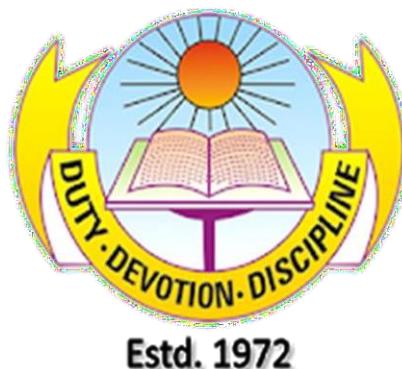


# **SUNBEAM SCHOOL VARUNA**

## **VARANASI**



**Computer Science (New) (083)**

**Project On**

**Python and MySQL**

**(Book Shop Management System)**

**Session 2020-21**

**Under the guidance of: -**

**Ms. Ruhi Seth**

**Made by:-**

**Swarit Jain**

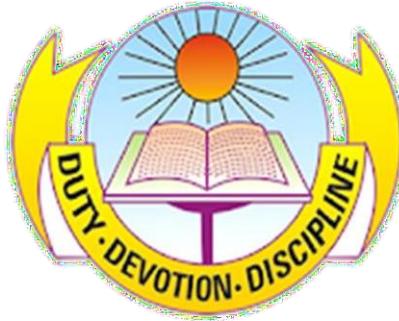
**(XII-A, )**

**Tanishq Singh**

**(XII-C, )**

# Acknowledgement

I would like to convey my thanks to ***Ms. Ruhi Seth***, my ***COMPUTER SCIENCE*** teacher, Sunbeam School, Varuna for her immense help and guidance in the completion of my project. It is only due to her effort that my project completed successfully. This report is submitted as a part of practical examination included in curriculum of CBSE for ***AISSE*** (All India Senior Secondary Examination) to be held in the year 2020-21.



Estd. 1972

# SUNBEAM SCHOOL VARUNA

## Certificate

This is to certify that **Swarit Jain (Class 12-A)** of Sunbeam School Varuna, Varanasi has completed this project under my supervision. He took much interest and shown utmost sincerity in completion of this project. He has successfully completed the project work in ‘Computer Science (083)’ up to my satisfaction.

---

(P.G.T. Computer Science)

# Index

S.No.	Content	Pg.No.
1.	About Python	5-6
2.	About MySQL	7-8

## Source Code

S.No.	Content	Pg.No.
3.	MySQL Data with test entries (.sql file)	9-13
4.	books.py (handles books)	14-20
5.	orders.py (handles orders for books)	21-50
6.	acc_ctrl.py (handles accounts)	51-59
7.	config.py (generates invoice number and invoice PDF)	60-66
8.	main_menu.py (Main Menu of Program)	67-73

## Outputs

S.No.	Content	Pg.No.
9.	Outputs of a Non-Administrator User	74-86
10.	Outputs of an Administrator User	87-108
11.	Outputs of Handled Errors	109-112
12.	Bibliography	113

# What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- Web development (Server-side),
- Software development,
- Mathematics,
- System scripting

## I) What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

## II) Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

- Python can be treated in a procedural way, an object-oriented way or a functional way.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

### III) Good to know

- The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- Python comes with a pre-built text editor. It is also possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

### IV) Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

# **What is MySQL ?**

- MySQL is a database system used for developing web-based software applications.
- MySQL used for both small and large applications.
- MySQL is a relational database management system (*RDBMS*).
- MySQL is fast, reliable, and flexible and easy to use.
- MySQL supports standard SQL (*Structured Query Language*).
- MySQL is free to download and use.
- MySQL was developed by Michael Widenius and David Axmark in 1994.
- MySQL is presently developed, distributed, and supported by Oracle Corporation.
- MySQL Written in C, C++.

## **Main Features of MySQL**

- MySQL server design is multi-layered with independent modules.
- MySQL is fully multithreaded by using kernel threads. It can handle multiple CPUs if they are available.
- MySQL provides transactional and non-transactional storage engines.
- MySQL has a high-speed thread-based memory allocation system.

- MySQL Handles large databases.
- MySQL Server works in client/server or embedded systems.
- MySQL Works on many different platforms.
- MySQL supports in memory heap table

## Who Uses MySQL

- Some of the most famous websites like Facebook, Wikipedia, Google (not for search), YouTube, Flickr.
- Content Management Systems (CMS) like WordPress, Drupal, Joomla, phpBB etc.
- A large number of web developers worldwide are using MySQL to develop web applications.

# **Complete MySQL Data with Test Entries**



main\* ×



```
1 • DROP DATABASE IF EXISTS book_shop;
2 • CREATE DATABASE book_shop;
3 • USE book_shop;
4
5 -- -----Books Table -----
6
7 • CREATE TABLE books
8 (
9     book_id INT PRIMARY KEY AUTO_INCREMENT,
10    title VARCHAR(100) NOT NULL,
11   author_fname VARCHAR(100) NOT NULL,
12  author_lname VARCHAR(100) NOT NULL,
13 released_year INT NOT NULL,
14 category VARCHAR(40) NOT NULL,
15 publication VARCHAR(100) NOT NULL,
16 stock_quantity INT NOT NULL,
17 pages INT NOT NULL,
18 pg_word INT,
19 price INT NOT NULL
20 );
21
22 --
23
24 • INSERT INTO books (title, author_fname, author_lname, released_year, stock_quantity, pages, price, pg_word, publication, category)
25 VALUES
26 ('the namesake', 'jhumpa', 'lahiri', 2007, 32, 304, 299, 150, 'Harpercollins (5 March 2007)', 'Classic'),
27 ('norse mythology', 'neil', 'gaiman', 2019, 43, 304, 341, 200, 'Bloomsbury Publishing (18 December 2019)', 'Mythology'),
28 ('american gods', 'neil', 'gaiman', 2002, 12, 736, 364, 145, 'Headline Review (4 March 2002)', 'Drama'),
29 ('interpreter of maladies', 'jhumpa', 'lahiri', 2005, 97, 198, 259, 125, 'Harpercollins Publishers India (5 September 2005)', 'Fantasy'),|
```



```
main* X
[File] [New] [Open] [Save] [Print] [Exit] [Help] [Don't Limit]
59     pieces INT NOT NULL,      -- 10
60     bookname VARCHAR(100) NOT NULL,    -- 11
61     order_dt TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- 12
62     user_name VARCHAR(30) NOT NULL, -- 13
63     order_type CHAR(7) NOT NULL CHECK(order_type in ('replace', 'new', 'return')), -- 14
64     FOREIGN KEY (ord_no) REFERENCES books(book_id)
65   );
66
67 • ALTER TABLE orders engine = InnoDB;
68
69 • INSERT INTO orders (ord_no, customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, order_dt, user_name, order_type)
70   VALUES
71   ('Swarit', 'C-121, SBI Colony, Main Branch Campus, Opposite Kachahari', 'Varanasi', 'Varanasi', 'Uttar Pradesh', 221002, 'India',
72   'swaritjain123@gmail.com', 8004124465, 5, 'american gods', '2020-10-26 15:30:11', 'admin', 'new'),
73   ('Swarit', 'C-121, SBI Colony, Main Branch Campus, Opposite Kachahari', 'Varanasi', 'Varanasi', 'Uttar Pradesh', 221002, 'India',
74   'swaritjain123@gmail.com', 8004124465, 1, 'coraline', '2020-11-01 15:30:11', 'admin', 'new'),
75   (5, 'Swarit', 'C-121, SBI Colony, Main Branch Campus, Opposite Kachahari', 'Varanasi', 'Varanasi', 'Uttar Pradesh', 221002, 'India',
76   'swaritjain123@gmail.com', 8004124465, 2, 'the namesake', '2020-11-02 15:30:11', 'admin', 'return');
77
78   -- Orders inserted above are being used in the program to show outputs of replace and return functions with the help of date difference b/w 7 to 14 days
79
80   ----- Accounts Table -----
81 • CREATE TABLE accounts
82   (
83     ID BIGINT PRIMARY KEY AUTO_INCREMENT,
84     username VARCHAR(30) UNIQUE NOT NULL,
85     passwd VARCHAR(40) NOT NULL,
86     name_u VARCHAR(50) NOT NULL,
87     email VARCHAR(50) NOT NULL CHECK(email LIKE '%@%') UNIQUE
88   );
```

```
91      ----- Test Accounts -----
92
93 • INSERT INTO accounts (username, passwd, name_u, email)
94   VALUES
95   ('admin', 'admin', 'admin', 'swaritjain123@gmail.com'),
96   ('testuser', 'testuser', 'test', 'tanishq.rc@gmail.com');
97
98 • ALTER TABLE accounts engine = InnoDB;
99 -----
```

**books.py**

## Class 12 Computer Project &gt; 📚 books.py &gt;

1: Project

books.py ×

```
1 import mysql.connector as conn1
2 import string
3 #
4 ''' This function will help the administrator to add as many books they want in the 'books' table '''
5
6 def Add_book():
7     mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
8     cursor = mycon1.cursor(buffered=True)
9     no_books = int(input("Enter number of books you want to add: "))
10    for i in range(no_books):
11        try:
12            book_name = str(input("      Enter the book name: ")).lower()
13            writer_fname = str(input("      Enter the first name of author: ")).lower()
14            writer_lname = str(input("      Enter the last name of author: ")).lower()
15            year = int(input("      Enter the year of Release: "))
16            cat = str(input("      Enter the category of book: ")).lower()
17            pub = str(input("      Enter the Publisher's Name: ")).lower()
18            stock = int(input("      Enter the number of books in stock: "))
19            page = int(input("      Enter number of pages of book: "))
20            words = int(input("      Enter the number of words per page (average) approximate: "))
21            price = int(input("      Enter the price of the book: "))
22            cursor.execute(
23                f"INSERT INTO books (title, author_fname, author_lname, released_year, stock_quantity, category, "
24                f"publication, pages, pg_word, price) VALUES('{book_name}', '{writer_fname}', '{writer_lname}', {year}, "
25                f"{stock}, '{cat}', '{pub}', {page}, {words}, {price})")
26            mycon1.commit()
27            print("Successfully Added the book.")
28        except:
29            print("Wrong Data Given.")
30            Add_book()
31
32        mycon1.close()
33
34    return
```

2: Structure

3: Favorites

4: AWS Explorer

```
34  #  
35  ''' This function will help the administrator to update books' stock quantity,  
36  price, number of pages, release year '''  
37  
38 def Update_book():  
39     mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")  
40     cursor = mycon1.cursor(buffered=True)  
41     no_books = int(input("Enter number of books you want to update: "))  
42     count = 0  
43     while count < no_books:  
44         book_name = str(input("Enter the name of the book you want to update: ")).lower()  
45         cursor.execute("SELECT * FROM books WHERE title = '{}'".format(book_name))  
46         data = cursor.fetchone()  
47         if data == None:  
48             print("No book like '{}' exists in database.".format(book_name)) # Checking if the book exists in database  
49             Update_book() # If not found, function runs again.  
50         else:  
51             print(" 1. Update the book's year: \n",  
52                 " 2. Update the book's stock: \n",  
53                 " 3. Update the book's number of pages: \n",  
54                 " 4. Update the book's price: ")  
55         choice = int(input("Enter your choice: "))  
56         if choice == 1:  
57             year = int(input("Enter the new year of the book: "))  
58             cursor.execute("UPDATE books SET released_year = {} where title = '{}'".format(year, book_name))  
59             mycon1.commit() # Updating the book's release year  
60             print("Successfully Updated")  
61         elif choice == 2:  
62             stock = int(input("Enter the stock of the book: "))  
63             cursor.execute("UPDATE books SET stock_quantity = {} where title = '{}'".format(stock, book_name))  
64             mycon1.commit() # Updating the book's stock quantity  
65             print("Successfully Updated")
```

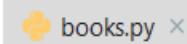
 books.py x

```
66     elif choice == 3:
67         page = int(input("Enter the number of pages of the book: "))
68         cursor.execute("UPDATE books SET pages = {} where title = '{}'".format(page, book_name))
69         mycon1.commit() # Updating the book's number of pages as per new release
70         print("Successfully Updated")
71     elif choice == 4:
72         price_n = int(input("Enter the new price: "))
73         cursor.execute("UPDATE books SET price = {} where title = '{}'".format(price_n, book_name))
74         mycon1.commit() # Updating the book's price
75         print("Successfully Updated")
76     else:
77         print("Wrong Input")
78         Update_book()
79     count += 1
80 mycon1.close()
81 return

83 #
84 ''' This function will help the administrator to delete books from the books table'''

86 def Delete_book():
87     mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
88     cursor = mycon1.cursor(buffered=True)
89     no_books = int(input("Enter number of books you want to delete: "))
90     count = 0
91     while count < no_books:
92         book_name = str(input("Enter the name of the book you want to delete: ")).lower()
93         cursor.execute("SELECT * FROM books WHERE title = '{}'".format(book_name))
94         data = cursor.fetchone()
95         if data == None:
96             print("No book like '{}' exists in database.".format(book_name)) # Checking if the book exists in database
```

