of your orders are eligible to cancel. Sorry :)")
    else:
        print("This is the list of your orders which are eligible to cancel: ")
        for row in data:
            tp = row
            list1.append(tp[0])
            print("    Order ID       : ", tp[0])
            print("    Book Name      : ", capwords(tp[11]))
            print("    Pieces         : ", tp[10])
            print("    Customer Name  : ", capwords(tp[1]))
```

```
663             print("    Customer Name  : ", capwords(tp[1]))
664             print("    Address        : ", tp[2])
665             print("    District       : ", tp[3])
666             print("    City           : ", tp[4])
667             print("    State          : ", tp[5])
668             print("    Pincode        : ", tp[6])
669             print("    Country        : ", tp[7])
670             print("    E-mail Address : ", tp[8])
671             print("    Phone          : ", tp[9])
672             print("    Order Date     : ", tp[12])
673             print("    Order Type     : ", tp[14])
674
675             if tp[14] not in ['return', 'replace']:
676                 cursor.execute(f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
677                 data2 = cursor.fetchone()
678                 if not data2:
679                     cursor.execute(f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
680                     data3 = cursor.fetchone()
681                     print(f"    Order Status   :  Delivered on {data3[0]}")
682                 else:
683                     print("    Order Status   :  In transit")
684             print()
685
686         def choose():
687             choice = int(input("Enter the Order ID to cancel: "))
688             return choice
689
690         ch = choose()
691         if ch in list1:
692             cursor.execute(f"SELECT * FROM orders WHERE ord_no = {ch}")
693             data2 = cursor.fetchone()
694             cursor.execute(f"SELECT publication, author_fname, author_lname FROM books WHERE title = '{data2[11]}'")
```

```
        data3 = cursor.fetchone()
        msg = f"Your Order has been cancelled as per your request. Please review the product on our website. \n\n" \
              f"Book Name: {capwords(data2[11])} \n" \
              f"Publication: {capwords(data3[0])} \n" \
              f"Author's Name: {capwords(data3[1])} {capwords(data3[2])} \n" \
              f"Order ID: {data2[0]} \n" \
              f"Name: {capwords(data2[1])} \n" \
              f"Address: {data2[2]} \n" \
              f"District: {data2[3]} \n" \
              f"City: {data2[4]} \n" \
              f"State: {data2[5]} \n" \
              f"Pincode: {data2[6]} \n" \
              f"Country: {data2[7]} \n" \
              f"Phone: {data2[9]} \n" \
              f"Pieces Ordered: {data2[10]} \n" \
              f"Order Date & Time: {data2[12]} \n" \
              f"Order Status: Cancelled on {datetime.datetime.now()}"
        email = data2[8]
        sub = 'Regarding Cancellation of Order on S & T Bookshop'
        Send_email(email, sub, msg)
        cursor.execute("SET SQL_SAFE_UPDATES = 0")
        cursor.execute(
            f"UPDATE books SET stock_quantity = stock_quantity + {data2[10]} WHERE title = '{data2[11]}'")
        cam.commit()
        cursor.execute(f"DELETE FROM orders where ord_no = {ch}")
        cam.commit()
        print("Successfully Cancelled the Order.")
    if len(list1) > 1:
        ch2 = str(input("   Want to cancel more Orders? (y OR n) "))
        if ch2 == 'y' or ch2 == 'Y':
            Cancel_ord_u(username)
```

```
                Cancel_ord_u(username)
            elif ch2 == 'n' or ch2 == 'N':
                cam.close()
                return
            else:
                print("Wrong Input")
                return
        else:
            pass

# ---------------------------------------------------------------------
'''This function will allow administrator to update his/her order address if they do it within 7 days after
   order is being placed. An email will be sent with few order details and new location of delivery.'''

def Up_ord_au(username):
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute(f"SELECT * FROM orders WHERE user_name = '{username}' and DATEDIFF(NOW(), order_dt) <= 7 ORDER BY ord_no DESC")
    data = cursor.fetchall()
    list1 = []
    if data == None:
        print("No 'IN TRANSIT' orders in database for your username.")
    else:
        print("This is the list of your orders eligible for updating the address: ")
        for row in data:
            tp = row
            list1.append(tp[0])
            print("    Order ID:       ", tp[0])
            print("    Book Name:      ", capwords(tp[11]))
            print("    Pieces:         ", tp[10])
            print("    Customer Name:  ", capwords(tp[1]))
            print("    Address:        ", tp[2])
```

