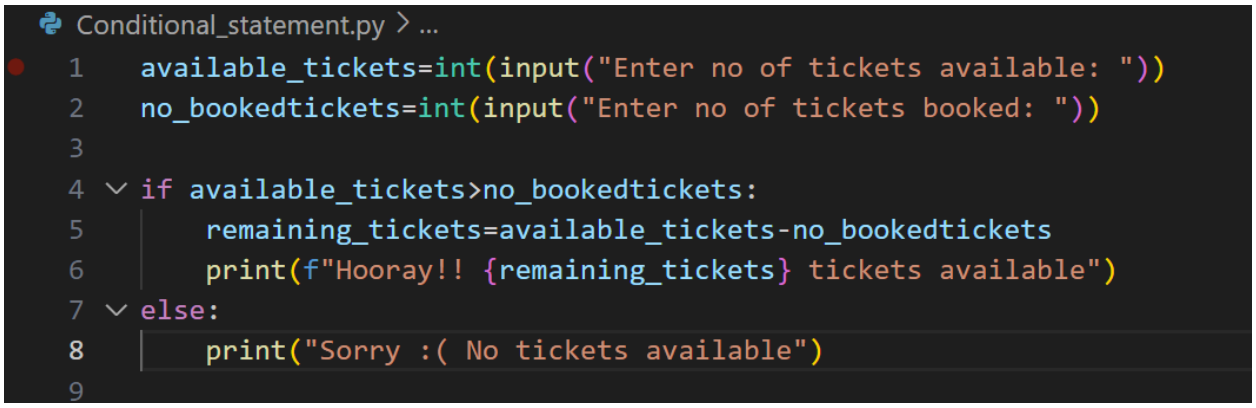
PYTHON ASSIGNMENT

TICKET BOOKING SYSTEM

Name:Nidya Thirshala

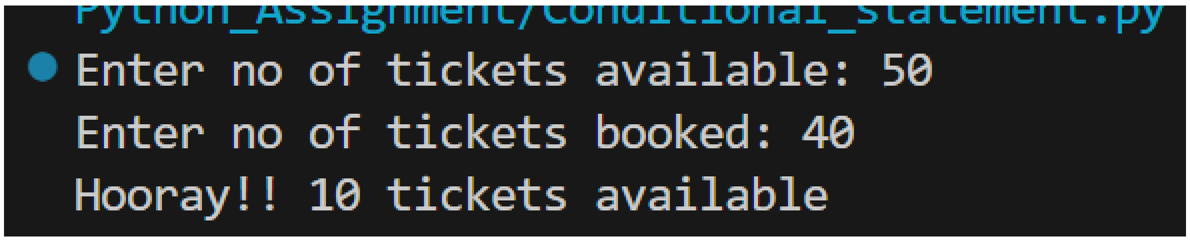
Task 1: Conditional Statements

Tasks:
1. Write a program that takes the availableTicket and noOfBookingTicket as input.
2. Use conditional statements (if-else) to determine if the ticket is available or not.
3. Display an appropriate message based on ticket availability.



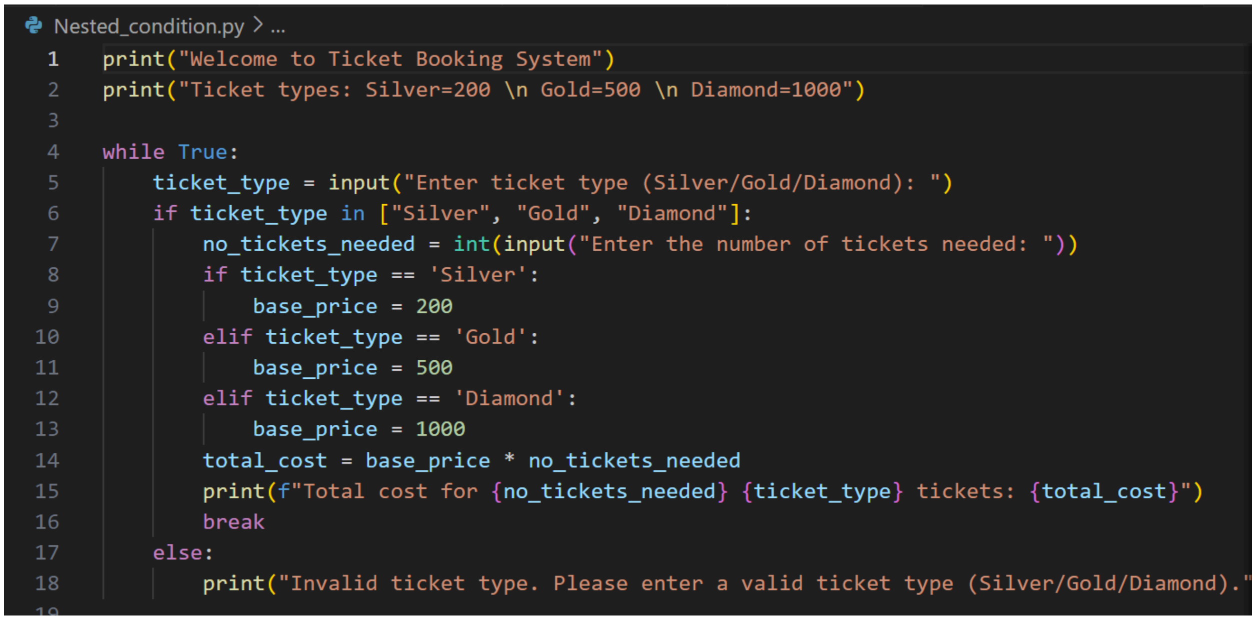
Conditional_statement.py > ...

```python
available_tickets=int(input("Enter no of tickets available: "))
no_bookedtickets=int(input("Enter no of tickets booked: "))

if available_tickets>no_bookedtickets:
    remaining_tickets=available_tickets-no_bookedtickets
    print(f"Hooray!! {remaining_tickets} tickets available")
else:
    print("Sorry :( No tickets available")
```



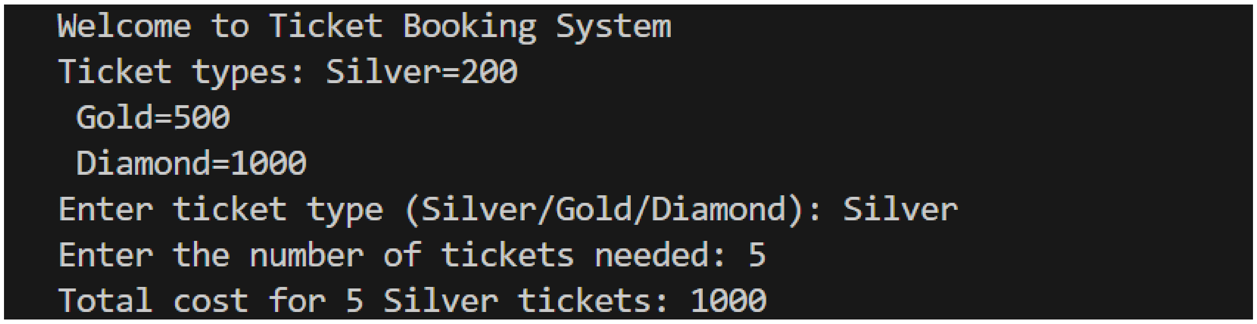
```
Python_Assignment/Conditional_statement.py
Enter no of tickets available: 50
Enter no of tickets booked: 40
Hooray!! 10 tickets available
```