1: Project



```
97     else:
98         try:
99             cursor.execute("DELETE FROM books where title = '{}'".format(book_name))
100            mycon1.commit() # Deleting the book
101            print("Successfully Deleted.")
102        except:
103            print("This book can't be deleted because Orders for this book exists.")
104        count += 1
105    mycon1.close()
106    return
107
108 #
109 """ This function will help administrator to view every detail of as many books admin wants."""
110
111 def View_details():
112     mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
113     cursor = mycon1.cursor(buffered=True)
114     no_books = int(input("Enter number of books of which you want details of?: "))
115     count = 0
116     while count < no_books:
117         book_name = str(input("Enter the name of the book " + str(count + 1) + ": ")).lower()
118         cursor.execute("SELECT * FROM books WHERE title = '{}'".format(book_name))
119         data = cursor.fetchone()
120         if data == None:
121             choice = str(input(
122                 f"No book like '{book_name}' exists in database. Renter Again? (Y or N): ")) # Checking the Wrong Input
123             if choice == 'Y' or 'y':
124                 View_details()
125             else:
126                 exit(0)
127         else:
128             print("    Book ID: ", data[0])
```

2: Structure

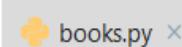


2: Favorites



AWS Explorer





```
127     else:
128         print("    Book ID:           ", data[0])
129         print("    Book Name:        ", string.capwords(data[1]))
130         print("    Author's Name:    ", string.capwords(data[2]) + ' ' + string.capwords(data[3]))
131         print("    Release Year:    ", data[4])
132         print("    Category:         ", string.capwords(data[5]))
133         print("    Publication:      ", string.capwords(data[6]))
134         print("    Pages:             ", data[8])
135         print("    Words Per Page (approx): ", data[9])
136         print("    Stock Quantity:   ", data[7])
137         print("    Price:             ", data[10])
138     count += 1
139 mycon1.close()
140 return
141 #
143 '''This function will help administrator to view details of every book in the shop'''
144
145 def View_all():
146     mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
147     cursor = mycon1.cursor(buffered=True)
148     cursor.execute("SELECT * FROM books")
149     data = cursor.fetchall()
150     for row in data:
151         tp = row # fetchall return tuple of tuples. Therefore, taking one tuple at a time in 'tp' through for loop.
152         print("    Book ID:           ", tp[0])
153         print("    Book Name:        ", string.capwords(tp[1]))
154         print("    Author's Name:    ", string.capwords(tp[2]) + ' ' + string.capwords(tp[3]))
155         print("    Release Year:    ", tp[4])
156         print("    Category:         ", string.capwords(tp[5]))
157         print("    Publication:      ", string.capwords(tp[6]))
158         print("    Pages:             ", tp[8])
```

```
159     print(" Words Per Page (approx): ", tp[9])
160     print(" Stock Quantity:      ", tp[7])
161     print(" Price:                 ", tp[10])
162     print()
163     mycon1.close()
164     return
165
166 # -----
167 # /
```

*End of Module*

/

**orders.py**

1: Project



orders.py X

```
1 # Inbuilt Modules
2 import smtplib
3 from string import capwords
4 from time import sleep
5 import datetime
6
7 # Project Modules
8 import mysql.connector as mycam
9 import config
10
11
12 # -----
13 # |          Below Functions are Common          |
14 # -----
15
16 # -----
17 ''' This function will allow every user in database to place order for any book if it exists in database.
18 User is prompted every time to place another order if he/she wants.
19 When user says 'n' or 'no' function are called from other modules to generate invoice and invoice number.'''
20
21 def Place_ord(username):
22     ord_id_ls = []
23     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
24     cursor = cam.cursor(buffered=True)
25     cursor.execute("SET SQL_SAFE_UPDATES = 0")
26
27     def work():
28         print()
29         book_name = str(input('Enter the name of the book ' + ': ')).lower()
30         cursor.execute("SELECT * FROM books WHERE title='{}'".format(book_name))
31         data = cursor.fetchone()
32         if data == None:
```

2: Structure



2: Favorites



AWS Explorer



1: Project



orders.py ×

```
31     data = cursor.fetchone()
32
33     if data == None:
34         print('Sorry, No book named', book_name, 'found.')
35     else:
36         print(f" Book named , {capwords(book_name)}, found details are as follows:-")
37         print(" Author's name:      ", capwords(data[2]) + ' ' + capwords(data[3]))
38         print(" Released Year:      ", data[4])
39         print(" Number of Pages:    ", data[8])
40         print(" Words Per Page (Approx): ", data[9])
41         print(" Publication:        ", capwords(data[6]))
42         print(" Category of Book:   ", capwords(data[5]))
43         print(" Available quantity: ", data[7])
44         print(" Price of the book:   ", data[10])
45
46         choice = str(input("Do you want to order this Book ? "))
47
48         if choice in ['yes', 'Yes', 'YES', 'y', 'Y']:
49             quantity = int(input('Enter the QUANTITY you want to order: '))
50             print('          Checking Stock, please wait...')
51             sleep(2)
52
53             if quantity <= data[7]: # Checking if pieces specified to be bought are present or not for the specified book.
54                 print('          Enough quantity found')
55                 print('Please fill up your details below: ')
56
57                 name = str(input('          Enter Name: '))
58                 address = str(input('          Enter Address: '))
59                 dist = str(input('          Enter District: '))
60                 city = str(input('          Enter City: '))
61                 stu = str(input('          Enter State/UT: '))
62                 pin = int(input("          Enter Pincode: "))
63                 country = str(input('          Enter Country: '))
64                 phone = int(input('          Enter Phone number: '))
```

2: Structure



Favorites



AWS Explorer



```
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.")
```

1: Project

orders.py x

```
93         print()
94         print("          Order Placed Successfully.")
95     else:
96         print('Your order has been discarded')
97
98     else:
99         print('Sorry', data[7], 'pieces are left')
100        print('Thanks for visiting, keep shopping :)')
101    elif choice in ['no', 'No', 'NO', 'n', 'N']:
102        print('Okay.')
103    else:
104        print('Invalid Input')
105
106    print()
107    ch = str(input("Want to place more orders? (y/n): ")).lower()
108    if ch in ['y', 'yes']:
109        work()
110    else:
111        details.append(email[0])
112        details.append(username)
113        inv = config.Inv_no_gen(username)
114        config.Pdf_generate(details, ord_id_ls, inv[0], inv[1])
115        print("Thanks for visiting us :)")
116        print()
117    work()
118    return
119
120    # -----
121    '''This function will be used only if the order is specified for return/replace or order is cancelled
122    For that, message and subject or email is defined in their respective functions.
123    Sender's Email Address and Password is given in config module of the project
124    An email will be sent to the email address of the user with the details.'''

```

2: Structure

3: Favorites

4: Star

5: AWS Explorer

Project

Structure

Favorites

AWS Explorer

orders.py x

```
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")
```

1: Project

orders.py ×

```
157     data = cursor.fetchall()
158     if not data:
159         print('None of your order is eligible to return or replace.')
160     else:
161         print('We have found', len(data), 'orders have been delivered to you')
162         print('List of the orders that have been delivered to you are as follows:-')
163         for i in data:
164             L.append(i[0])
165             print(' Book Name      : ', i[11])
166             print(' Order Date     : ', i[12])
167             print(' Order ID       : ', i[0])
168             print(' Customer Name  : ', i[1])
169             print(f" Location       : {i[2]}, {i[3]}, {i[4]}, {i[5]}, {i[6]}, {i[7]}")
170             print(' Email          : ', i[8])
171             print(' Phone no       : ', i[9])
172             print(' No of Books    : ', i[10])
173             print()
174
175             choice = str(input('Do you want to return or replace any of the above orders (y/n) ')).lower()
176
177             def work(c):
178                 if c == 'y':
179                     ID = int(input('Enter the order ID for which you want to return or replace: '))
180                     if ID not in L:
181                         print('No order ID ', ID, ' matched from the orders that have been delivered to you')
182                     else:
183                         print('You have chosen order ID: ', ID)
184                         print('Few details of your order are as follows: ')
185                         print()
186                         cursor.execute(f"SELECT * FROM orders WHERE ord_no = {ID}")
187                         data2 = cursor.fetchone()
188                         print(' Book name: ', data2[11])
```

2: Structure

3: Favorites

4: AWS Explorer

```
165     print(' Book Name : ', i[11])
166     print(' Order Date : ', i[12])
167     print(' Order ID : ', i[0])
168     print(' Customer Name : ', i[1])
169     print(f" Location : {i[2]}, {i[3]}, {i[4]}, {i[5]}, {i[6]}, {i[7]}")
170     print(' Email : ', i[8])
171     print(' Phone no : ', i[9])
172     print(' No of Books : ', i[10])
173     print()
174
175 choice = str(input('Do you want to return or replace any of the above orders (y/n) ')).lower()
176
177 def work(c):
178     if c == 'y':
179         ID = int(input('Enter the order ID for which you want to return or replace: '))
180         if ID not in L:
181             print('No order ID ', ID, ' matched from the orders that have been delivered to you')
182         else:
183             print('You have chosen order ID: ', ID)
184             print('Few details of your order are as follows: ')
185             print()
186             cursor.execute(f"SELECT * FROM orders WHERE ord_no = {ID}")
187             data2 = cursor.fetchone()
188             print(' Book name: ', data2[11])
189             print(' Quantity: ', data2[10])
190             print(' Order date: ', data2[12])
191             print()
192             print('By SNT.bookshop Return and Replace policies, you have below two options')
193             print('1. Return')
194             print('2. Replace')
195
196             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()
```

```
229     print(' Successfully requested for return to your order.')
230     print(
231         ' Our courier boy will come within the next day to collect the package after verifying its condition,
232         ' be ready with the packed book')
233     print(' Money will be refund to your account within 2-3 days')
234     return
235 else:
236     print(' Return process cancelled')
237
238 elif opt == 2:
239     print('You have entered under our Replace policy.')
240     confirm = str(input('Are you sure you want to replace for this order? (y/n) ')).lower()
241
242 if confirm in ['y', 'yes']:
243     msg = f"Your request for replacing your order has been accepted. Some details for it are as follows: \n \
244         f'Book Name: {capwords(data2[11])} \n' \
245         f'Order ID: {data2[0]} \n' \
246         f'Name: {data2[1]} \n' \
247         f'Address: {data2[2]} \n' \
248         f'District: {data2[3]} \n' \
249         f'City: {data2[4]} \n' \
250         f'State: {data2[5]} \n' \
251         f'Pincode: {data2[6]} \n' \
252         f'Country: {data2[7]} \n' \
253         f'Phone: {data2[9]} \n' \
254         f'Pieces Ordered: {data2[10]} \n' \
255         f'Order Date & Time: {data2[12]} \n' \
256         f'Delivery will be provided within 7 working days. \n' \
257         f'Sorry for the inconvenience :)"
258
259     sub = f"Information of Replacing Order on S & T Bookshop"
260     print(" Successfully requested for replacement of your order.")
261     Send_email(data2[8], sub, msg) # Calling Function to send email
```

 orders.py x

```
260
261     send_email(data2[0], sub, msg) # Calling function to send email
262     # Real Email is sent here.
263     # Email is sent actually to the mailbox if the given email really exists
264     cursor.execute(
265         "INSERT INTO orders "
266         f"(customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, "
267         f"user_name, order_type) VALUES ('{data2[1]}', '{data2[2]}', '{data2[3]}', '{data2[4]}', '{data2[5]}', {data2[6]}, "
268         f"'{data2[7]}', '{data2[8]}', {data2[9]}, {data2[10]}, "
269         f"'{data2[11]}', '{username}', 'replace')")
270     cam.commit()
271     cursor.execute(f"DELETE FROM orders WHERE ord_no = {ID}")
272     cam.commit()
273     print(
274         ' Our courier boy will come within the next day to collect the package after verifying its condition, '
275         ' be ready with the packed book')
276     print(
277         " Replacement will be completed within 7 working days until your order has been delivered. ")
278     elif confirm in ['n', 'no']:
279         print("Replacement Cancelled")
280         print("Exiting")
281         return
282     else:
283         print('Invalid option chosen.')
284     elif c in ['n', 'no']:
285         print("Exiting")
286         return
287     else:
288         print("Wrong Input")
289         again_t = str(input("Do you want to return or replace any of the above orders? (y/n) ")).lower()
290         if again_t in ['yes', 'y']:
291             work(again_t)
```

```
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291     elif again_t in ['n', 'no']:
292         print("Exiting")
293         return
294     else:
295         print("Invalid Input.")
296
297     work(choice)
298     cam.close()
299     return
300
301
302 # -----
303 # |          Below Functions are for Administrator
304 # |
305 # -----
306 # |
307 '''This function will allow the administrator to cancel as many orders he wants by specifying the Order ID
308 An email will be sent to the user with details. Order can be cancelled only within 7 days after placing order'''
309
310 def Cancel_ord(): # Admin can cancel any user's order
311     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
312     cursor = cam.cursor(buffered=True)
313     cursor.execute("SELECT ord_no FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7")
314     rec = cursor.fetchall()
315     if len(rec) == 0:
316         print("No orders to cancel.")
317         print("Sorry")
318     else:
319         print(f"Only {len(rec)} order(s) is/are eligible to cancel.")
320         num = int(input(f"Enter number of orders you want to cancel (not more than {len(rec)}): "))
321         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]}"
```