```
757         print("    District:       ", tp[3])
758         print("    City:           ", tp[4])
759         print("    State/UT:       ", tp[5])
760         print("    Pincode:        ", tp[6])
761         print("    Country:        ", tp[7])
762         print("    E-mail Address: ", tp[8])
763         print("    Phone:          ", tp[9])
764         print("    Order Date:     ", tp[12])
765         print()
766
767     def choose():
768         choice = int(input("Enter the Order ID to update address: "))
769         return choice
770     ch = choose()
771     if ch in list1:
772         new_addr = str(input("Enter the new address: "))
773         new_city = str(input("Enter City: "))
774         new_pinc = int(input('Enter pincode: '))
775         new_dist = str(input("Enter District: "))
776         new_stat = str(input("Enter State/UT" + ': '))
777         cursor.execute("SET SQL_SAFE_UPDATES = 0")
778         cursor.execute(f"UPDATE orders SET address = '{new_addr}' WHERE ord_no = {ch}")
779         cam.commit()
780         cursor.execute(f"UPDATE orders SET city = '{new_city}' WHERE ord_no = '{ch}'")
781         cam.commit()
782         cursor.execute(f"UPDATE orders SET pincode = {new_pinc}  WHERE ord_no = {ch}")
783         cam.commit()
784         cursor.execute(f"UPDATE orders SET district = '{new_dist}' WHERE ord_no = '{ch}'")
785         cam.commit()
786         cursor.execute(f"UPDATE orders SET state_ut = '{new_stat}' WHERE ord_no = '{ch}'")
787         cam.commit()
788         print('Successfully Updated')
```

```
            cursor.execute(f"SELECT email FROM accounts WHERE username = '{username}'")
            email = cursor.fetchone()
            msg = f"Your Order has been updated successfully. \n" \
                  f"New Location of the order is {new_addr}, {new_city}, {new_pinc}, {new_city}, {new_stat} :)"
            sub = "Information of Order Update on S & T Book Shop Management System"
            Send_email(email[0], sub, msg)
            print()

        if len(list1) > 1:
            ch2 = str(input("   Want to update more Orders? (y OR n): "))
            if ch2 == 'y' or ch2 == 'Y':
                Up_ord_au(username)
            elif ch2 == 'n' or ch2 == 'N':
                return
            else:
                print("Wrong Input")
                return
        else:
            pass
    cam.close()
    return

#  ---------------------------------------------------------------------
'''This function will allow the signed in user to view all their details of every order placed since
   registered with detail of order type (i.e, new or for replace or for return)'''

def View_au(username):
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    print("This is the list of your orders.: ")
    cursor.execute(f"SELECT * FROM orders WHERE user_name = '{username}' ORDER BY ord_no DESC")
    data = cursor.fetchall()
```

orders.py

```
821     if data == None:
822         print("No orders in database.")
823     else:
824         print("                  Latest Orders Sorted by Date:. ")
825         for row in data:
826             tp = row
827             print("    Order ID       : ", tp[0])
828             print("    Book Name      : ", capwords(tp[11]))
829             print("    Pieces         : ", tp[10])
830             print("    Customer Name  : ", capwords(tp[1]))
831             print("    Address        : ", tp[2])
832             print("    District       : ", tp[3])
833             print("    City           : ", tp[4])
834             print("    State/UT       : ", tp[5])
835             print("    Pincode        : ", tp[6])
836             print("    Country        : ", tp[7])
837             print("    E-mail Address : ", tp[8])
838             print("    Phone          : ", tp[9])
839             print("    Order Date     : ", tp[12])
840             print("    Order Type     : ", capwords(tp[14]))
841
842             if tp[14] not in ['return', 'replace']:
843                 cursor.execute(f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
844                 data2 = cursor.fetchone()
845                 if not data2:
846                     cursor.execute(f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
847                     data3 = cursor.fetchone()
848                     print(f"    Order Status   :  Delivered on {data3[0]}")
849                 else:
850                     print("    Order Status   :  In transit")
851             print()
```

```
852         cam.close()
853         return
854
855     # ------------------------------------------------------------------------------------------
856     # |                                    End of Module                                       |
857     # # ------------------------------------------------------------------------------------------
858
```