Task 2,3: Nested Conditional Statements,Looping

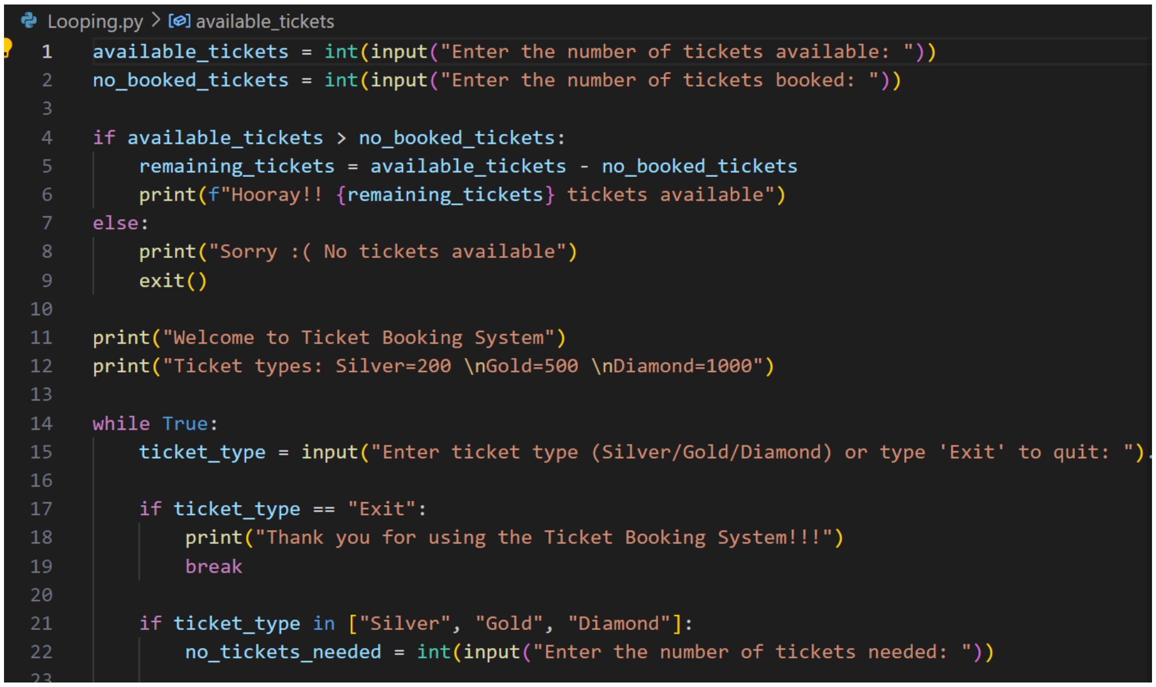
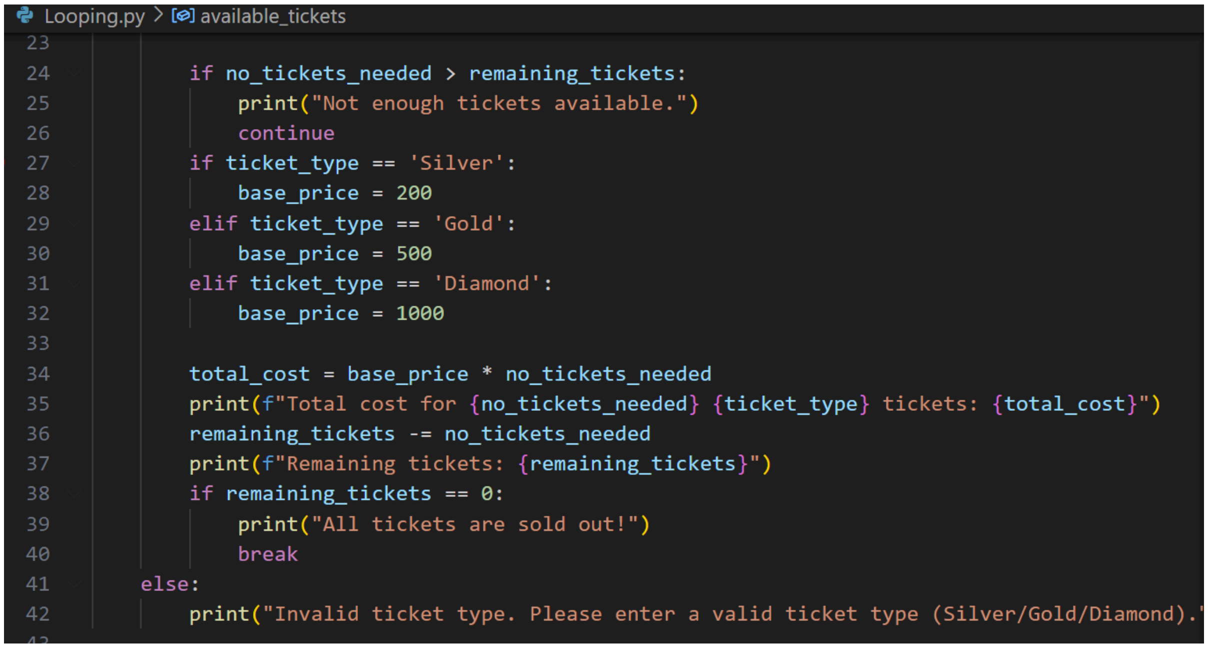
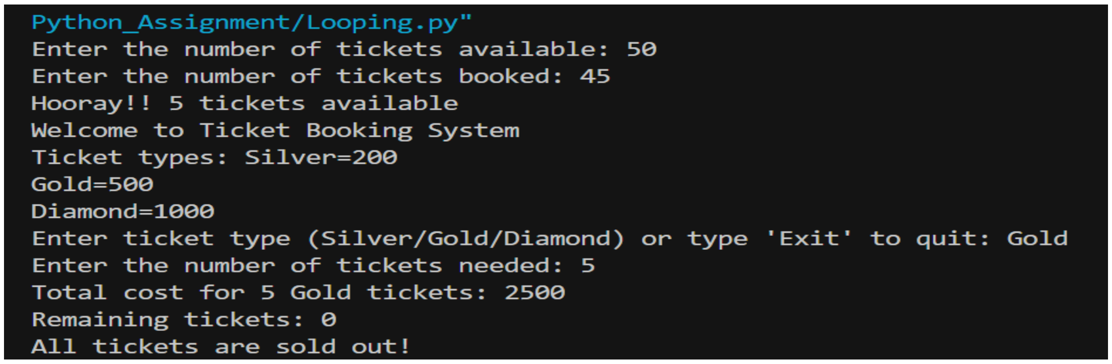


Nested_condition.py > ...

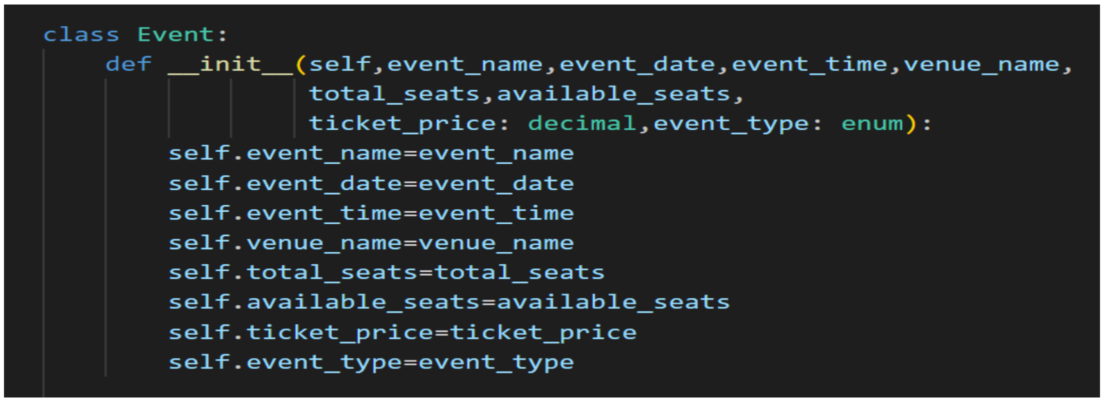
```python
print("Welcome to Ticket Booking System")
print("Ticket types: Silver=200 \n Gold=500 \n Diamond=1000")

while True:
    ticket_type = input("Enter ticket type (Silver/Gold/Diamond): ")
    if ticket_type in ["Silver", "Gold", "Diamond"]:
        no_tickets_needed = int(input("Enter the number of tickets needed: "))
        if ticket_type == 'Silver':
            base_price = 200
        elif ticket_type == 'Gold':
            base_price = 500
        elif ticket_type == 'Diamond':
            base_price = 1000
        total_cost = base_price * no_tickets_needed
        print(f"Total cost for {no_tickets_needed} {ticket_type} tickets: {total_cost}")
        break
    else:
        print("Invalid ticket type. Please enter a valid ticket type (Silver/Gold/Diamond)."
```



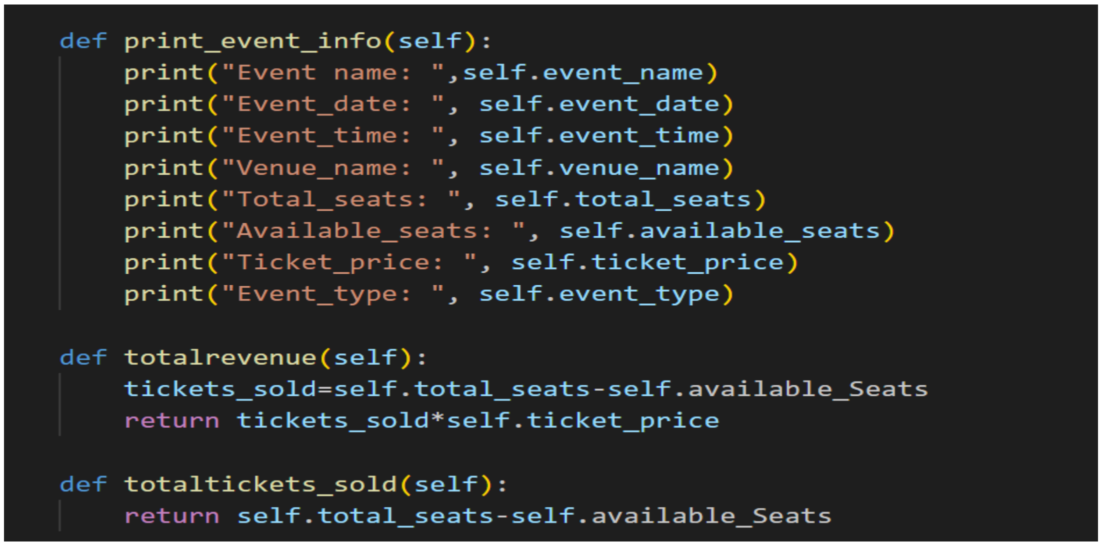
```
Welcome to Ticket Booking System
Ticket types: Silver=200
 Gold=500
 Diamond=1000
Enter ticket type (Silver/Gold/Diamond): Silver
Enter the number of tickets needed: 5
Total cost for 5 Silver tickets: 1000
```


Looping.py > available_tickets
```python
available_tickets = int(input("Enter the number of tickets available: "))
no_booked_tickets = int(input("Enter the number of tickets booked: "))

if available_tickets > no_booked_tickets:
    remaining_tickets = available_tickets - no_booked_tickets
    print(f"Hooray!! {remaining_tickets} tickets available")
else:
    print("Sorry :( No tickets available")
    exit()

print("Welcome to Ticket Booking System")
print("Ticket types: Silver=200 \nGold=500 \nDiamond=1000")

while True:
    ticket_type = input("Enter ticket type (Silver/Gold/Diamond) or type 'Exit' to quit: ")

    if ticket_type == "Exit":
        print("Thank you for using the Ticket Booking System!!!")
        break

    if ticket_type in ["Silver", "Gold", "Diamond"]:
        no_tickets_needed = int(input("Enter the number of tickets needed: "))
```

Looping.py > available_tickets
```python

            if no_tickets_needed > remaining_tickets:
                print("Not enough tickets available.")
                continue
            if ticket_type == 'Silver':
                base_price = 200
            elif ticket_type == 'Gold':
                base_price = 500
            elif ticket_type == 'Diamond':
                base_price = 1000

            total_cost = base_price * no_tickets_needed
            print(f"Total cost for {no_tickets_needed} {ticket_type} tickets: {total_cost}")
            remaining_tickets -= no_tickets_needed
            print(f"Remaining tickets: {remaining_tickets}")
            if remaining_tickets == 0:
                print("All tickets are sold out!")
                break
        else:
            print("Invalid ticket type. Please enter a valid ticket type (Silver/Gold/Diamond).'
```

```
Python_Assignment/Looping.py"
Enter the number of tickets available: 50
Enter the number of tickets booked: 45
Hooray!! 5 tickets available
Welcome to Ticket Booking System
Ticket types: Silver=200
Gold=500
Diamond=1000
Enter ticket type (Silver/Gold/Diamond) or type 'Exit' to quit: Gold
Enter the number of tickets needed: 5
Total cost for 5 Gold tickets: 2500
Remaining tickets: 0
All tickets are sold out!
```