1: Project



orders.py ×

2: Structure



2: Favorites



AWS Explorer

```
351             f"\"Pieses ordered: {data[10]} \n\" \n"
352             f"Order Date & Time: {data[12]}"
353             email = data[8]
354             sub = 'Regarding Cancellation of Order on S & T Bookshop'
355             Send_email(email, sub, msg)
356             cursor.execute(f"DELETE FROM orders WHERE ord_no = {order_id}")
357             cam.commit()
358             print("Successfully cancelled your order.")
359         else:
360             print("Order not eligible to cancel.")
361     else:
362         print(f"Not possible to cancel more than {len(rec)} orders.")
363         Cancel_ord()
364     cam.close()
365     return
366
367 # -----
368 '''This function will allow administrator to update the order address of multiple orders'''
369
370 def Update_ord_addr(): # Admin can update any user's order
371     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
372     cursor = cam.cursor(buffered=True)
373     cursor.execute("SELECT * FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7")
374     lim = cursor.fetchall()
375     if not lim:
376         print("No orders are eligible for update of address.")
377     else:
378         print(f"Only {len(lim)} order(s) is/are eligible for update of address.")
379
380     def work(data):
381         num = int(input('Enter the number of orders\'s addresses you want to update: '))
382         lst = [(data[i][0]) for i in range(len(data))]
```

Project 1: orders.py x

```
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                 msg += 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:
```

1: Project

orders.py x

```
413         continue
414     else:
415         print("Invalid Input. Try Again")
416         work(data)
417     work(lim)
418     cam.close()
419     return
420
421 # -----
422 '''This function will allow administrator to view all orders in database of every user.'''
423
424 def View_all_ords(): # Admin can view all user's order
425     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
426     cursor = cam.cursor(buffered=True)
427     print(" 1 -> View all New Orders Placed \n"
428          " 2 -> View all Orders for Return \n"
429          " 3 -> View all Orders for Replacement \n"
430          " 4 -> View all Orders including all categories")
431     ch = int(input("  Enter your choice (1 to 4): "))
432
433     def start(choice):
434         if choice == 1:
435             cursor.execute("SELECT * FROM orders WHERE order_type = 'new' ORDER BY ord_no DESC")
436             data1 = cursor.fetchall()
437             if not data1:
438                 print("No orders.")
439             else:
440                 for row in data1:
441                     tp = row
442                     print("    Order ID      : ", tp[0])
443                     print("    Book Name    : ", capwords(tp[11]))
444                     print("    Pieces       : ", tp[10])
```

2: Structure

3: Favorites

4: AWS Explorer

```
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])
```

```
477     print("    Order Date      : ", tp[12])
478     print("    Username       : ", tp[13])
479     print()
480 elif choice == 3:
481     cursor.execute("SELECT * FROM orders WHERE order_type = 'replace' ORDER BY ord_no DESC")
482     data1 = cursor.fetchall()
483     if not data1:
484         print("No orders.")
485     else:
486         for row in data1:
487             tp = row
488             print("    Order ID      : ", tp[0])
489             print("    Book Name     : ", capwords(tp[11]))
490             print("    Pieces        : ", tp[10])
491             print("    Customer Name : ", capwords(tp[1]))
492             print("    Address       : ", tp[2])
493             print("    District      : ", tp[3])
494             print("    City          : ", tp[4])
495             print("    State         : ", tp[5])
496             print("    Pincode       : ", tp[6])
497             print("    Country       : ", tp[7])
498             print("    E-mail Address: ", tp[8])
499             print("    Phone         : ", tp[9])
500             print("    Order Date    : ", tp[12])
501             print("    Username      : ", tp[13])
502             print()
503 elif choice == 4:
504     cursor.execute("SELECT * FROM orders ORDER BY ord_no DESC")
505     data = cursor.fetchall()
506     if not data:
507         print("No orders.")
508     else:
```

1: Project

orders.py x

2: Structure

2: Favorites

WS Explorer

```
509         for row in data:
510             tp = row
511             print("    Order ID      : ", tp[0])
512             print("    Book Name    : ", capwords(tp[11]))
513             print("    Pieces       : ", tp[10])
514             print("    Customer Name: ", capwords(tp[1]))
515             print("    Address      : ", tp[2])
516             print("    District     : ", tp[3])
517             print("    City          : ", tp[4])
518             print("    State         : ", tp[5])
519             print("    Pincode      : ", tp[6])
520             print("    Country       : ", tp[7])
521             print("    E-mail Address: ", tp[8])
522             print("    Phone         : ", tp[9])
523             print("    Order Date   : ", tp[12])
524             print("    Username      : ", tp[13])
525             print("    Order Type   : ", tp[14])
526             if tp[14] not in ['return', 'replace']:
527                 cursor.execute(
528                     f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
529                 data2 = cursor.fetchone()
530                 if not data2:
531                     cursor.execute(
532                         f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
533                     data3 = cursor.fetchone()
534                     print(f"    Order Status  : Delivered on {data3[0]}")
535                 else:
536                     print("    Order Status  : In transit")
537             print()
538         else:
539             ch2 = int(input("Wrong Input. Enter Again: "))
```



```
538
539     ch2 = int(input("Wrong Input. Enter Again: "))
540     start(ch2)
541
542     start(ch)
543     cam.close()
544     return
545
546 # -----
547 '''This function will allow administrator to view all orders of a user'''
548
549 def View_all_u(): # Admin can view any user's order particularly
550     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
551     cursor = cam.cursor(buffered=True)
552     cursor.execute("SELECT ord_no FROM orders")
553     data = cursor.fetchall()
554     if not data:
555         print("No orders in database.")
556     else:
557         user = str(input("Enter the customer's username: "))
558         cursor.execute(f"SELECT * FROM orders WHERE user_name = '{user}' ORDER BY order_dt DESC")
559         data2 = cursor.fetchall()
560         if data2 == None:
561             print(f"No orders of username '{user}' .")
562         else:
563             for row in data2:
564                 tp = row
565                 print("    Order ID      : ", tp[0])
566                 print("    Book Name     : ", capwords(tp[11]))
567                 print("    Pieces        : ", tp[10])
568                 print("    Customer Name : ", capwords(tp[1]))
569                 print("    Address       : ", tp[2])
```

```
569     print("    Address      : ", tp[2])
570     print("    District     : ", tp[3])
571     print("    City         : ", tp[4])
572     print("    State        : ", tp[5])
573     print("    Pincode      : ", tp[6])
574     print("    Country       : ", tp[7])
575     print("    E-mail Address : ", tp[8])
576     print("    Phone         : ", tp[9])
577     print("    Order Date    : ", tp[12])
578     print("    Order Type     : ", capwords(tp[14]))
579     if tp[14] not in ['return', 'replace']:
580         cursor.execute(
581             f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
582         data2 = cursor.fetchone()
583         if not data2:
584             cursor.execute(
585                 f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
586             data3 = cursor.fetchone()
587             print(f"    Order Status   : Delivered on {data3[0]}")
588         else:
589             print("    Order Status   : In transit")
590     print()
591     cam.close()
592     return
593
594 #
595 '''This function will allow administrator to view all orders of a particular date'''
596
597 def View_ord_dt(): # Admin can view any user's order by date
598     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
599     cursor = cam.cursor(buffered=True)
600     cursor.execute("SELECT ord_no FROM orders")
```

1: Project

orders.py x

```
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:
```

2: Structure

3: Favorites

4: AWS Explorer

orders.py

```
632
633     print("    Order Status : In transit")
634     print()
635     cam.close()
636     return
637
638 # -----
639 # |           Below Functions are only for non-admin users
640 # |
641 #
642 # -----
643 '''This function will allow users to cancel their order only if they cancel it within 7 days of the date since
644 order is placed. An email will be sent to the user with the details.'''
645
646 def Cancel_ord_u(username):
647     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
648     cursor = cam.cursor(buffered=True)
649     cursor.execute(
650         f"SELECT * from orders where user_name = '{username}' and DATEDIFF(NOW(), order_dt) <= 7 ORDER by ord_no DESC")
651     data = cursor.fetchall()
652     list1 = []
653     if not data:
654         print("None of your orders are eligible to cancel. Sorry :)")
655     else:
656         print("This is the list of your orders which are eligible to cancel: ")
657         for row in data:
658             tp = row
659             list1.append(tp[0])
660             print("    Order ID      : ", tp[0])
661             print("    Book Name    : ", capwords(tp[11]))
662             print("    Pieces       : ", tp[10])
663             print("    Customer Name : ", capwords(tp[1]))
```

 orders.py ×

```
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]}'")
```

1: Project

orders.py x

2: Structure

3: Favorites

4: WS Explorer

```
694 cursor.execute("SELECT * FROM orders, books WHERE books.title = orders.title")
695
696 data3 = cursor.fetchone()
697 msg = f"Your Order has been cancelled as per your request. Please review the product on our website. \n\n \
698     f"Book Name: {capwords(data2[11])} \n" \
699     f"Publication: {capwords(data3[0])} \n" \
700     f"Author's Name: {capwords(data3[1])} {capwords(data3[2])} \n" \
701     f"Order ID: {data2[0]} \n" \
702     f"Name: {capwords(data2[1])} \n" \
703     f"Address: {data2[2]} \n" \
704     f"District: {data2[3]} \n" \
705     f"City: {data2[4]} \n" \
706     f"State: {data2[5]} \n" \
707     f"Pincode: {data2[6]} \n" \
708     f"Country: {data2[7]} \n" \
709     f"Phone: {data2[9]} \n" \
710     f"Pieces Ordered: {data2[10]} \n" \
711     f"Order Date & Time: {data2[12]} \n" \
712     f"Order Status: Cancelled on {datetime.datetime.now()}"
713 email = data2[8]
714 sub = 'Regarding Cancellation of Order on S & T Bookshop'
715 Send_email(email, sub, msg)
716 cursor.execute("SET SQL_SAFE_UPDATES = 0")
717 cursor.execute(
718     f"UPDATE books SET stock_quantity = stock_quantity + {data2[10]} WHERE title = '{data2[11]}'")
719 cam.commit()
720 cursor.execute(f"DELETE FROM orders where ord_no = {ch}")
721 cam.commit()
722 print("Successfully Cancelled the Order.")
723 if len(list1) > 1:
724     ch2 = str(input("    Want to cancel more Orders? (y OR n) "))
725     if ch2 == 'y' or ch2 == 'Y':
726         Cancel_ord_u(username)
```

Project

Structure

Favorites

AWS Explorer

orders.py x

```
725             Cancel_ord_u(username)
726         elif ch2 == 'n' or ch2 == 'N':
727             cam.close()
728             return
729         else:
730             print("Wrong Input")
731             return
732     else:
733         pass
734
735 # -----
736 '''This function will allow administrator to update his/her order address if they do it within 7 days after
737 order is being placed. An email will be sent with few order details and new location of delivery.'''
738
739 def Up_ord_au(username):
740     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
741     cursor = cam.cursor(buffered=True)
742     cursor.execute(f"SELECT * FROM orders WHERE user_name = '{username}' and DATEDIFF(NOW(), order_dt) <= 7 ORDER BY ord_no DESC")
743     data = cursor.fetchall()
744     list1 = []
745     if data == None:
746         print("No 'IN TRANSIT' orders in database for your username.")
747     else:
748         print("This is the list of your orders eligible for updating the address: ")
749         for row in data:
750             tp = row
751             list1.append(tp[0])
752             print("    Order ID:      ", tp[0])
753             print("    Book Name:     ", capwords(tp[11]))
754             print("    Pieces:        ", tp[10])
755             print("    Customer Name: ", capwords(tp[1]))
756             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')
```

Project

Structure

Favorites

AWS Explorer

orders.py

```
789         cursor.execute(f"SELECT email FROM accounts WHERE username = '{username}'")
790         email = cursor.fetchone()
791         msg = f"Your Order has been updated successfully. \n"
792         msg += f"New Location of the order is {new_addr}, {new_city}, {new_pinc}, {new_city}, {new_stat} :)"
793         sub = "Information of Order Update on S & T Book Shop Management System"
794         Send_email(email[0], sub, msg)
795         print()
796
797     if len(list1) > 1:
798         ch2 = str(input("  Want to update more Orders? (y OR n): "))
799         if ch2 == 'y' or ch2 == 'Y':
800             Up_ord_au(username)
801         elif ch2 == 'n' or ch2 == 'N':
802             return
803         else:
804             print("Wrong Input")
805             return
806         else:
807             pass
808     cam.close()
809     return
810
811 #
812 '''This function will allow the signed in user to view all their details of every order placed since
813 registered with detail of order type (i.e, new or for replace or for return)'''
814
815 def View_au(username):
816     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
817     cursor = cam.cursor(buffered=True)
818     print("This is the list of your orders.: ")
819     cursor.execute(f"SELECT * FROM orders WHERE user_name = '{username}' ORDER BY ord_no DESC")
820     data = cursor.fetchall()
```



```
820     data = cursor.fetchall()
821
822     if data == None:
823         print("No orders in database.")
824     else:
825         print("          Latest Orders Sorted by Date:. ")
826
827         for row in data:
828             tp = row
829
830             print("    Order ID      : ", tp[0])
831             print("    Book Name    : ", capwords(tp[11]))
832             print("    Pieces       : ", tp[10])
833             print("    Customer Name: ", capwords(tp[1]))
834             print("    Address      : ", tp[2])
835             print("    District     : ", tp[3])
836             print("    City         : ", tp[4])
837             print("    State/UT     : ", tp[5])
838             print("    Pincode      : ", tp[6])
839             print("    Country      : ", tp[7])
840             print("    E-mail Address: ", tp[8])
841             print("    Phone        : ", tp[9])
842             print("    Order Date   : ", tp[12])
843             print("    Order Type   : ", capwords(tp[14]))
844
845             if tp[14] not in ['return', 'replace']:
846                 cursor.execute(f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
847                 data2 = cursor.fetchone()
848                 if not data2:
849                     cursor.execute(f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
850                     data3 = cursor.fetchone()
851                     print(f"    Order Status  : Delivered on {data3[0]}")
852                 else:
853                     print("    Order Status  : In transit")
854
855             print()
```

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

*End of Module*

**acc\_ctrl.py**

Project  
1: acc\_ctrl.py

```
1 import mysql.connector as conn2
2 from time import sleep
3 from validate_email import validate_email
4
5 # -----
6 '''This function will help the administrator to add new users.
7     Here, Username entered will be accepted only if same username is already not present in database.
8     Secondly, here email entered will be accepted only if it really exists
9     (i.e) the given email-address has an MX record and SMTP server port. Disposable emails will
10    not be accepted here. Module used here is 'validate_email' '''
11
12 def Add_new():
13     mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
14     cursor = mycon2.cursor(buffered=True)
15     no_user = int(input('Enter number of user you want to add: '))
16     for i in range(no_user):
17         input_username = input("    Enter Login ID for the user: ")
18         cursor.execute(f"SELECT username FROM accounts WHERE username = '{input_username}'")
19         data = cursor.fetchone()
20         if not data:
21             input_password = input("    Enter Password of the user: ")
22             input_name = input("    Enter name of the user: ")
23
24             def ent():
25                 input_email = input("    Enter email of the user: ")
26                 cursor.execute(f"SELECT * FROM accounts WHERE email = '{input_email}'")
27                 data2 = cursor.fetchone()
28                 if data2 != None:
29                     print("E-mail Address Already Registered.")
30                     ent()
```

## Structure

## Favorites

## AWS Explorer

1: Project

acc\_ctrl.py x

2: Structure

3: Favorites

4: WS Explorer

```
30     ent()
31 else:
32     print("Validating Details, please wait.....")
33     try:
34         is_valid = validate_email(input_email)
35         if is_valid:
36             cursor.execute(
37                 f"INSERT INTO accounts (username, passwd, name_u, email) VALUES('{input_username}', '{input_password}', "
38                 f"'{input_name}', '{input_email}')")
39             mycon2.commit()
40             print("Registering")
41             sleep(2)
42             print(f"Successfully Added User '{input_username}'")
43     except:
44         print("Some Error Occurred.")
45
46     ent()
47 else:
48     print("User already in database.")
49     continue
50 mycon2.close()
51 return
52
53 # -----
54 '''This function will allow the administrator to update the every detail of the user like name, password
55 username and email-address. Here email entered will be accepted only if it really exists (i.e) the given
56 email-address has an MX record and SMTP server port. Disposable emails will not be accepted here.
57 Modules used here are 'validate_email' and 'mysql.connector' '''
58
59 def Update_user():
```

```
59     def Update_user():
60         mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
61         cursor = mycon2.cursor(buffered=True)
62         no_user = int(input('Enter number of user you want to update: '))
63         count = 0
64         while count < no_user:
65             user_name = str(input("    Enter the username of the user you want to update: "))
66             cursor.execute("SELECT * FROM accounts WHERE username = '{}'".format(user_name))
67             data = cursor.fetchone()
68             if data == None:
69                 print(f"No user like '{user_name}' exists in database.") # Checking the wrong Input
70                 Update_user()
71             elif data[0] == 1:
72                 print("Can't Update user. The specified username is Administrator.")
73             else:
74                 print("      1 -> Update the user's username \n",
75                     "      2 -> Update the user's password \n"
76                     "      3 -> Update the user's name \n"
77                     "      4 -> Update the user's email address")
78             choice = int(input("Enter your choice (1 to 4): "))
79             if choice == 1:
80                 new_username = str(input(f"    Enter the new username of {user_name}: "))
81                 cursor.execute("SELECT * FROM accounts WHERE username = '{}'".format(new_username))
82                 data2 = cursor.fetchone()
83                 if data2 == None:
84                     cursor.execute(f"UPDATE accounts SET username = '{new_username}' WHERE username = '{user_name}'")
85                     mycon2.commit() # Updating the username
86                     print("Successfully Updated")
87                 else:
88                     print("Username Already Exists. \n User Not Updated :(")
```

1: Project

acc\_ctrl.py x

```
89     elif choice == 2:
90         new_password = str(input("    Enter the new password of the {user_name}: "))
91         cursor.execute(f"UPDATE accounts SET passwd = '{new_password}' WHERE username = '{user_name}'")
92         mycon2.commit() # Updating the user
93         print("Successfully Updated")
94     elif choice == 3:
95         new_name = str(input("    Enter the new name of the {user_name}: "))
96         cursor.execute(f"UPDATE accounts SET name_u = '{new_name}' WHERE username = '{user_name}'")
97         mycon2.commit() # Updating the user
98         print("Successfully Updated")
99     elif choice == 4:
100        new_mail = str(input("    Enter the new email address of the {user_name}: "))
101        cursor.execute(f"SELECT email FROM accounts WHERE email = '{new_mail}'")
102        data3 = cursor.fetchone()
103        if data3 == None:
104            is_valid = validate_email(new_mail)
105            if is_valid:
106                cursor.execute(f"UPDATE accounts SET email = '{new_mail}' WHERE username = '{user_name}'")
107                mycon2.commit() # Updating the user
108                print("Successfully Updated")
109            else:
110                print("Invalid Email Address. User not updated :(")
111            else:
112                print("Email Address Already registered. \n User Not updated :(")
113            else:
114                print("Wrong Input")
115            Update_user()
116            count += 1
117        mycon2.close()
118        return
```

2: Structure

3: Favorites

4: AWS Explorer

1: Project

acc\_ctrl.py x

```
120 # -----
121 ''' This function will help administrator to delete any user he/she wants from database
122     by entering his/her username if it exists. But admin can't delete the administrator account
123     Module used here is 'mysql.connector' '''
124
125 def Del_user():
126     mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
127     cursor = mycon2.cursor(buffered=True)
128     no_user = int(input("Enter number of accounts you want to delete: "))
129     count = 0
130     while count < no_user:
131         user_name = str(input("    Enter the username of the user you want to delete: "))
132         cursor.execute(f"SELECT * FROM accounts WHERE username = '{user_name}'")
133         data = cursor.fetchone()
134         if data == None:
135             print(f"No user like '{user_name}' exists in database." ) # Checking the Wrong Input
136             Del_user()
137         elif data[0] == 1:
138             print("Can't Delete user. The specified username is Administrator.")
139         else:
140             cursor.execute(f"DELETE FROM accounts where username = '{user_name}'")
141             mycon2.commit() # Deleting the user
142             sleep(2)
143             print("Successfully Deleted.")
144             count += 1
145     mycon2.close()
146
147     return
```

2: Structure

3: Favorites

4: Errors

1: Project  
acc\_ctrl.py

```
148 # -----
149 ''' This function will allow administrator to view every detail of every user in database.
150     Module used here is mysql.connector '''
151
152 def All_u():
153     mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
154     cursor = mycon2.cursor(buffered=True)
155     cursor.execute("SELECT * FROM accounts")
156     data = cursor.fetchall()
157     count = 0
158     for row in data:
159         tp = row
160         count += 1
161         print()
162         print(f"      User {count}")
163         print("      ID          : ", tp[0])
164         print("      Username    : ", tp[1])
165         print("      Password    : ", tp[2])
166         print("      Name        : ", tp[3])
167         print("      E-mail Address : ", tp[4])
168     mycon2.close()
169     return
170
171 # -----
172 '''This function will help administrator to view details of the user by entering
173     his/her username if it exists in database.'''
174
175 def View_user():
176     mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
177     cursor = mycon2.cursor(buffered=True)
```

2: Favorites

3: WS Explorer

## acc\_ctrl.py x

```
177     cursor = mycon2.cursor(buffered=True)
178     num = int(input("Enter the number of users of which you want details of: "))
179     count = num
180     while count != 0:
181         user_name = str(input("    Enter the username of the user: "))
182         cursor.execute(f"SELECT * FROM accounts WHERE username = '{user_name}'")
183         data = cursor.fetchone()
184         if not data:
185             print("Username specified is invalid. Enter Again.")
186             print()
187         else:
188             print(f"    ID: {data[0]}")
189             print(f"    Username: {data[1]}")
190             print(f"    Name: {data[3]}")
191             print(f"    Email-Address: {data[4]}")
192             print()
193             count -= 1
194     return
195
196 # -----
197 '''Sign_Up function will allow any new user running the software to register himself in database.
198 For this, they have to create a username, password, give their name and enter their email address which
199 really exists as it will be checked then only the user will be registered.
200 Module used here is 'validate_email' and 'mysql.connector' '''
201
202 def Sign_Up():
203     mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
204     cursor = mycon2.cursor(buffered=True)
205     input_username = input("            Create Username (Login ID) : ")
206     cursor.execute(f"SELECT * FROM accounts WHERE username = '{input_username}'")
```

```
206     cursor.execute(f"SELECT * FROM accounts WHERE username = '{input_username}'")
207     data = cursor.fetchone()
208     if data == None:
209         input_password = input("          Create Password: ")
210         input_name = input("          Enter your name: ")
211
212     def ent():
213         input_email = input("          Enter your email address: ")
214         cursor.execute(f"SELECT * FROM accounts WHERE email = '{input_email}'")
215         data2 = cursor.fetchone()
216         if data2 != None:
217             print("E-mail Address Already Registered.")
218             ent()
219         else:
220             print("Validating Details, please wait.....")
221             is_valid = validate_email(input_email)
222             if is_valid:
223                 cursor.execute(
224                     f"INSERT INTO accounts (username, passwd, name_u, email) VALUES('{input_username}', "
225                     f"'{input_password}', '{input_name}', '{input_email}')")
226                 mycon2.commit()
227                 print("Registering")
228                 sleep(2)
229                 print(f"Successfully Added User '{input_username}'")
230
231             ent()
232         else:
233             print("User Already In Database.")
234             print("Try Again")
235             Sign_Up()
236     mycon2.close()
237     return
```

**config.py**

## Class 12 Computer Project &gt; config.py &gt;

1: Project config.py x

```
1 import mysql.connector as mycam2
2 from fpdf import FPDF
3 from pdf_mail import sendpdf
4 from datetime import date
5 from string import capwords
6 #
7 '''Below EMAIL_ADDRESS and PASSWORD is for sending email(s) to users (constant values)'''
8
9 EMAIL_ADDRESS = 'snt.bookshop@gmail.com'
10 PASSWORD = 'S&T@bookshp@341'
11
12 #
13 '''This function is defined to send invoice of the order placed by any user
14 which will fetch the given PDF INVOICE from the specific path where all invoices generated are saved.
15 Module used here is 'pdf_mail' '''
16
17 def Pdf_mailing(file_name, address):
18     sender_email_address = EMAIL_ADDRESS
19     receiver_email_address = f"{address}"
20     sender_email_password = PASSWORD
21     subject_of_email = "Information of Order Placed on S & T Book Shop"
22     body_of_email = "This is an auto generated bill of supply for your ordered book(s). \nIf any error is there, please reply to us " \
23                     "on our E-mail Address snt.bookshop@gmail.com. " \
24                     "Thanks for shopping. :)"
25     filename = file_name
26     location_of_file = "D:/Swarit/Class 12/Class 12 Computer Project/invoices"
27     k = sendpdf(sender_email_address, receiver_email_address,
28                 sender_email_password,
29                 subject_of_email,
30                 body_of_email,
31                 filename, location_of_file)
32     k.email_send()
```