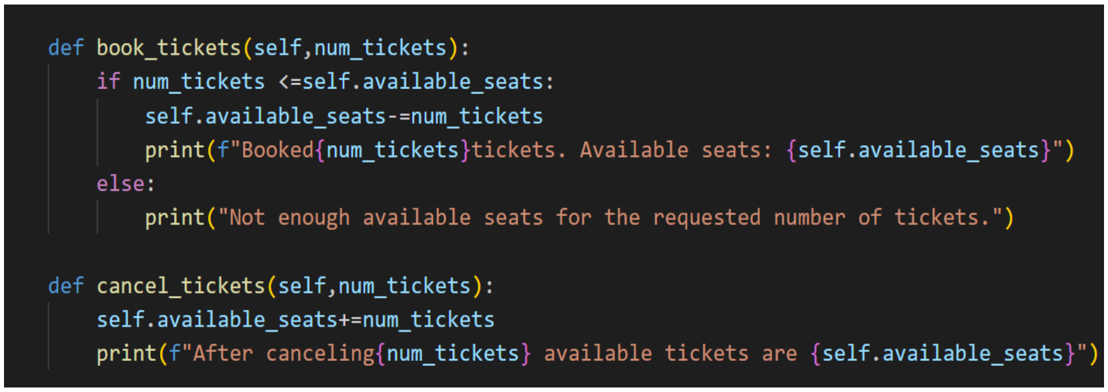
Task 4: Class & Object
Event class:



```python
class Event:
    def __init__(self,event_name,event_date,event_time,venue_name,
                 total_seats,available_seats,
                 ticket_price: decimal,event_type: enum):
        self.event_name=event_name
        self.event_date=event_date
        self.event_time=event_time
        self.venue_name=venue_name
        self.total_seats=total_seats
        self.available_seats=available_seats
        self.ticket_price=ticket_price
        self.event_type=event_type
```

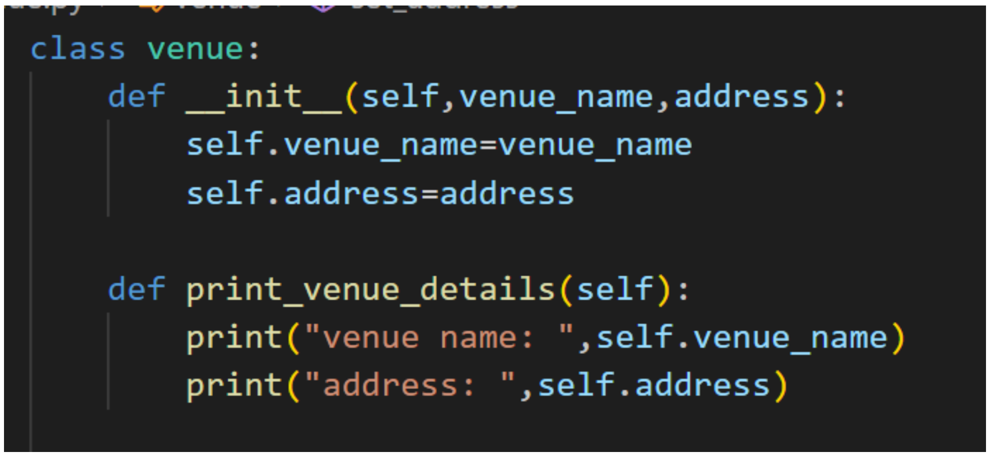


```python
    def print_event_info(self):
        print("Event name: ",self.event_name)
        print("Event_date: ", self.event_date)
        print("Event_time: ", self.event_time)
        print("Venue_name: ", self.venue_name)
        print("Total_seats: ", self.total_seats)
        print("Available_seats: ", self.available_seats)
        print("Ticket_price: ", self.ticket_price)
        print("Event_type: ", self.event_type)

    def totalrevenue(self):
        tickets_sold=self.total_seats-self.available_Seats
        return tickets_sold*self.ticket_price

    def totaltickets_sold(self):
        return self.total_seats-self.available_Seats
```



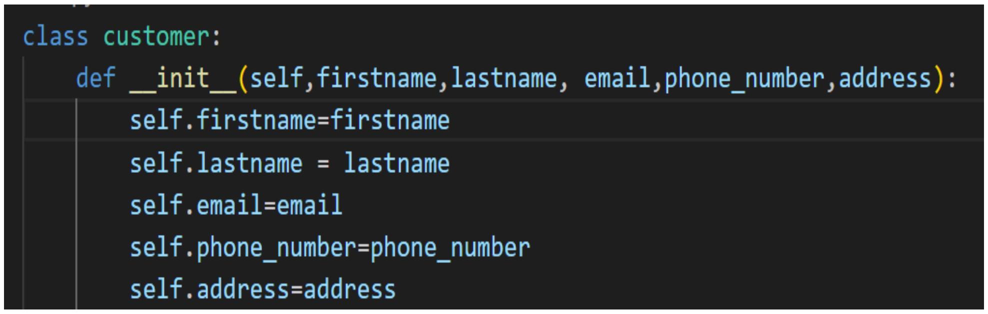
```python
    def book_tickets(self,num_tickets):
        if num_tickets <=self.available_seats:
            self.available_seats-=num_tickets
            print(f"Booked{num_tickets}tickets. Available seats: {self.available_seats}")
        else:
            print("Not enough available seats for the requested number of tickets.")

    def cancel_tickets(self,num_tickets):
        self.available_seats+=num_tickets
        print(f"After canceling{num_tickets} available tickets are {self.available_seats}")
```

Venue class:



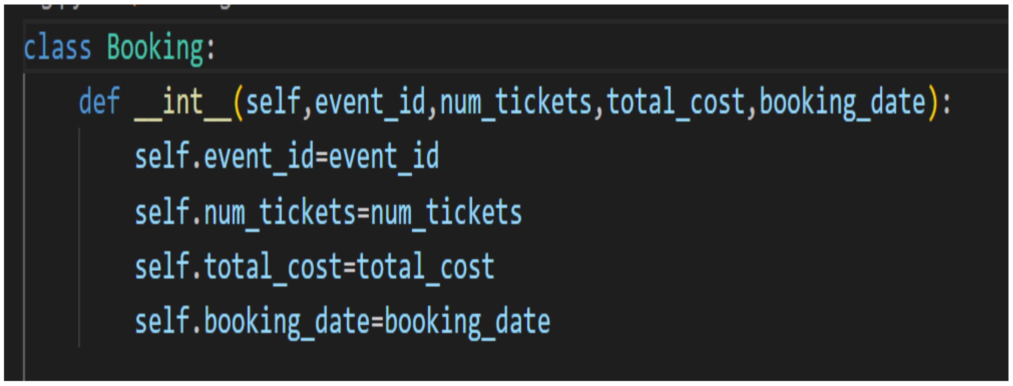
```python
class venue:
    def __init__(self,venue_name,address):
        self.venue_name=venue_name
        self.address=address

    def print_venue_details(self):
        print("venue name: ",self.venue_name)
        print("address: ",self.address)
```

Customer class:

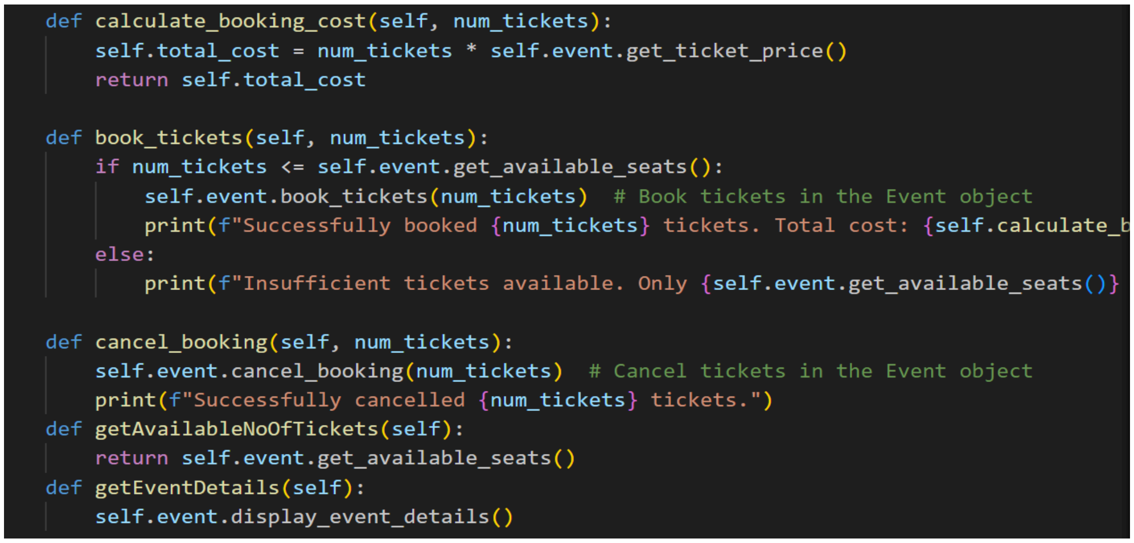


```python
class customer:
    def __init__(self,firstname,lastname, email,phone_number,address):
        self.firstname=firstname
        self.lastname = lastname
        self.email=email
        self.phone_number=phone_number
        self.address=address
```

Booking class:



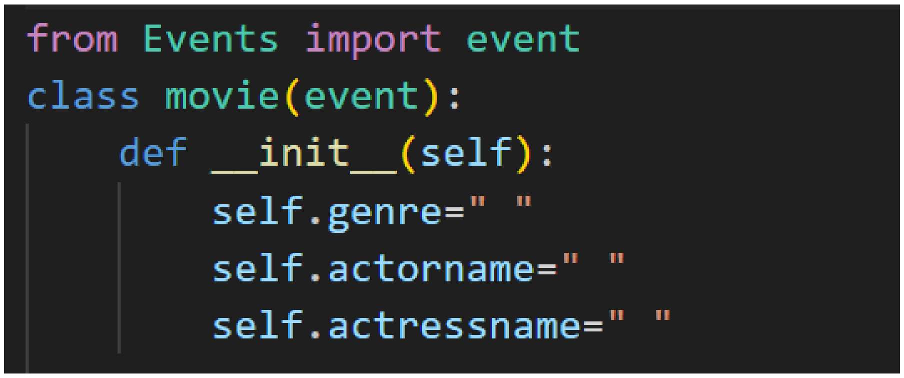
```python
class Booking:
    def _int_(self,event_id,num_tickets,total_cost,booking_date):
        self.event_id=event_id
        self.num_tickets=num_tickets
        self.total_cost=total_cost
        self.booking_date=booking_date
```



```python
def calculate_booking_cost(self, num_tickets):
    self.total_cost = num_tickets * self.event.get_ticket_price()
    return self.total_cost

def book_tickets(self, num_tickets):
    if num_tickets <= self.event.get_available_seats():
        self.event.book_tickets(num_tickets)  # Book tickets in the Event object
        print(f"Successfully booked {num_tickets} tickets. Total cost: {self.calculate_b
    else:
        print(f"Insufficient tickets available. Only {self.event.get_available_seats()}

def cancel_booking(self, num_tickets):
    self.event.cancel_booking(num_tickets)  # Cancel tickets in the Event object
    print(f"Successfully cancelled {num_tickets} tickets.")
def getAvailableNoOfTickets(self):
    return self.event.get_available_seats()
def getEventDetails(self):
    self.event.display_event_details()
```