Z: Structure

2: Favorites

AWS Explorer

```
36     '''This function do two works. First, it makes a string sum of ASCII values of the username of the user  
37     Second it trims the string generated to length of 10 characters, query of the numbers of orders placed  
38     by the user since registered and store it in another string which is displayed in invoice number field  
39     of invoice generated on placing order.  
40     Module used here is 'mysql.connector'. '''  
41  
42     def Inv_no_gen(a):  
43         cam = mycam2.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')  
44         cursor = cam.cursor()  
45         st = ''  
46         st2 = ''  
47         for char in a:  
48             x = ord(char)  
49             st += str(x) # String sum of ASCII characters of username passed into function generated in st  
50  
51         # Underneath 'for' loop is defined to limit the number of characters of the username to 10 so that  
52         # when invoice number will be displayed in the invoice it doesn't go out of the cell defined for it.  
53         # This is stored in 'st2'  
54         if len(st) <= 10:  
55             st2 += st  
56         else:  
57             for i in range(0, 10):  
58                 st2 += st[i]  
59  
60             # This query is done so as to get the total number of orders placed since the user has registered  
61             cursor.execute(f"SELECT count(ord_no) FROM orders WHERE user_name = '{a}'")  
62             ord_count = cursor.fetchone()  
63             # We are adding the result of the above query to above generated 'st2' and adding 1 to it  
64             # so that the invoice number generated is unique every time and no invoice gets replaced  
65             # due to same name  
66             st2 += f'{ord_count[0] + 1}'  
67             return st, st2
```

```
69
70 # -----
71 '''This function generates PDF invoice with all the details of the user and the
72 book shop and save it on local desktop. User is asked whether he/she wants to get invoice on his/her
73 email-address or not. Invoice will be sent to the user's email address only if the user says 'y'.
74 Invoice name is string sum of all the ASCII values of the characters of username of the user and the last digit
75 will be one more than the number of all orders placed since the user has registered himself/herself.
76 This is done so that the invoice generated last time do not get replaced upon new order placed on local desktop.
77 Modules used here are 'fpdf' and 'mysql.connector.'''
78
79 def Pdf_generate(ls1, ls2, inv, inv_no):
80     lst_sum = []
81     cam = mycam2.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
82     cursor = cam.cursor()
83     cursor.execute(f"SELECT count(ord_no) FROM orders WHERE user_name = '{ls1[10]}'")
84     ord_count = cursor.fetchone()
85     pdf = FPDF()
86     pdf.add_page()
87
88     pdf.set_font("Arial", 'B', size=15)
89
90     pdf.cell(130, 5, 'S & T BOOKSHOP', 0, 0, 'L')
91     pdf.cell(60, 5, 'Bill of Supply/ Invoice', 0, 1, 'C')
92
93     pdf.set_font("Arial", '', size=12)
94
95     pdf.cell(130, 5, 'Street Address: K-61/74, Sector 8, Pandeypur', 0, 0, 'L')
96     pdf.cell(60, 5, '', 0, 1, 'L')
97
98     pdf.cell(130, 5, 'Varanasi, India, ZIP-221002', 0, 0, 'L')
99     pdf.cell(25, 5, 'Date: ', 0, 0, 'L')
100    pdf.cell(34, 5, f'{date.today()}', 0, 1, 'L')
```

Project



config.py ×

```
1: Project
  102     pdf.cell(130, 5, 'Phone: (+91)8004125336', 0, 0, 'L')
  103     pdf.cell(25, 5, 'Invoice #', 0, 0, 'L')
  104     pdf.cell(34, 5, f'{inv_no}', 0, 1, 'L')

  105
  106     pdf.cell(130, 5, 'Fax: [+ 91 542 14512]', 0, 0, 'L')
  107     pdf.cell(25, 5, 'Customer ID', 0, 0, 'L')
  108     pdf.cell(34, 5, f'{ls1[8][0]}', 0, 1, 'L')

  109
  110     pdf.cell(130, 5, 'GSTIN: 22AABCU9603R1ZX', 0, 1, 'L')

  111
  112     pdf.cell(189, 10, '', 0, 1, 'L')

  113
  114     pdf.set_font("Arial", 'B', size=15)

  115
  116     pdf.cell(100, 5, 'Billing Address / Shipping Address: ', 0, 1)

  117
  118     pdf.set_font('Arial', '', size=12)

  119
  120     pdf.cell(10, 5, '', 0, 0)
  121     pdf.cell(90, 5, f'{ls1[0]}', 0, 1) # Name

  122
  123     pdf.cell(10, 5, '', 0, 0)
  124     pdf.cell(90, 5, f'{ls1[1]}, {ls1[3]}', 0, 1) # Address 1

  125
  126     pdf.cell(10, 5, '', 0, 0)
  127     pdf.cell(90, 5, f'{ls1[2]}, {ls1[4]}', 0, 1) # Address 2

  128
  129     pdf.cell(10, 5, '', 0, 0)
  130     pdf.cell(90, 5, f'{ls1[5]}, {ls1[6]}', 0, 1) # Pincode

  131
  132     pdf.cell(10, 5, '', 0, 0)
  133     pdf.cell(90, 5, f'Phone: {ls1[7]}', 0, 1) # Phone
```

Structure



Favorites



AWS Explorer



```
config.py x
135     pdf.cell(189, 10, '', 0, 1, 'L')
136
137     pdf.set_font('Arial', 'B', size=12)
138
139     pdf.cell(10, 5, 'S.no', 1, 0, 'C')
140     pdf.cell(100, 5, 'Description', 1, 0, 'C')
141     pdf.cell(25, 5, 'Quantity', 1, 0, 'C')
142     pdf.cell(25, 5, 'Rate', 1, 0, 'C')
143     pdf.cell(30, 5, 'Amount(Rs.)', 1, 1, 'C')
144
145     pdf.set_font('Arial', '', 12)
146
147 # For Loop defined below is to enter the details(quantity, rate, bookname, author's name and amount)
148 # into the invoice (i.e.) one invoice will be generated until an unless user exits the function.
149 # The last provided address and phone will be written in the billing address of thr invoice
150
151 for i in range(len(ls2)):
152     cursor.execute(f"SELECT bookname, pieces FROM orders WHERE ord_no = {ls2[i]}")
153     data = cursor.fetchone()
154     cursor.execute(f"SELECT author_fname, author_lname, price FROM books WHERE title = '{data[0]}'")
155     data2 = cursor.fetchone()
156     pdf.cell(10, 5, f"{i + 1}", 1, 0, 'R')
157     pdf.cell(100, 5, f'{capwords(data[0])} - {capwords(data2[0])} {capwords(data2[1])}', 1, 0, align='L')
158     pdf.cell(25, 5, f'{data[1]}', 1, 0, align='R')
159     pdf.cell(25, 5, f'{data2[2]}', 1, 0, align='R')
160     pdf.cell(30, 5, f'{data[1] * data2[2]}', 1, 1, align='R')
161     lst_sum.append(data[1] * data2[2])
162
163     # Here total of all the books ordered will be calculated
164     total = 0
165     for num in lst_sum:
166         total += num
```

1: Project

2: Structure

3: Favorites

config.py x

```
166
167     pdf.cell(10, 5, '', 0, 0)
168     pdf.cell(100, 5, '', 0, 0)
169     pdf.cell(25, 5, '', 0, 0)
170
171     pdf.set_font('Arial', 'B', size=12)
172
173     pdf.cell(25, 5, 'Total', 1, 0, 'C')
174
175     pdf.set_font('Arial', 'B', size=12)
176     # Total calculated above will be displayed here
177     pdf.cell(30, 5, f'{total}', 1, 1, 'R')
178
179     file_n = f"invoice{inv}{ord_count[0] + 1} "
180     path = f"D:/Swarit/Class 12/Class 12 Computer Project/invoices/{file_n}.pdf"
181     pdf.output(f"{path}") # Invoice generated and saved to local desktop
182
183     choice = str(input("    Do you want the invoice to be sent to your email address? (y/n): ")).lower()
184     if choice in ['y', 'yes']:
185         Pdf_mailing(file_n, address=ls1[9])
186         print(f"An email has been sent to you with an invoice to your email address {ls1[9]}")
187     else:
188         pass
189
190     # / End of Module /
191
192
```

**main\_menu.py**

## Class 12 Computer Project &gt; main\_menu.py &gt;

1: Project main\_menu.py x

```
1 # Python Modules
2 import mysql.connector as conn
3 import time
4
5 # Project Modules
6 import books
7 import orders
8 import acc_ctrl
9
10 start_time = time.time()
11
12 print(" ----- [ Welcome to S & T Book Shop Management System ] -----")
13
14
15 def Main_menu(a, b, c):
16     # Login in with Account
17     mycon = conn.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
18     login = 0
19     while login != 1: # Checking
20         cursor = mycon.cursor() # for a valid user
21         cursor.execute(f"select * from accounts where username = '{a}' and passwd = '{b}' ")
22         data = cursor.fetchone()
23         if data == None:
24             print("Wrong Credentials")
25             break
26         else:
27             if data[0] == 1: # Checking
28                 while c != 1: # for the root(admin) user
29                     print(" Successfully Logged In. \n"
30                           " You are Administrator :)")
31                     c = 1
32             print("Select your choice: \n")
```

2: Structure

3: Favorites

4: AWS Explorer

 main\_menu.py X

```
32     print("Select your choice: \n"
33         "    1 -> Add A New Book \n" +
34         "    2 -> Update A book Information \n" +
35         "    3 -> Remove A book \n" +
36         "    4 -> View Details of a book \n" +
37         "    5 -> View All Books \n" +
38         "    6 -> Place an Order \n" +
39         "    7 -> Cancel an Order \n"
40         "    8 -> Return/Replace an Order \n" +
41         "    9 -> Update an Order Address \n" +
42         "   10 -> View All Orders \n" +
43         "   11 -> View Orders of a User(s) \n"
44         "   12 -> View Orders by Date \n" +
45         "   13 -> Add a New User \n" +
46         "   14 -> Update A User's Credentials \n" +
47         "   15 -> Remove A User \n"
48         "   16 -> View all Users \n"
49         "   17 -> View Details of some users \n" +
50         "   18 -> Exit")
51 ch = int(input("Enter your choice from (1 to 18): "))
52 if ch == 1:
53     books.Add_book()
54 elif ch == 2:
55     books.Update_book()
56 elif ch == 3:
57     books.Delete_book()
58 elif ch == 4:
59     books.View_details()
60 elif ch == 5:
61     books.View_all()
62 elif ch == 6:
63     orders.Place_ord(a)
```

1: Project

2: Structure

3: Favorites

4: AWS Explorer

## main\_menu.py

```
64         elif ch == 7:
65             orders.Cancel_ord()
66         elif ch == 8:
67             orders.Return_replace(a)
68         elif ch == 9:
69             orders.Update_ord_addr()
70         elif ch == 10:
71             orders.View_all_ords()
72         elif ch == 11:
73             orders.View_all_u()
74         elif ch == 12:
75             orders.View_ord_dt()
76         elif ch == 13:
77             acc_ctrl.Add_new()
78         elif ch == 14:
79             acc_ctrl.Update_user()
80         elif ch == 15:
81             acc_ctrl.Del_user()
82         elif ch == 16:
83             acc_ctrl.All_u()
84         elif ch == 17:
85             acc_ctrl.View_user()
86         elif ch == 18:
87             end_time = time.time()
88             print("Successfully Logged Out. Your active time was ", end_time - start_time, "seconds")
89             exit(0)
90         else:
91             print("Wrong Input")
92     login = 1
93     do_again = input("Go Back to Main Menu? (y -> Yes, n -> No and Logout) ")
94     if do_again in ['y', 'yes', 'Y', 'YES']:
95         Main_menu(a, b, c)
```

1: Project

main\_menu.py x

```
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
```

```
else:
    end_time = time.time()
    print("Process Successfully Done. Logged Out")
    print("Your active time was ", end_time - start_time, "seconds")
    exit(0)

else:
    print("    Successfully Logged In.")
    print("    1-> Place an Order \n"
          "    " 2-> Cancel an Order \n"
          "    " 3-> Return or Replace an Order \n"
          "    " 4-> Update an Order Address \n"
          "    " 5-> View all your Orders \n" +
          "    " 6-> Exit")

    ch2 = int(input("Enter your choice from (1 to 6): "))
    if ch2 == 1:
        orders.Place_ord(a)
    elif ch2 == 2:
        orders.Cancel_ord_u(a)
    elif ch2 == 3:
        orders.Return_replace(a)
    elif ch2 == 4:
        orders.Up_ord_au(a)
    elif ch2 == 5:
        orders.View_au(a)
    elif ch2 == 6:
        end_time = time.time()
        print("Successfully Logged Out. Your active time was ", end_time - start_time, "seconds")
        exit(0)

else:
    print("Wrong Input.")

login = 1
do_again = input("Go Back to Main Menu? (y -> Yes, n -> No and Logout) ")
```

2: Structure

3: Favorites

4: AWS Explorer

```
128     if do_again in ['y', 'yes', 'Y', 'YES']:
129         Main_menu(a, b, c)
130     else:
131         end_time = time.time()
132         print("Process Successfully Done. Logged Out")
133         print("Your active time was ", end_time - start_time, "seconds")
134         exit(0)
135     log_ag = str(input("Enter login details again? (y -> Yes, n -> No and exit): "))
136     if log_ag == 'y' or log_ag == 'Y':
137         user_id = Usern()
138         user_pass = Userp()
139         Main_menu(user_id, user_pass, c)
140     else:
141         end_time = time.time()
142         print("Successfully Logged Out. Your active time was ", end_time - start_time, "seconds")
143         exit(0)
144
145
146 # __ main __
147 # Calling the main program
148 print("1 -> Sign Up \n"
149       "2 -> Sign In")
150 sign = int(input("Enter your choice: "))
151 count = 0
152
153 if sign == 1:
154     tp = acc_ctrl.Sign_Up()
155     print("Restart the software to Login.....:")
156 elif sign == 2:
157
158     def Usern():
```

```
158     def Usern():
159         input_username = input(" Enter Login ID: ").strip()
160         return input_username
161
162     def Userp():
163         input_password = input(" Enter Password: ").strip()
164         return input_password
165
166     x = Usern()
167     y = Userp()
168     Main_menu(x, y, count)
169
170 else:
171     print("Wrong Input")
172
```

1: Structure

2: Favorites

# **Outputs of Non-Administrator User**

# Signing Up

```
Run:  main_menu x
C:\Users\dell\AppData\Local\Programs\Python\Python39\python.exe "D:/Swar
----- [ Welcome to S & T Book Shop Management System ]
1 --> Sign Up
2 --> Sign In
Enter your choice: 1
Create Username (Login ID) : sjain12sw
Create Password: qwerty
Enter your name: Swasti Jain
Enter your email address: swjain446@gmail.com
Validating Details, please wait.....
Registering
Successfully Added User 'sjain12sw'
Restart the software to Login.....:)
Process finished with exit code 0
```

# Placing an Order

```
Run: main_menu x
C:\Users\dell\AppData\Local\Programs\Python\Python39\python.exe "D:/Swarit/Class 12/Class 12 Computer Project/main_menu.py"
----- [ Welcome to S & T Book Shop Management System ] -----
1 -> Sign Up
2 -> Sign In
Enter your choice: 2
Enter Login ID: sjain12sw
Enter Password: qwerty
Successfully Logged In.
1-> Place an Order
2-> Cancel an Order
3-> Return or Replace an Order
4-> Update an Order Address
5-> View all your Orders
6-> Exit
Enter your choice from (1 to 6): 1

Enter the name of the book : coraline
Book named , Coraline, found details are as follows:-
Author's name: Neil Gaiman
Released Year: 2016
Number of Pages: 208
Words Per Page (Approx): 175
Publication: Bloomsbury Children's Books (30 August 2016)'
Category of Book: Mystery
Available quantity: 100
Price of the book: 216
Do you want to order this Book ? y
Enter the QUANTITY you want to order: 2
```

P.T.O.

Run: main\_menu x

▶ Enter the QUANTITY you want to order: 2  
Checking Stock, please wait...  
Enough quantity found

Please fill up your details below:

Enter Name: Swasti Jain  
Enter Address: S-61/74, Pandeypur  
Enter District: Varanasi  
Enter City: Varanasi  
Enter State/UT: Uttar Pradesh  
Enter Pincode: 221004  
Enter Country: India  
Enter Phone number: 9876543210

Now we came at the final step of the process...

Total Amount to be paid is: 432

No Delivery Charges Applicable, i.e. Delivery Free

Are you sure you want to confirm this order? y

Processing your order....

Order Placed Successfully.

Want to place more orders? (y/n): y

Enter the name of the book : American Gods

Book named , American Gods, found details are as follows:-

Author's name: Neil Gaiman  
Released Year: 2002  
Number of Pages: 736  
Words Per Page (Approx): 145

Run:

main\_menu x

Publication: Headline Review (4 March 2002)  
Category of Book: Drama  
Available quantity: 12  
Price of the book: 364

Do you want to order this Book ? y

Enter the QUANTITY you want to order: 5

Checking Stock, please wait...

Enough quantity found

Please fill up your details below:

Enter Name: Swasti Jain

Enter Address: S-61/74, Pandeypur

Enter District: Varanasi

Enter City: Varanasi

Enter State/UT: Uttar Pradesh

Enter Pincode: 221004

Enter Country: India

Enter Phone number: 9876543210

Now we came at the final step of the process...

Total Amount to be paid is: 1820

No Delivery Charges Applicable, i.e. Delivery Free

Are you sure you want to confirm this order? y

Processing your order....

Order Placed Successfully.

Want to place more orders? (y/n): n

Do you want the invoice to be sent to your email address? (y/n): y

An email has been sent to you with an invoice to your email address swjain446@gmail.com

Thanks for visiting us :)

Go Back to Main Menu? (y -> Yes, n -> No and Logout) n

Process Successfully Done. Logged Out

Your active time was 184.12249279022217 seconds

Process finished with exit code 0

# Email Sent on Placing Order on Email When Signing Up

M Information of Order Placed on S & T Book Shop x +

mail.google.com/mail/u/0/?hl=en\_GB#inbox/ FMfcgxwKjTPhtkrWLLxFVfzFsJBKRFTg

YouTube Unacademy Smartkeeda - Leadin... Python File Handlin... BANKING CHRONI...

Gmail Search mail

Compose

Inbox 2

Starred

Snoozed

Sent

Drafts

STUDY MATERIALS

More

Meet

New meeting

Join a meeting

Hangouts

Swasti

No recent chats Start a new one

Information of Order Placed on S & T Book Shop Inbox x

snt.bookshop@gmail.com to me ▾

This is an auto generated bill of supply for your ordered book(s).  
If any error is there, please reply to us on our E-mail Address snt.bookshop@gmail.com. Thanks for shopping. :)



PDF invoice115106971...

Reply Forward

Swasti Jain swjain446@gmail.com

Manage your Google Account

swaritjain123@gmail.co... swaritjain123@gmail.com ✓ Signed out

Add another account

Sign out

Privacy Policy • Terms of Service

# Auto Generated Invoice Sent as PDF to Email Address of the User

Information of Order Placed on S & T BOOKSHOP

mail.google.com/mail/u/0/?hl=en\_GB#inbox/ FMfcgxwKjTPhtkrWLLxFVfzFsJBKRFTg?projector=1&messagePartId=0.1

YouTube Unacademy Smartkeeda - Lead... Python File Handlin... BANKING CHRONI...

← PDF invoice1151069710511049501151193 Search mail

Compose

Inbox 2

Starred

Snoozed

Sent

Drafts

STUDY MATERIALS

More

New meeting

Join a meeting

Swasti

No recent chats Start a new one

Enable desktop notifications for Gmail. OK

Open with Google Docs

**S & T BOOKSHOP**  
Street Address: K-61/74, Sector 8, Pandeypur  
Varanasi, India, ZIP-221002  
Phone: (+91)8004125336  
Fax: [+ 91 542 14512]  
GSTIN: 22AABCU9603R1ZX

**Bill of Supply/ Invoice**  
Date: 2020-11-13  
Invoice # 11510697103  
Customer ID 3

**Billing Address / Shipping Address:**  
Swasti Jain  
S-61/74, Pandeypur, Varanasi  
Varanasi, Uttar Pradesh  
221004, India  
Phone: 9876543210

S.no	Description	Quantity	Rate	Amount(Rs.)
1	Coraline - Neil Gaiman	2	216	432
2	American Gods - Neil Gaiman	5	364	1820
			Total	2252

Page 1 / 1 - +

1 of 3 18:15 (3 minutes ago)

# Cancelling an Order

```
Run: main_menu x
C:\Users\dell\AppData\Local\Programs\Python\Python39\python.exe "D:/Swarit/Class 12/Class 12 Computer Project/main_menu.p
----- [ Welcome to S & T Book Shop Management System ] -----
1 -> Sign Up
2 -> Sign In
Enter your choice: 2
Enter Login ID: sjain12sw
Enter Password: qwerty
Successfully Logged In.
1-> Place an Order
2-> Cancel an Order
3-> Return or Replace an Order
4-> Update an Order Address
5-> View all your Orders
6-> Exit
Enter your choice from (1 to 6): 2
This is the list of your orders which are eligible to cancel:
Order ID      : 2
Book Name     : American Gods
Pieces        : 5
Customer Name : Swasti Jain
Address       : Flat No.89, Shree Ram Apartments, Sigra
District      : Varanasi
City          : Varanasi
State         : Uttar Pradesh
Pincode       : 221002
Country        : India
E-mail Address : swjain446@gmail.com
Phone         : 9876543210
```

P.T.O

Run: main\_menu x

↻ 🖍 Phone : 9876543210  
✖ ⚙ Order Date : 2020-11-13 18:15:02  
⬆️ ⚙ Order Type : new  
⬇️ ⚙ Order Status : In transit

🕒 ⚙ Order ID : 1  
🖨️ ⚙ Book Name : Coraline  
🖨️ ⚙ Pieces : 2  
🔴 ⚙ Customer Name : Swasti Jain  
📍 ⚙ Address : S-61/74, Pandeypur  
📍 ⚙ District : Varanasi  
📍 ⚙ City : Varanasi  
📍 ⚙ State : Uttar Pradesh  
📍 ⚙ Pincode : 221004  
📍 ⚙ Country : India  
✉️ ⚙ E-mail Address : swjain446@gmail.com  
📞 ⚙ Phone : 9876543210  
📅 ⚙ Order Date : 2020-11-13 18:14:02  
📍 ⚙ Order Type : new  
📍 ⚙ Order Status : In transit

Enter the Order ID to cancel: 2

An email with all details is sent to you on your E-mail Address swjain446@gmail.com

Successfully Cancelled the Order.

Want to cancel more Orders? (y OR n) n

Go Back to Main Menu? (y -> Yes, n -> No and Logout) y

# Email Sent on Cancelling the Order

Regarding Cancellation of Order o X + mail.google.com/mail/u/0/?hl=en\_GB#inbox/ FMfcgxwKjTPjDqqPgfhqjmMSrrxFXNt

YouTube Unacademy Smartkeeda - Lead... Python File Handlin... BANKING CHRONI...

Gmail Search mail

Compose

Inbox 2

Starred Snoozed Sent Drafts STUDY MATERIALS More

Meet New meeting Join a meeting

Hangouts Swasti +

Regarding Cancellation of Order on S & T Bookshop Inbox x

snt.bookshop@gmail.com to ▾

Your Order has been cancelled as per your request. Please review the product on our website.

Book Name: American Gods  
Publication: Headline Review (4 March 2002)  
Author's Name: Neil Gaiman  
Order ID: 2  
Name: Swasti Jain  
Address: Flat No.89, Shree Ram Apartments, Sigma  
District: Varanasi  
City: Varanasi  
State: Uttar Pradesh  
Pincode: 221002  
Country: India  
Phone: 9876543210  
Pieces Ordered: 5  
Order Date & Time: 2020-11-13 18:15:02  
Order Status: Cancelled on 2020-11-13 18:30:00.913499

Swasti Jain swjain446@gmail.com

Manage your Google Account

swaritjain123@gmail.co... swaritjain123@gmail.com v Signed out

Add another account

Sign out

Privacy Policy • Terms of Service

Enter your choice from (1 to 6): 4

This is the list of your orders eligible for updating the address:

Order ID: 1  
Book Name: Coraline  
Pieces: 2  
Customer Name: Swasti Jain  
Address: S-61/74, Pandeypur  
District: Varanasi  
City: Varanasi  
State/UT: Uttar Pradesh  
Pincode: 221004  
Country: India  
E-mail Address: swjain446@gmail.com  
Phone: 9876543210  
Order Date: 2020-11-13 18:14:02

Enter the Order ID to update address: 1

Enter the new address: Flat No.89, Shree Ram Apartments, Sigra

Enter City: Varanasi

Enter pincode: 221002

Enter District: Varanasi

Enter State/UT: Uttar Pradesh

Successfully Updated

An email with all details is sent to you on your E-mail Address swjain446@gmail.com

Go Back to Main Menu? (y -> Yes, n -> No and Logout) n

Process Successfully Done. Logged Out

Your active time was 318.53522515296936 seconds

# Email Sent on Updating Order Address

The screenshot shows a Gmail inbox interface. The top bar displays the title "Information of Order Update on S & T Book Shop Management System" and the URL "mail.google.com/mail/u/0/?hl=en\_GB#inbox/ FMfcgxwKjTPhtksRFmHkktISPJVisnL". Below the bar are several tabs and icons for YouTube, Unacademy, Smartkeeda, Python File Handling, and BANKING CHRONI... The main area shows an email from "snt.bookshop@gmail.com" with the subject "Information of Order Update on S & T Book Shop Management System". The email body contains the message: "Your Order has been updated successfully. New Location of the order is Flat No.89, Shree Ram Apartments, Sigra, Varanasi, 221002, Varanasi, Uttar Pradesh :)" and two buttons for "Reply" and "Forward". The left sidebar lists "Compose", "Inbox" (with 2 notifications), "Starred", "Snoozed", "Sent", "Drafts", "STUDY MATERIALS", "More", "Meet", and "New meeting". The top right corner shows a gear icon, a question mark, and other settings.