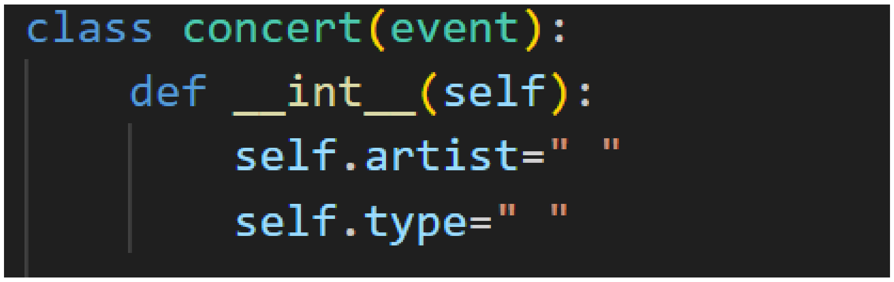
Task 5: Inheritance and polymorphism

Subclass:Movie



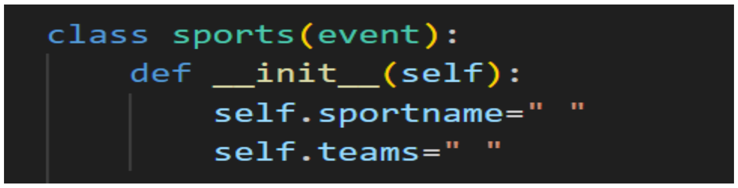
```python
from Events import event
class movie(event):
    def __init__(self):
        self.genre=" "
        self.actorname=" "
        self.actressname=" "
```

Subclass:Concert



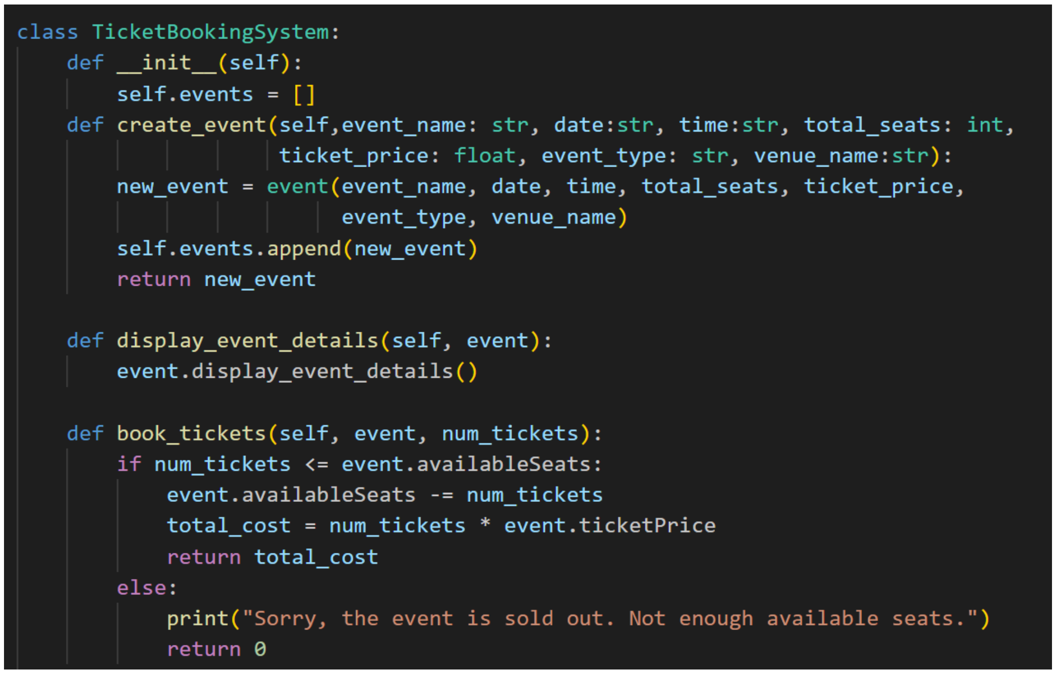
```python
class concert(event):
    def __int__(self):
        self.artist=" "
        self.type=" "
```

Subclass:Sport



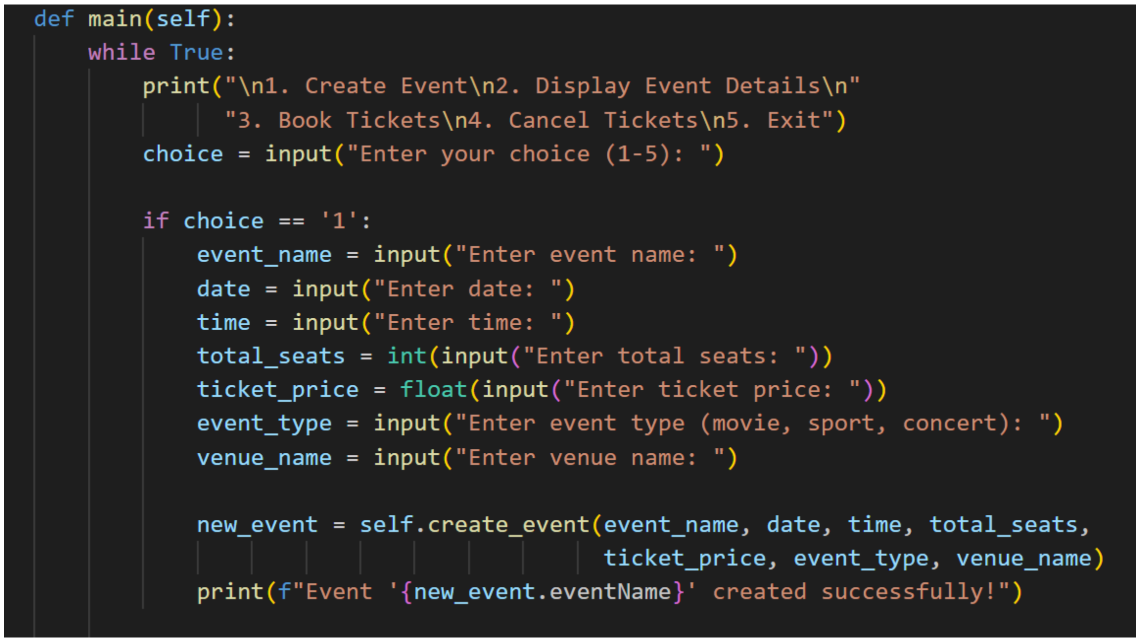
```python
class sports(event):
    def __init__(self):
        self.sportname=" "
        self.teams=" "
```

TicketBookingSystem:

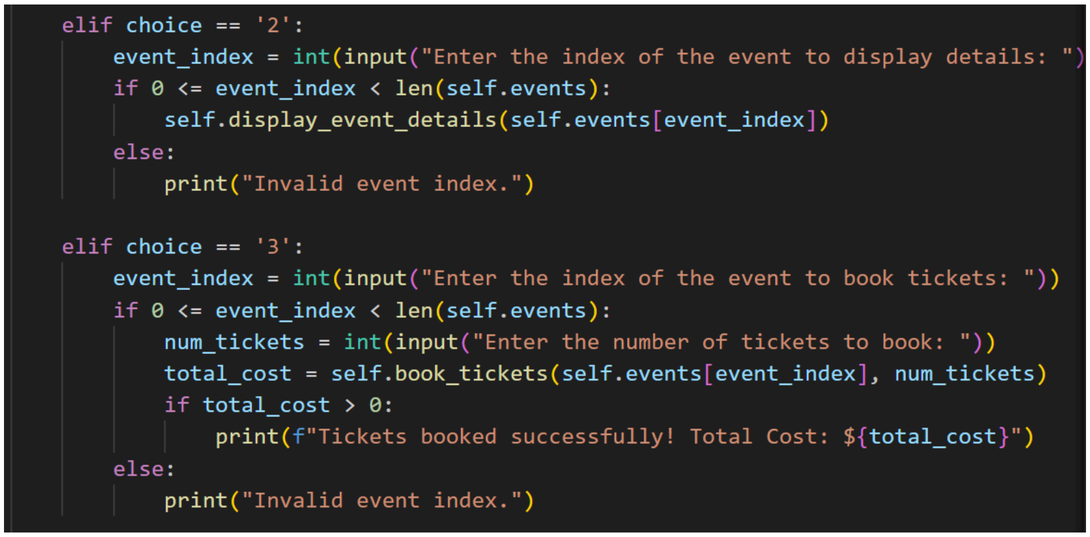


```
class TicketBookingSystem:
    def __init__(self):
        self.events = []
    def create_event(self,event_name: str, date:str, time:str, total_seats: int,
                     ticket_price: float, event_type: str, venue_name:str):
        new_event = event(event_name, date, time, total_seats, ticket_price,
                          event_type, venue_name)
        self.events.append(new_event)
        return new_event

    def display_event_details(self, event):
        event.display_event_details()

    def book_tickets(self, event, num_tickets):
        if num_tickets <= event.availableSeats:
            event.availableSeats -= num_tickets
            total_cost = num_tickets * event.ticketPrice
            return total_cost
        else:
            print("Sorry, the event is sold out. Not enough available seats.")
            return 0
```

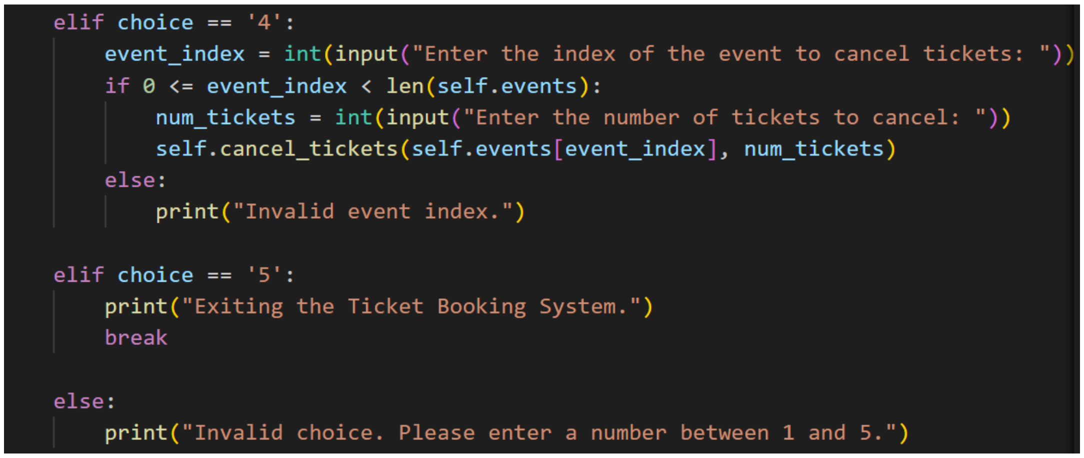
Main()



```
def main(self):
    while True:
        print("\n1. Create Event\n2. Display Event Details\n"
              "3. Book Tickets\n4. Cancel Tickets\n5. Exit")
        choice = input("Enter your choice (1-5): ")

        if choice == '1':
            event_name = input("Enter event name: ")
            date = input("Enter date: ")
            time = input("Enter time: ")
            total_seats = int(input("Enter total seats: "))
            ticket_price = float(input("Enter ticket price: "))
            event_type = input("Enter event type (movie, sport, concert): ")
            venue_name = input("Enter venue name: ")

            new_event = self.create_event(event_name, date, time, total_seats,
                                          ticket_price, event_type, venue_name)
            print(f"Event '{new_event.eventName}' created successfully!")
```