Information of Order Update on S & T Book Shop Management System

snt.bookshop@gmail.com

Your Order has been updated successfully.

New Location of the order is Flat No.89, Shree Ram Apartments, Sigra, Varanasi, 221002, Varanasi, Uttar Pradesh :)

Reply Forward

Inbox 2

Starred

Snoozed

Sent

Drafts

STUDY MATERIALS

More

Meet

New meeting

# Exiting

```
Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
Successfully Logged In.

1-> Place an Order
2-> Cancel an Order
3-> Return or Replace an Order
4-> Update an Order Address
5-> View all your Orders
6-> Exit

Enter your choice from (1 to 6): 6
Successfully Logged Out. Your active time was 66.1079671382904 seconds

Process finished with exit code 0
```

# **Outputs of Administrator User**

# Main Menu of Admin User

```
Run: main_menu x
▶ C:\Users\dell\AppData\Local\Programs\Python\Python39\python.exe "D:/Swarit/Class 12/Class 12 Computer Project/main_menu.py"
----- [ Welcome to S & T Book Shop Management System ] -----
↑ 1 -> Sign Up
↓ 2 -> Sign In
Enter your choice: 2
Enter Login ID: admin
Enter Password: admin
Successfully Logged In.
You are Administrator :)

Select your choice:
1 -> Add A New Book
2 -> Update A book Information
3 -> Remove A book
4 -> View Details of a book
5 -> View All Books
6 -> Place an Order
7 -> Cancel an Order
8 -> Return/Replace an Order
9 -> Update an Order Address
10 -> View All Orders
11 -> View Orders of a User(s)
12 -> View Orders by Date
13 -> Add a New User
14 -> Update A User's Credentials
15 -> Remove A User
16 -> View all Users
17 -> View Details of some users
18 -> Exit
```

# Adding a Book

```
Run: main_menu x
▶ 18 -> Exit
█ Enter your choice from (1 to 18): 1
█ Enter number of books you want to add: 1
█     Enter the book name: Computer Science With Python
█     Enter the first name of author: Sumita
█     Enter the last name of author: Arora
█     Enter the year of Release: 2020
█     Enter the category of book: Educational
█     Enter the Publisher's Name: Dhanpat Rai Publications
█     Enter the number of books in stock: 23
█     Enter number of pages of book: 690
█     Enter the number of words per page (average) approximate: 150
█     Enter the price of the book: 699
█ Successfully Added the book.
█ Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
```

# Updating a Book

```
▶ 1 Enter your choice from (1 to 18): 2
█ Enter number of books you want to update: 1
█ Enter the name of the book you want to update: Computer Science With Python
█     1. Update the book's year:
█     2. Update the book's stock:
█     3. Update the book's number of pages:
█     4. Update the book's price:
█     Enter your choice: 4
█     Enter the new price: 590
█     Successfully Updated
█ Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
```

## Deleting a Book

```
Enter your choice from (1 to 18): 3
Enter number of books you want to delete: 1
Enter the name of the book you want to delete: Computer Science With Python
Successfully Deleted.
Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
Process Successfully Done. Logged Out
Your active time was 30.177558183670044 seconds
```

```
Process finished with exit code 0
```

## Adding Users

```
Enter your choice from (1 to 18): 13
Enter number of user you want to add: 2
    Enter Login ID for the user: swarit_test1
    Enter Password of the user: swarit_test1
    Enter name of the user: Swarit
    Enter email of the user: swaritjain693@gmail.com
Validating Details, please wait.....
Registering
Successfully Added User 'swarit_test1'
    Enter Login ID for the user: swarit_test2
    Enter Password of the user: swarit_test2
    Enter name of the user: Swarit
    Enter email of the user: aayushia234@gmail.com
Validating Details, please wait.....
Registering
Successfully Added User 'swarit_test2'
Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
```

# Updating Users

```
Enter your choice from (1 to 18): 14
Enter number of user you want to update: 1
Enter the username of the user you want to update: swarit_test1
1 -> Update the user's username
2 -> Update the user's password
3 -> Update the user's name
4 -> Update the user's email address
Enter your choice (1 to 4): 3
Enter the new name of the swarit_test1: Swarit Jain test1
Successfully Updated
Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
```

# Removing User

```
Enter your choice from (1 to 18): 15
Enter number of accounts you want to delete: 2
Enter the username of the user you want to delete: swarit_test1
Successfully Deleted.
Enter the username of the user you want to delete: swarit_test2
Successfully Deleted.
Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
```

# View All Users

Enter your choice from (1 to 18): 16

User 1

ID : 1  
Username : admin  
Password : admin  
Name : admin  
E-mail Address : swaritjain123@gmail.com

User 2

ID : 2  
Username : testuser  
Password : testuser  
Name : test  
E-mail Address : tanishq.rc@gmail.com

User 3

ID : 3  
Username : sjain12sw  
Password : qwerty  
Name : Swasti Jain  
E-mail Address : swjain446@gmail.com

Go Back to Main Menu? (y -> Yes, n -> No and Logout) |

## View Details of a Particular User

```
Enter your choice from (1 to 18): 17
```

```
Enter the number of users of which you want details of: 1
```

```
Enter the username of the user: sjain12sw
```

```
ID: 3
```

```
Username: sjain12sw
```

```
Name: Swasti Jain
```

```
Email-Address: swjain446@gmail.com
```

```
Go Back to Main Menu? (y -> Yes, n -> No and Logout)
```

# Returning an Order

```
Run: main_menu x
▶ 🖊 Enter your choice from (1 to 18): 8
▀ ⓘ We have found 2 orders have been delivered to you
▀ ↑ List of the orders that have been delivered to you are as follows:-
    Book Name      : the namesake
    Order Date     : 2020-11-02 15:30:11
    Order ID       : 5
    Customer Name  : Swarit
    Location        : C-121, SBI Colony, Main Branch Campus, Opposite Kachahari, Varanasi, Varanasi, Uttar Pradesh, 221002, India
    Email          : swaritjain123@gmail.com
    Phone no        : 8004124465
    No of Books    : 2

    Book Name      : coraline
    Order Date     : 2020-11-01 15:30:11
    Order ID       : 4
    Customer Name  : Swarit
    Location        : C-121, SBI Colony, Main Branch Campus, Opposite Kachahari, Varanasi, Varanasi, Uttar Pradesh, 221002, India
    Email          : swaritjain123@gmail.com
    Phone no        : 8004124465
    No of Books    : 1

Do you want to return or replace any of the above orders (y/n) y
Enter the order ID for which you want to return or replace: 5
You have chosen order ID: 5
Few details of your order are as follows:
```

Book name: the namesake

Quantity: 2

Order date: 2020-11-02 15:30:11

By SNT.bookshop Return and Replace policies, you have below two options

1. Return

2. Replace

Enter the option number that you want to select: 1

You have entered under our Return policy

Are you sure you want to return this order? (y/n) y

Processing the Return

An email with all details is sent to you on your E-mail Address swaritjain123@gmail.com

Successfully requested for return to your order.

Our courier boy will come within the next day to collect the package after verifying its condition, be ready with the packed book

Money will be refund to your account within 2-3 days

Go Back to Main Menu? (y -> Yes, n -> No and Logout) y

Note: - Orders Shown here are test orders of 'admin' user which were inserted after the testuser (sjain12sw) placed his/her order. Two Orders displayed above are used one by one for return and replacement.

P.T.O.

# Email sent on Returning the order

The screenshot shows a Gmail inbox interface. On the left, there's a sidebar with icons for Compose, Inbox (which is selected), Starred, Snoozed, Sent, Drafts, and More. Below these are sections for Meet (New meeting, Join a meeting) and Hangouts. The main area displays an email message from "snt.bookshop@gmail.com" with the subject "Information of Returning Order on S & T Bookshop". The message body contains details about the return request, including the book name ("The Namesake"), order ID (5), recipient's name ("Swarit"), address ("C-121, SBI Colony, Main Branch Campus, Opposite Kachahari"), district ("Varanasi"), city ("Varanasi"), state ("Uttar Pradesh"), pincode ("221002"), country ("India"), phone number ("8004124465"), pieces ordered ("2"), and order date and time ("2020-11-02 15:30:11"). At the bottom of the email view are "Reply" and "Forward" buttons. To the right of the email, there's a user profile for "Swarit Jain" with the email "swaritjain123@gmail.com". The profile picture is a blue-toned image of a circuit board. Below the profile are links for "Manage your Google Account", "Add another account", and "Sign out". At the very bottom of the page are links for "Privacy Policy" and "Terms of Service".

Compose

Inbox

Starred

Snoozed

Sent

Drafts

More

Meet

New meeting

Join a meeting

Hangouts

Information of Returning Order on S & T Bookshop

snt.bookshop@gmail.com

to

Your request for returning your order has been accepted. Some details for it are as follows:

Book Name: The Namesake

Order ID: 5

Name: Swarit

Address: C-121, SBI Colony, Main Branch Campus, Opposite Kachahari

District: Varanasi

City: Varanasi

State: Uttar Pradesh

Pincode: 221002

Country: India

Phone: 8004124465

Pieces Ordered: 2

Order Date & Time: 2020-11-02 15:30:11

Reply Forward

Swarit Jain  
swaritjain123@gmail.com

Manage your Google Account

Add another account

Sign out

Privacy Policy • Terms of Service

P.T.O.

# Replacing an Order

Run: main\_menu x

```
▶ 🖊 Enter your choice from (1 to 18): 8
■ 🔍 We have found 1 orders have been delivered to you
List of the orders that have been delivered to you are as follows:-
    Book Name      : coraline
    Order Date     : 2020-11-01 15:30:11
    Order ID       : 4
    Customer Name  : Swarit
    Location        : C-121, SBI Colony, Main Branch Campus, Opposite Kachahari, Varanasi, Varanasi, Uttar Pradesh, 221002, India
    Email           : swaritjain123@gmail.com
    Phone no        : 8004124465
    No of Books     : 1
```

Do you want to return or replace any of the above orders (y/n) y

Enter the order ID for which you want to return or replace: 4

You have chosen order ID: 4

Few details of your order are as follows:

```
Book name: coraline
Quantity: 1
Order date: 2020-11-01 15:30:11
```

By SNT.bookshop Return and Replace policies, you have below two options

1. Return
2. Replace

Enter the option number that you want to select: 2

You have entered under our Replace policy.

Are you sure you want to replace for this order? (y/n) y

Successfully requested for replacement of your order.

An email with all details is sent to you on your E-mail Address swaritjain123@gmail.com

Our courier boy will come within the next day to collect the package after verifying its condition, be ready with the packed book

Replacement will be completed within 7 working days until your order has been delivered.

Go Back to Main Menu? (y -> Yes, n -> No and Logout) n

## Email Sent on Replacing the Order

G Information of Replacing Order on X + mail.google.com/mail/u/0/#inbox/ FMfcgxwKjTPjDzctmsPPPwqDckShCkdJ

YouTube Unacademy Smartkeeda - Leadin... Python File Handlin... BANKING CHRONI...

Gmail Search mail

Compose

Inbox Starred Snoozed Sent Drafts More Meet New meeting Join a meeting Hangouts

Information of Replacing Order on S & T Bookshop Inbox x

snt.bookshop@gmail.com to

Your request for replacing your order has been accepted. Some details for it are as follows:

Book Name: Coraline  
Order ID: 4  
Name: Swarit  
Address: C-121, SBI Colony, Main Branch Campus, Opposite Kachahari  
District: Varanasi  
City: Varanasi  
State: Uttar Pradesh  
Pincode: 221002  
Country: India  
Phone: 8004124465  
Pieces Ordered: 1  
Order Date & Time: 2020-11-01 15:30:11  
Delivery will be provided within 7 working days.  
Sorry for the inconvenience :)

Swarit Jain  
swaritjain123@gmail.com

Manage your Google Account

Add another account

Sign out

Privacy Policy • Terms of Service

# Viewing all Orders for Replacement

```
Enter your choice from (1 to 18): 10
1 -> View all New Orders Placed
2 -> View all Orders for Return
3 -> View all Orders for Replacement
4 -> View all Orders including all categories
Enter your choice (1 to 4): 3

Order ID      : 8
Book Name     : Coraline
Pieces        : 1
Customer Name : Swarit
Address       : C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
District      : Varanasi
City          : Varanasi
State         : Uttar Pradesh
Pincode       : 221002
Country       : India
E-mail Address : swaritjain123@gmail.com
Phone         : 8004124465
Order Date    : 2020-11-13 20:13:48
Username      : admin
```

Go Back to Main Menu? (y -> Yes, n -> No and Logout) |

P.T.O.

## Viewing all Orders for Return

```
Enter your choice from (1 to 18): 10
1 -> View all New Orders Placed
2 -> View all Orders for Return
3 -> View all Orders for Replacement
4 -> View all Orders including all categories
Enter your choice (1 to 4): 2
Order ID      : 5
Book Name     : The Namesake
Pieces        : 2
Customer Name : Swarit
Address       : C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
District      : Varanasi
City          : Varanasi
State         : Uttar Pradesh
Pincode       : 221002
Country       : India
E-mail Address: swaritjain123@gmail.com
Phone         : 8004124465
Order Date    : 2020-11-02 15:30:11
Username      : admin
```

# Viewing all Orders of a particular User (here, admin)

```
Run: main_menu x
↻ ↎ Enter your choice from (1 to 18): 11
☒ ↎ Enter the customer's username: admin
☰ ↑ Order ID      : 8
☰ ↓ Book Name     : Coraline
☷ ↎ Pieces        : 1
☷ ↎ Customer Name : Swarit
☷ ↎ Address       : C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
☷ ↎ District      : Varanasi
☷ ↎ City          : Varanasi
☷ ↎ State         : Uttar Pradesh
☷ ↎ Pincode       : 221002
☷ ↎ Country        : India
☷ ↎ E-mail Address : swaritjain123@gmail.com
☷ ↎ Phone          : 8004124465
☷ ↎ Order Date    : 2020-11-13 20:13:48
☷ ↎ Order Type    : Replace

☰ ↑ Order ID      : 5
☰ ↓ Book Name     : The Namesake
☷ ↎ Pieces        : 2
☷ ↎ Customer Name : Swarit
☷ ↎ Address       : C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
☷ ↎ District      : Varanasi
☷ ↎ City          : Varanasi
☷ ↎ State         : Uttar Pradesh
☷ ↎ Pincode       : 221002
☷ ↎ Country        : India
☷ ↎ E-mail Address : swaritjain123@gmail.com
☷ ↎ Phone          : 8004124465
☷ ↎ Order Date    : 2020-11-02 15:30:11
☷ ↎ Order Type    : Return
```

Order ID : 3  
Book Name : American Gods  
Pieces : 5  
Customer Name : Swarit  
Address : C-121, SBI Colony, Main Branch Campus, Opposite Kachahari  
District : Varanasi  
City : Varanasi  
State : Uttar Pradesh  
Pincode : 221002  
Country : India  
E-mail Address : swaritjain123@gmail.com  
Phone : 8004124465  
Order Date : 2020-10-26 15:30:11  
Order Type : New  
Order Status : Delivered on 2020-11-02

## View all Orders by date

Enter your choice from (1 to 18): 12

Enter date (YYYY-MM-DD): 2020-10-26

Order ID	:	3
Book Name	:	American Gods
Pieces	:	5
Customer Name	:	Swarit
Address	:	C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
District	:	Varanasi
City	:	Varanasi
State	:	Uttar Pradesh
Pincode	:	221002
Country	:	India
E-mail Address	:	swaritjain123@gmail.com
Phone	:	8004124465
Order Date	:	2020-10-26 15:30:11
Username	:	admin
Order Type	:	New
Order Status	:	Delivered on 2020-11-02

Go Back to Main Menu? (y -> Yes, n -> No and Logout) n

## Updating an Order Address

```
Enter your choice from (1 to 18): 9
```

```
Only 2 order(s) is/are eligible for update of address.
```

```
Enter the number of orders's addresses you want to update: 1
```

```
Enter the Order ID to update the order: 1
```

```
    Enter the new address for order ID 1: 1234, Max Colony, Rathyatra
```

```
    Enter the new city for order ID 1: Varanasi
```

```
    Enter the new pincode for order ID 1: 221005
```

```
    Enter the new district for Order ID1: Varanasi
```

```
    Enter the new State/UT for Order ID1: Uttar Pradesh
```

```
An email with all details is sent to you on your E-mail Address swjain446@gmail.com
```

```
Successfully Updated
```

```
regarding order update for new location on email address swjain446@gmail.com
```

```
Go Back to Main Menu? (y -> Yes, n -> No and Logout) |
```

Note: - Here Order ID '1' is of the testuser(sjain12sw).

P.T.O

# Email Sent on Order Address being Updated by Admin

The screenshot shows a Gmail inbox with a blurred background. On the left, there's a sidebar with icons for Compose, Inbox (2), Starred, Snoozed, Sent, Drafts, STUDY MATERIALS, More, Meet, New meeting, and Join a meeting. The main area displays an email message:

**Information of Order Update on S & T Book Shop Management System** Inbox x

**snt.bookshop@gmail.com** to

Your Order has been updated successfully  
New Location of the order is 1234, Max Colony, Rathyatra, Varanasi, 221005, Uttar Pradesh :)

Reply Forward

On the right side, there's a sidebar for Google Account settings:

- Google Account**  
Swasti Jain  
swjain446@gmail.com
- Swasti Jain  
swjain446@gmail.com
- [Manage your Google Account](#)
- Swarit Jain  
snt.bookshop@gmail.com Default
- swaritjain123@gmail.co... Signed out  
swaritjain123@gmail.com
- Add another account

P.T.O.

## Cancelling an Order

```
Enter your choice from (1 to 18): 7
Only 2 order(s) is/are eligible to cancel.
Enter number of orders you want to cancel (not more than 2): 1
    Enter the Order ID: 1
An email with all details is sent to you on your E-mail Address swjain446@gmail.com
Successfully cancelled your order.
Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
```

Note: - Here, order is being cancelled by admin and email is sent to the user's email address on which the order was placed.

P.T.O.

# Email Sent on Order being Cancelled by Admin

Screenshot of a Gmail inbox showing an email regarding the cancellation of an order.

**Inbox** (2)

**Compose**

**Regarding Cancellation of Order on S & T Bookshop** Inbox x

**snt.bookshop@gmail.com** to ▾

Your Order has been cancelled as per your request. Please review the product on our website.  
Book Name: Coraline  
Publication: Bloomsbury Children's Books (30 August 2016)  
Author's Name: Neil Gaiman  
Order ID: 1  
Name: Swasti Jain  
Address: 1234, Max Colony, Rathyatra  
District: Varanasi  
City: Varanasi  
State: Uttar Pradesh  
Pincode: 221005  
Country: India  
Phone: 9876543210  
Pieces Ordered: 2  
Order Date & Time: 2020-11-13 18:14:02

**Reply** **Forward**

**Swasti Jain** swjain446@gmail.com

**Manage your Google Account**

**Swarit Jain** snt.bookshop@gmail.com Default

**swaritjain123@gmail.co...** swaritjain123@gmail.com Signed out

**Add another account**

**Sign out of all accounts**

Privacy Policy • Terms of Service

## Exiting from Administrator Block

Enter your choice from (1 to 18): 18

Successfully Logged Out. Your active time was 27.018373727798462 seconds

# Error Handling

## Already Registered Email Address can't be used by any other user.

```
Run: main_menu x
C:\Users\dell\AppData\Local\Programs\Python\Python39\python.exe "D:/Swarit/Class 12/Class 12 Computer Project/main_menu.py"
----- [ Welcome to S & T Book Shop Management System ] -----
1 -> Sign Up
2 -> Sign In
Enter your choice: 1
Create Username (Login ID) : sam1234
Create Password: sam1234
Enter your name: Tanishq Singh
Enter your email address: tanishq.rc@gmail.com
E-mail Address Already Registered.
```

## Email Validation (can't insert invalid email(s))

```
Run: main_menu x
C:\Users\dell\AppData\Local\Programs\Python\Python39\python.exe "D:/Swarit/Class 12/Class 12 Computer Project/main_menu.py"
----- [ Welcome to S & T Book Shop Management System ] -----
1 -> Sign Up
2 -> Sign In
Enter your choice: 1
Create Username (Login ID) : asdfqwe123
Create Password: asdfqwe123
Enter your name: Tom Hanks
Enter your email address: jainsw123@gmail.com
Validating Details, please wait.....
Validation for 'jainsw123@gmail.com' failed: Email address undeliverable:
```

## Admin account can't be deleted.

```
13 -> Add a New User
14 -> Update A User's Credentials
15 -> Remove A User
16 -> View all Users
17 -> View Details of some users
18 -> Exit

Enter your choice from (1 to 18): 15
Enter number of accounts you want to delete: 1
Enter the username of the user you want to delete: admin
Can't Delete user. The specified username is Administrator.
Go Back to Main Menu? (y -> Yes, n -> No and Logout)
```

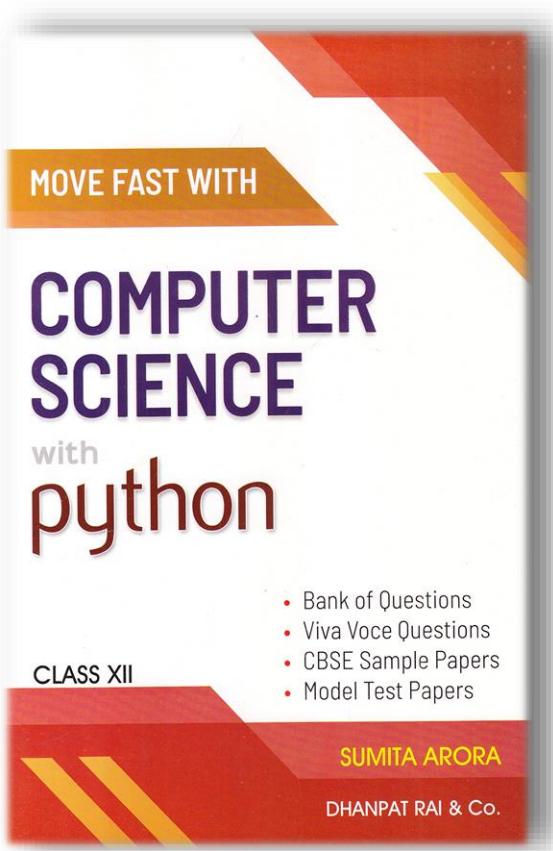
# Can't login by invalid credentials

```
Run: main_menu x
C:\Users\dell\AppData\Local\Programs\Python\Python39\python.exe "D:/Swarit/Class 12/Class 12 Computer Project/main_menu.py"
----- [ Welcome to S & T Book Shop Management System ] -----
1 -> Sign Up
2 -> Sign In
Enter your choice: 2
Enter Login ID: qaxzsw
Enter Password: qaxzsw
Wrong Credentials
Enter login details again? (y -> Yes, n -> No and exit): y
Enter Login ID: sjain12sw
Enter Password: qwerty
Successfully Logged In.
1-> Place an Order
2-> Cancel an Order
3-> Return or Replace an Order
4-> Update an Order Address
5-> View all your Orders
6-> Exit
Enter your choice from (1 to 6): |
```

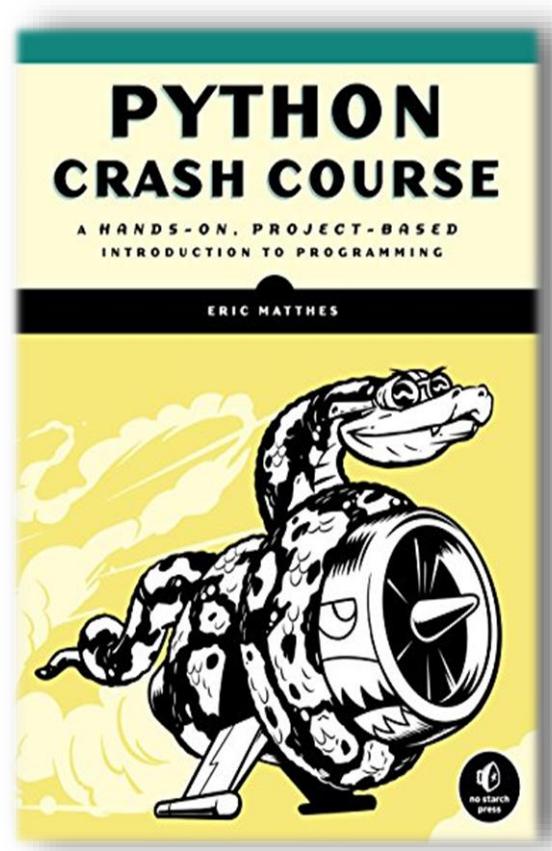
# Bibliography

## References

1. <https://www.w3schools.com>
2. <https://www.wikipedia.org>
3. <https://www.stackoverflow.com>
4. <https://www.geekforgeeks.org>



**Computer Science with Python – Sumita Arora**  
**Class 12**



**Python Crash Course – Eric Matthes**