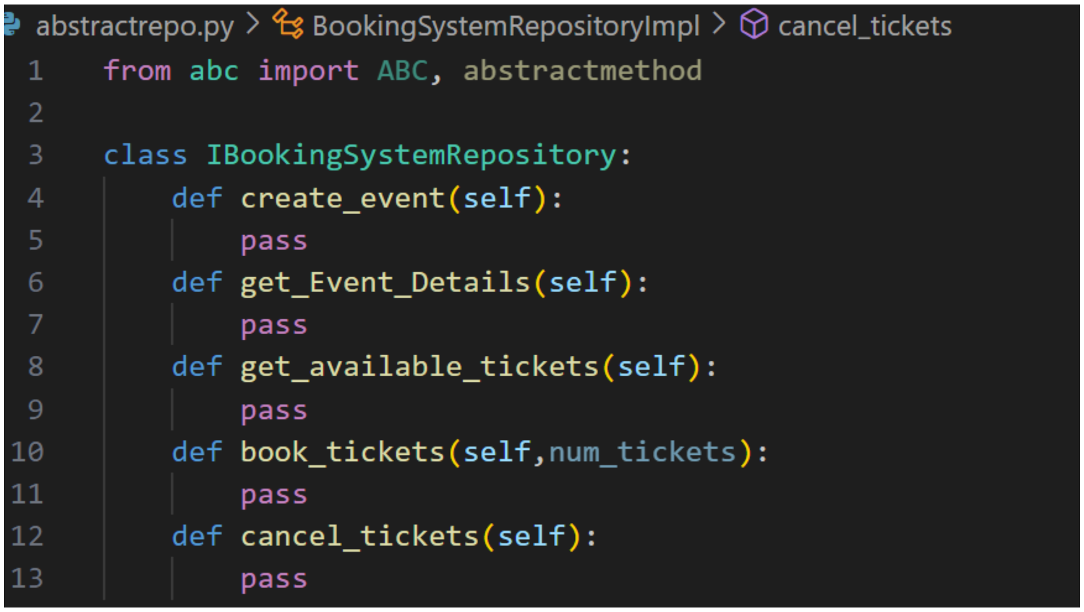


```
elif choice == '2':
    event_index = int(input("Enter the index of the event to display details: ")
    if 0 <= event_index < len(self.events):
        self.display_event_details(self.events[event_index])
    else:
        print("Invalid event index.")

elif choice == '3':
    event_index = int(input("Enter the index of the event to book tickets: "))
    if 0 <= event_index < len(self.events):
        num_tickets = int(input("Enter the number of tickets to book: "))
        total_cost = self.book_tickets(self.events[event_index], num_tickets)
        if total_cost > 0:
            print(f"Tickets booked successfully! Total Cost: ${total_cost}")
    else:
        print("Invalid event index.")
```



```
elif choice == '4':
    event_index = int(input("Enter the index of the event to cancel tickets: "))
    if 0 <= event_index < len(self.events):
        num_tickets = int(input("Enter the number of tickets to cancel: "))
        self.cancel_tickets(self.events[event_index], num_tickets)
    else:
        print("Invalid event index.")

elif choice == '5':
    print("Exiting the Ticket Booking System.")
    break

else:
    print("Invalid choice. Please enter a number between 1 and 5.")
```

Task 6: Abstraction



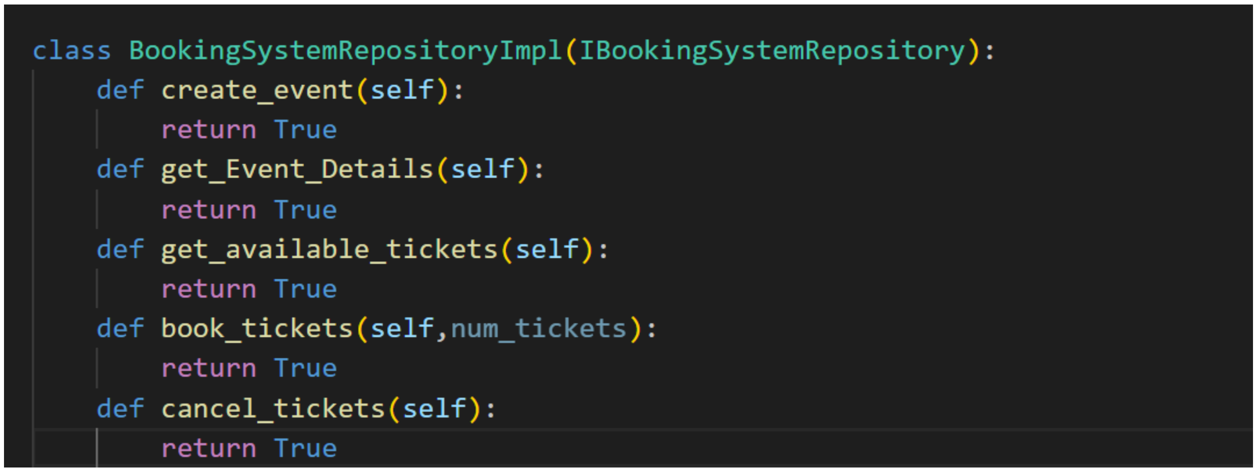
abstractrepo.py > BookingSystemRepositoryImpl > cancel_tickets

```
from abc import ABC, abstractmethod

class IBookingSystemRepository:
    def create_event(self):
        pass
    def get_Event_Details(self):
        pass
    def get_available_tickets(self):
        pass
    def book_tickets(self,num_tickets):
        pass
    def cancel_tickets(self):
        pass
```

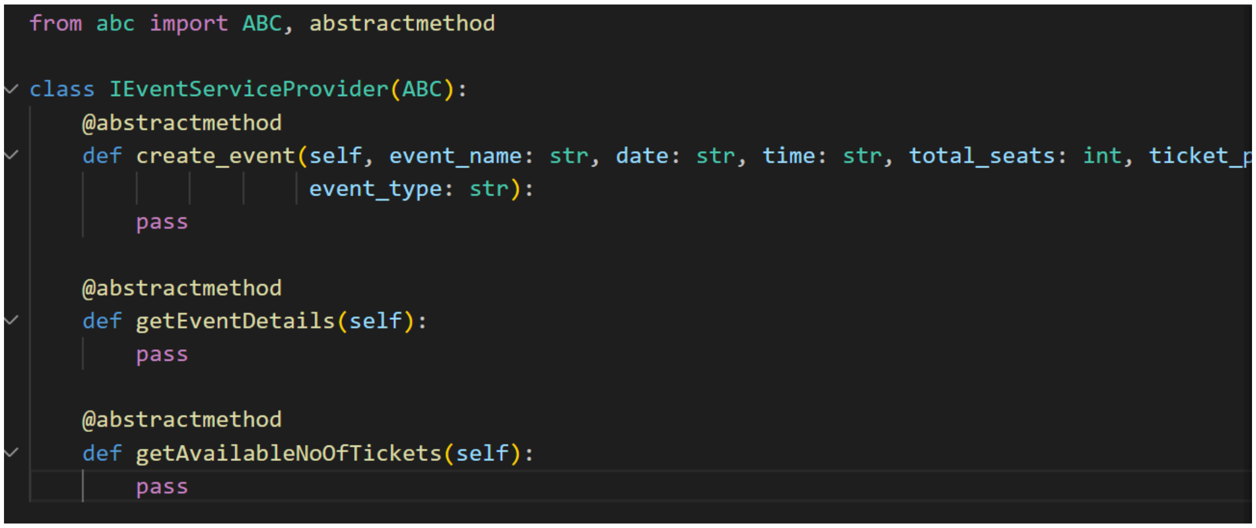
Task 7: Has A Relation / Association

The accessors and mutators are assigned to all the parent class and child class.

Task 8: Interface/abstract class, and Single Inheritance, static variable

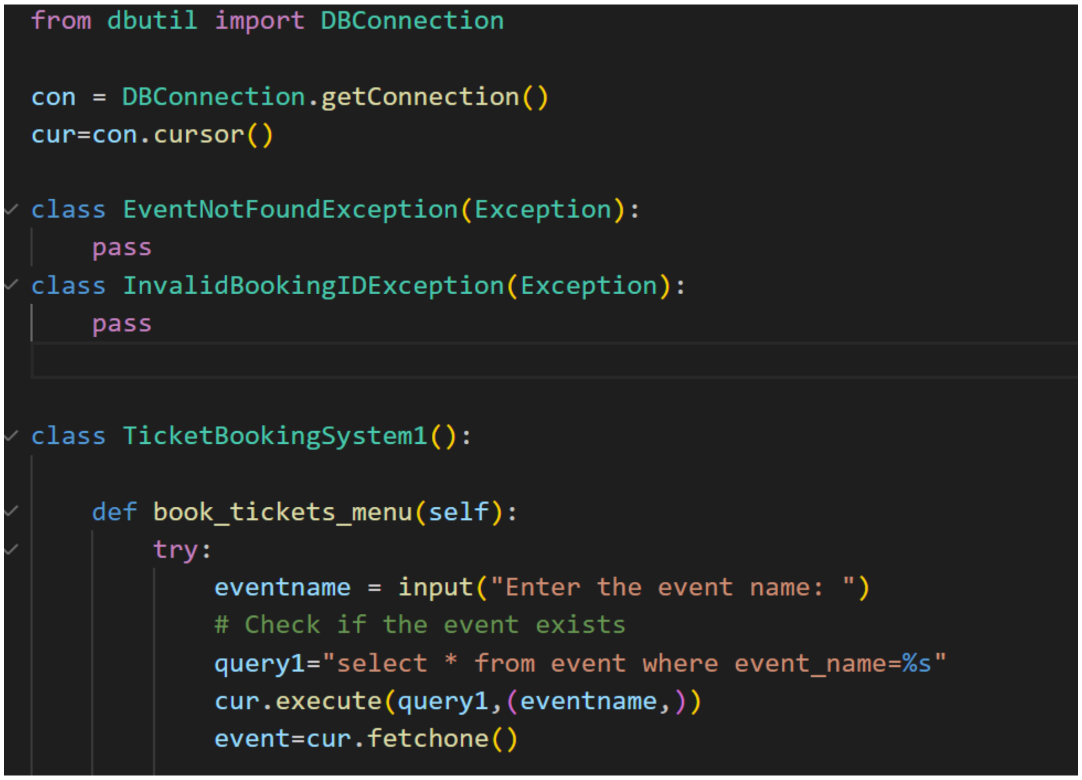


```python
class BookingSystemRepositoryImpl(IBookingSystemRepository):
    def create_event(self):
        return True
    def get_Event_Details(self):
        return True
    def get_available_tickets(self):
        return True
    def book_tickets(self,num_tickets):
        return True
    def cancel_tickets(self):
        return True
```

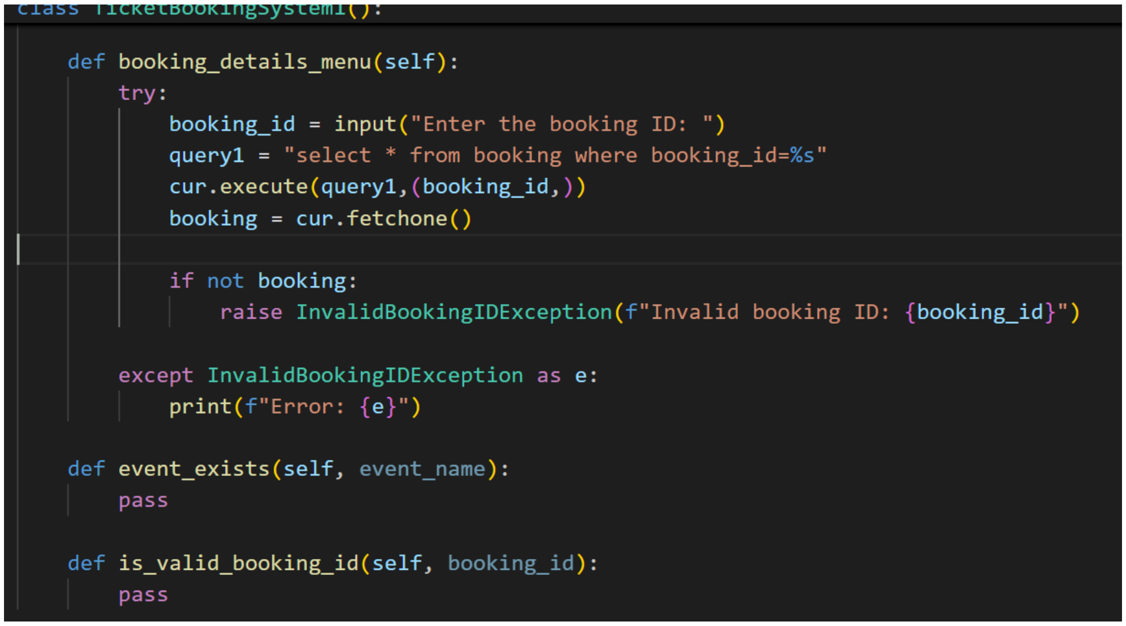


```python
from abc import ABC, abstractmethod

class IEventServiceProvider(ABC):
    @abstractmethod
    def create_event(self, event_name: str, date: str, time: str, total_seats: int, ticket_p
                    event_type: str):
        pass

    @abstractmethod
    def getEventDetails(self):
        pass

    @abstractmethod
    def getAvailableNoOfTickets(self):
        pass
```

Task 9:Exception Handling



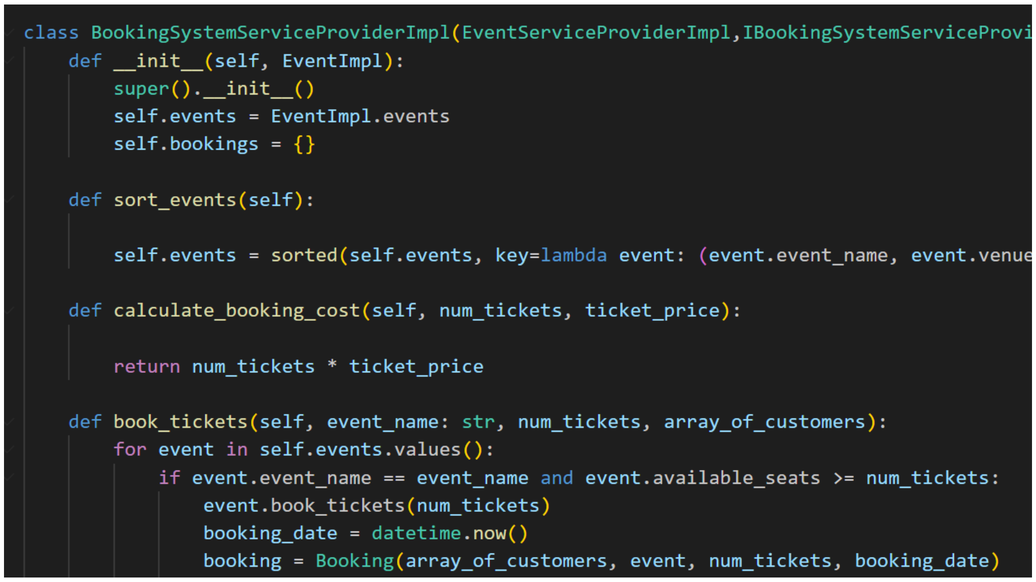
```
from dbutil import DBConnection

con = DBConnection.getConnection()
cur=con.cursor()

class EventNotFoundException(Exception):
    pass
class InvalidBookingIDException(Exception):
    pass


class TicketBookingSystem1():

    def book_tickets_menu(self):
        try:
            eventname = input("Enter the event name: ")
            # Check if the event exists
            query1="select * from event where event_name=%s"
            cur.execute(query1,(eventname,))
            event=cur.fetchone()
```



```
class TicketBookingSystem1():

    def booking_details_menu(self):
        try:
            booking_id = input("Enter the booking ID: ")
            query1 = "select * from booking where booking_id=%s"
            cur.execute(query1,(booking_id,))
            booking = cur.fetchone()

            if not booking:
                raise InvalidBookingIDException(f"Invalid booking ID: {booking_id}")

        except InvalidBookingIDException as e:
            print(f"Error: {e}")

    def event_exists(self, event_name):
        pass

    def is_valid_booking_id(self, booking_id):
        pass
```

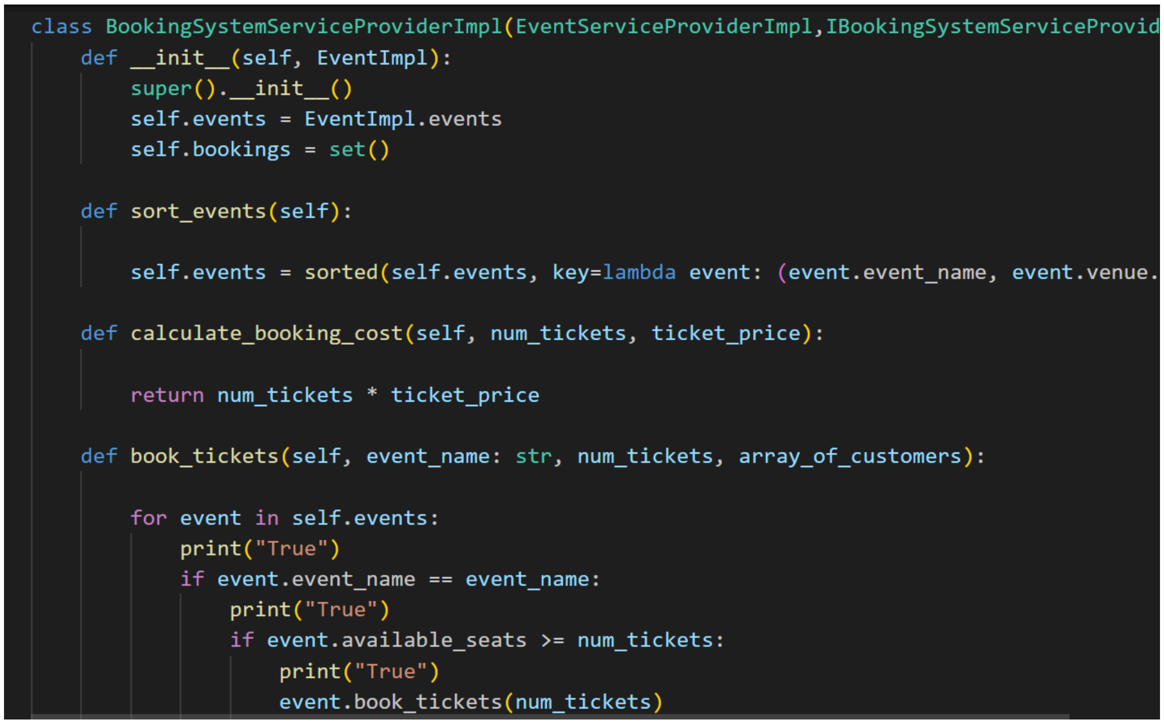
Task 10:Collection

Map:



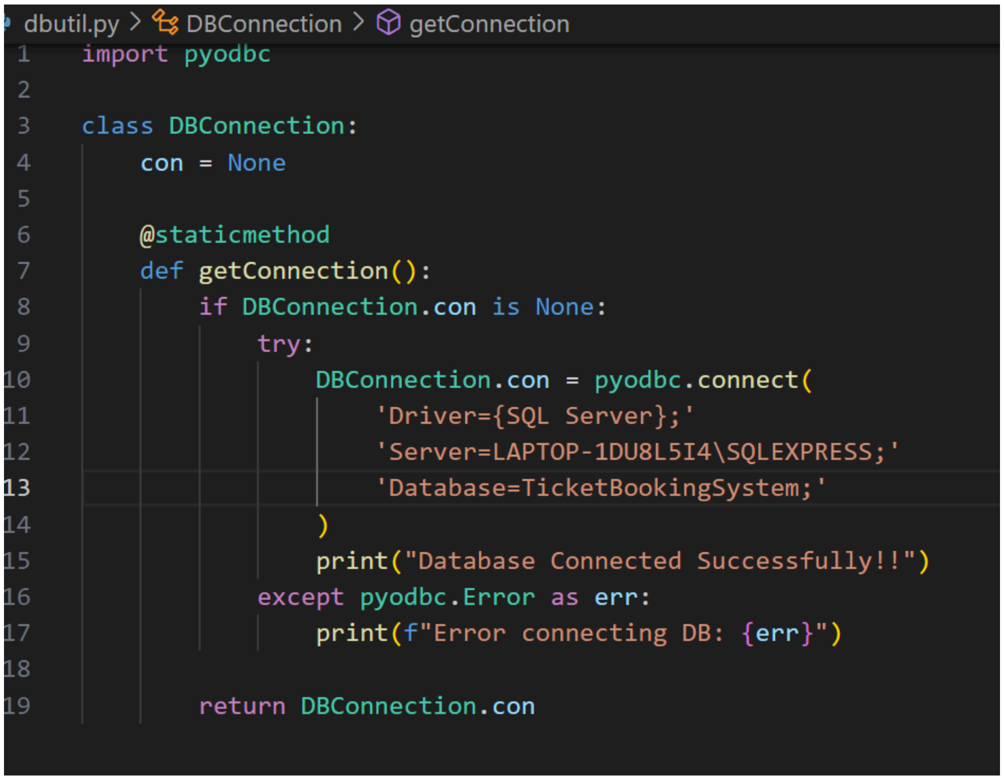
```
class BookingSystemServiceProviderImpl(EventServiceProviderImpl,IBookingSystemServiceProvi
    def __init__(self, EventImpl):
        super().__init__()
        self.events = EventImpl.events
        self.bookings = {}

    def sort_events(self):

        self.events = sorted(self.events, key=lambda event: (event.event_name, event.venue

    def calculate_booking_cost(self, num_tickets, ticket_price):

        return num_tickets * ticket_price

    def book_tickets(self, event_name: str, num_tickets, array_of_customers):
        for event in self.events.values():
            if event.event_name == event_name and event.available_seats >= num_tickets:
                event.book_tickets(num_tickets)
                booking_date = datetime.now()
                booking = Booking(array_of_customers, event, num_tickets, booking_date)
```

Set:



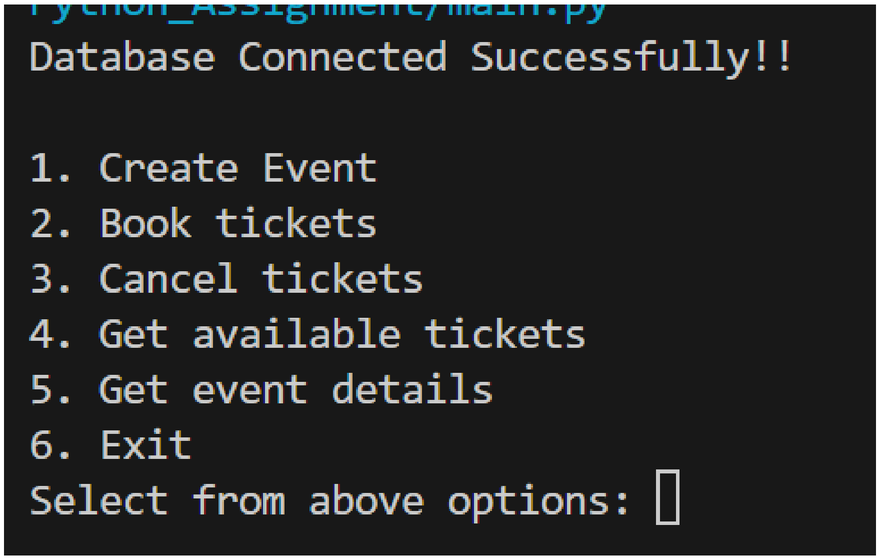
```
class BookingSystemServiceProviderImpl(EventServiceProviderImpl,IBookingSystemServiceProvid
    def __init__(self, EventImpl):
        super().__init__()
        self.events = EventImpl.events
        self.bookings = set()

    def sort_events(self):

        self.events = sorted(self.events, key=lambda event: (event.event_name, event.venue.

    def calculate_booking_cost(self, num_tickets, ticket_price):

        return num_tickets * ticket_price

    def book_tickets(self, event_name: str, num_tickets, array_of_customers):

        for event in self.events:
            print("True")
            if event.event_name == event_name:
                print("True")
                if event.available_seats >= num_tickets:
                    print("True")
                    event.book_tickets(num_tickets)
```

Task 11: DataBase Connectivity



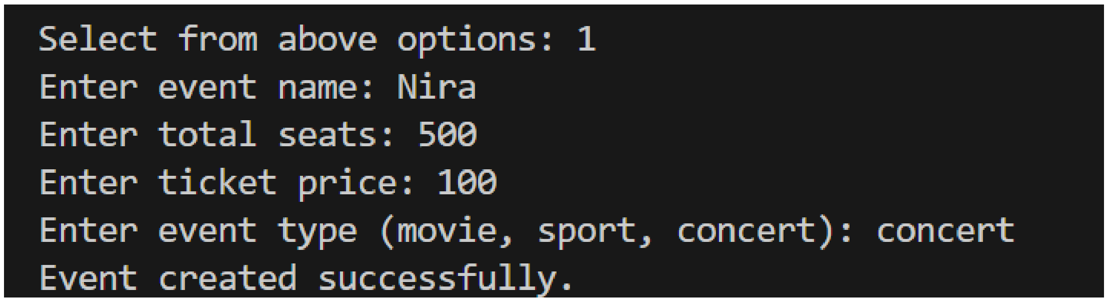
```python
import pyodbc

class DBConnection:
    con = None

    @staticmethod
    def getConnection():
        if DBConnection.con is None:
            try:
                DBConnection.con = pyodbc.connect(
                    'Driver={SQL Server};'
                    'Server=LAPTOP-1DU8L5I4\SQLEXPRESS;'
                    'Database=TicketBookingSystem;'
                )
                print("Database Connected Successfully!!")
            except pyodbc.Error as err:
                print(f"Error connecting DB: {err}")

        return DBConnection.con
```

Output:

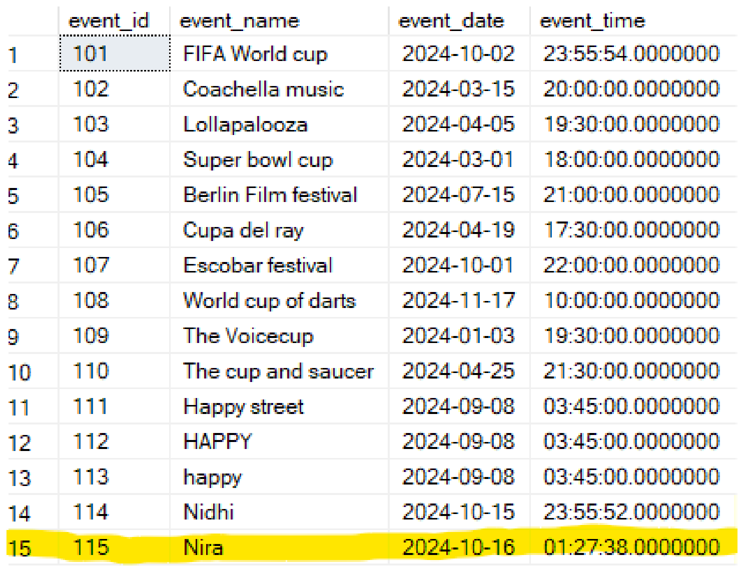


```
Database Connected Successfully!!

1. Create Event
2. Book tickets
3. Cancel tickets
4. Get available tickets
5. Get event details
6. Exit
Select from above options:
```

1.Create Event

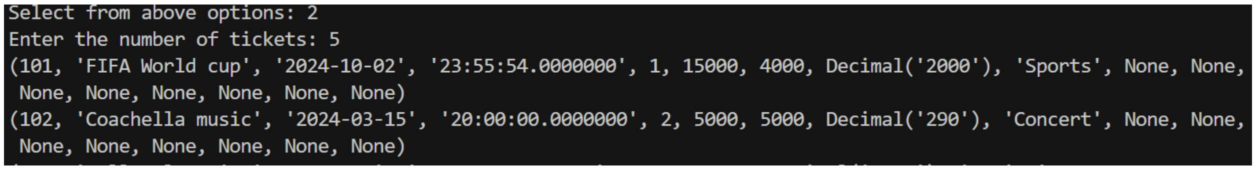


```
Select from above options: 1
Enter event name: Nira
Enter total seats: 500
Enter ticket price: 100
Enter event type (movie, sport, concert): concert
Event created successfully.
```

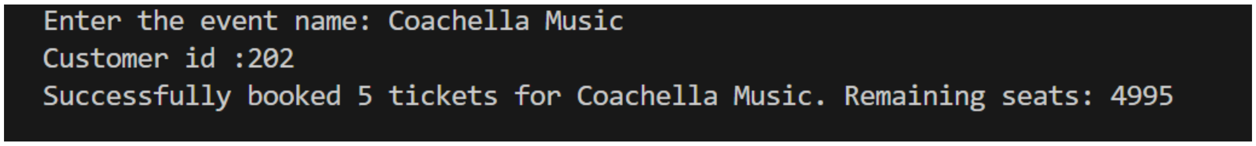


	event_id	event_name	event_date	event_time
1	101	FIFA World cup	2024-10-02	23:55:54.0000000
2	102	Coachella music	2024-03-15	20:00:00.0000000
3	103	Lollapalooza	2024-04-05	19:30:00.0000000
4	104	Super bowl cup	2024-03-01	18:00:00.0000000
5	105	Berlin Film festival	2024-07-15	21:00:00.0000000
6	106	Cupa del ray	2024-04-19	17:30:00.0000000
7	107	Escobar festival	2024-10-01	22:00:00.0000000
8	108	World cup of darts	2024-11-17	10:00:00.0000000
9	109	The Voicecup	2024-01-03	19:30:00.0000000
10	110	The cup and saucer	2024-04-25	21:30:00.0000000
11	111	Happy street	2024-09-08	03:45:00.0000000
12	112	HAPPY	2024-09-08	03:45:00.0000000
13	113	happy	2024-09-08	03:45:00.0000000
14	114	Nidhi	2024-10-15	23:55:52.0000000
15	115	Nira	2024-10-16	01:27:38.0000000

2.Book Tickets:



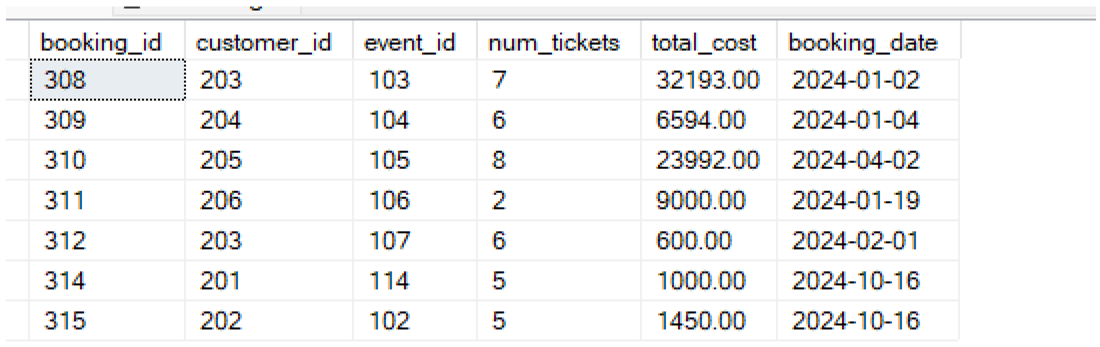
```
Select from above options: 2
Enter the number of tickets: 5
(101, 'FIFA World cup', '2024-10-02', '23:55:54.0000000', 1, 15000, 4000, Decimal('2000'), 'Sports', None, None,
 None, None, None, None, None, None)
(102, 'Coachella music', '2024-03-15', '20:00:00.0000000', 2, 5000, 5000, Decimal('290'), 'Concert', None, None,
 None, None, None, None, None, None)
```



```
Enter the event name: Coachella Music
Customer id :202
Successfully booked 5 tickets for Coachella Music. Remaining seats: 4995
```

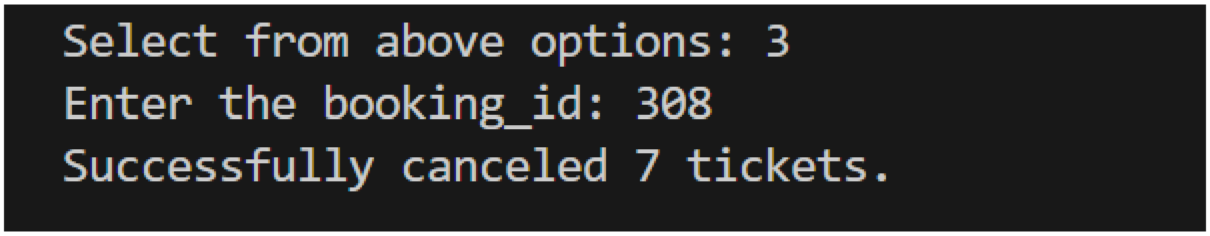
3.Cancel Tickets:

Before Cancelling Tickets:

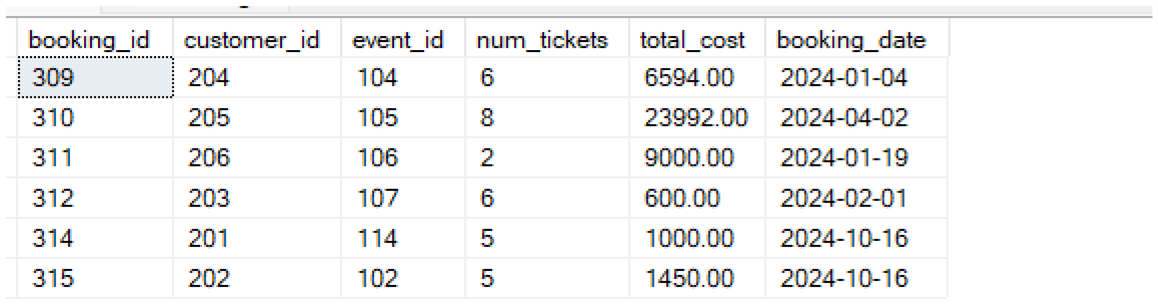


booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
308	203	103	7	32193.00	2024-01-02
309	204	104	6	6594.00	2024-01-04
310	205	105	8	23992.00	2024-04-02
311	206	106	2	9000.00	2024-01-19
312	203	107	6	600.00	2024-02-01
314	201	114	5	1000.00	2024-10-16
315	202	102	5	1450.00	2024-10-16

After Cancelling Tickets:

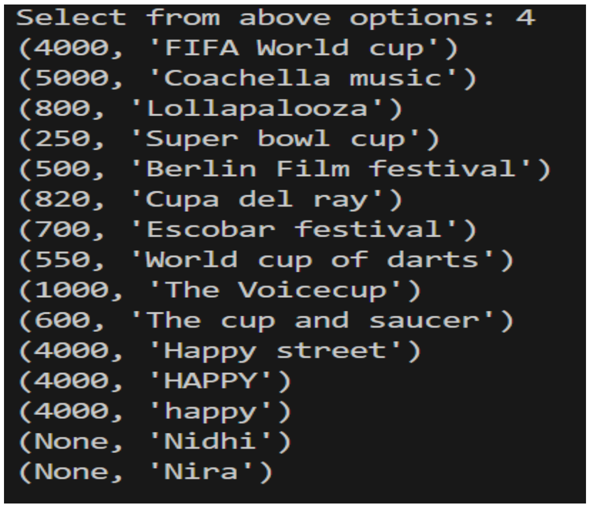


```
Select from above options: 3
Enter the booking_id: 308
Successfully canceled 7 tickets.
```



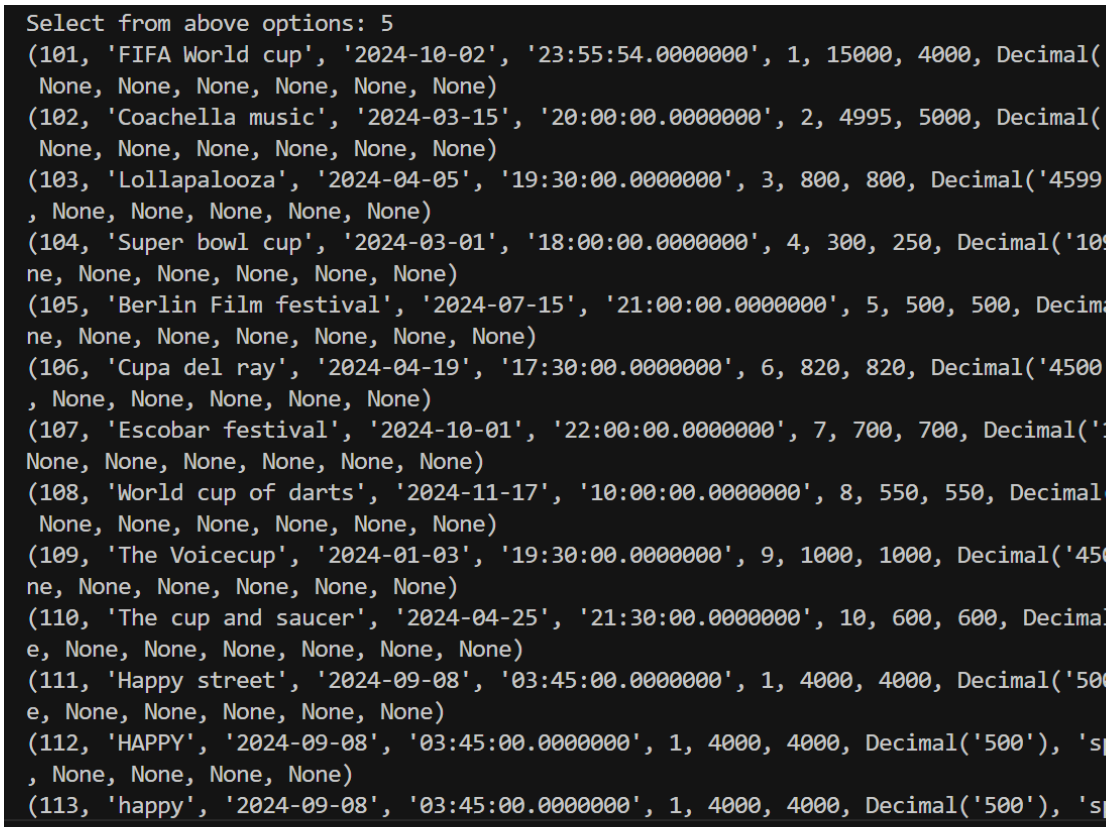
booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
309	204	104	6	6594.00	2024-01-04
310	205	105	8	23992.00	2024-04-02
311	206	106	2	9000.00	2024-01-19
312	203	107	6	600.00	2024-02-01
314	201	114	5	1000.00	2024-10-16
315	202	102	5	1450.00	2024-10-16

4.Get Available Tickets:



```
Select from above options: 4
(4000, 'FIFA World cup')
(5000, 'Coachella music')
(800, 'Lollapalooza')
(250, 'Super bowl cup')
(500, 'Berlin Film festival')
(820, 'Cupa del ray')
(700, 'Escobar festival')
(550, 'World cup of darts')
(1000, 'The Voicecup')
(600, 'The cup and saucer')
(4000, 'Happy street')
(4000, 'HAPPY')
(4000, 'happy')
(None, 'Nidhi')
(None, 'Nira')
```

5.Get Event Details:



```
Select from above options: 5
(101, 'FIFA World cup', '2024-10-02', '23:55:54.0000000', 1, 15000, 4000, Decimal(
 None, None, None, None, None, None)
(102, 'Coachella music', '2024-03-15', '20:00:00.0000000', 2, 4995, 5000, Decimal(
 None, None, None, None, None, None)
(103, 'Lollapalooza', '2024-04-05', '19:30:00.0000000', 3, 800, 800, Decimal('4599
, None, None, None, None, None)
(104, 'Super bowl cup', '2024-03-01', '18:00:00.0000000', 4, 300, 250, Decimal('10
ne, None, None, None, None, None)
(105, 'Berlin Film festival', '2024-07-15', '21:00:00.0000000', 5, 500, 500, Decim
ne, None, None, None, None, None, None)
(106, 'Cupa del ray', '2024-04-19', '17:30:00.0000000', 6, 820, 820, Decimal('4500
, None, None, None, None, None)
(107, 'Escobar festival', '2024-10-01', '22:00:00.0000000', 7, 700, 700, Decimal('
None, None, None, None, None, None)
(108, 'World cup of darts', '2024-11-17', '10:00:00.0000000', 8, 550, 550, Decimal
 None, None, None, None, None, None)
(109, 'The Voicecup', '2024-01-03', '19:30:00.0000000', 9, 1000, 1000, Decimal('45
ne, None, None, None, None, None)
(110, 'The cup and saucer', '2024-04-25', '21:30:00.0000000', 10, 600, 600, Decima
e, None, None, None, None, None, None)
(111, 'Happy street', '2024-09-08', '03:45:00.0000000', 1, 4000, 4000, Decimal('50
e, None, None, None, None, None)
(112, 'HAPPY', '2024-09-08', '03:45:00.0000000', 1, 4000, 4000, Decimal('500'), 's
, None, None, None, None)
(113, 'happy', '2024-09-08', '03:45:00.0000000', 1, 4000, 4000, Decimal('500'), 's
```