Task 1 to 6:

Task 1:

Code

```
# Ticket Booking System

# Input: Available tickets and number of tickets to book
availableTicket = int(input("Enter the number of available tickets: "))
noOfBookingTicket = int(input("Enter the number of tickets to book: "))

# Check availability
if availableTicket >= noOfBookingTicket:
    remainingTickets = availableTicket - noOfBookingTicket
    print("Booking successful!")
    print("Remaining tickets:", remainingTickets)
else:
    print("Tickets unavailable!")
```

output:

```
Enter the number of available tickets: 23
Enter the number of tickets to book: 4
Booking successful!
Remaining tickets: 19
```

Task 2 and 3:

Code:

```
def book_tickets():
    print("Ticket Categories:")
    print("1. Silver - Rs 50")
    print("2. Gold - Rs 100")
    print("3. Diamond - Rs 150")

    ticket_type = input("Enter ticket category (Silver/Gold/Diamond):
").strip().lower()
    no_of_tickets = int(input("Enter the number of tickets to book: "))

    if no_of_tickets > 0:
        if ticket_type == "silver":
```

```
            price = 50
        elif ticket_type == "gold":
            price = 100
        elif ticket_type == "diamond":
            price = 150
        else:
            print("Invalid ticket type. Please try again.")
            return

        total_cost = no_of_tickets * price
        print(f"Booking successful Total cost: Rs{total_cost}")
    else:
        print("Invalid number of tickets.")

book_tickets()
```

output:

```
Enter ticket category (Silver/Gold/Diamond): gold
Enter the number of tickets to book: 56
Booking successful Total cost: Rs5600
```

Task 4 and 5:

Code:

```
from abc import ABC, abstractmethod
from datetime import date, time, datetime
from enum import Enum

# Enum for Event Types
class EventType(Enum):
    MOVIE = "Movie"
    SPORTS = "Sports"
    CONCERT = "Concert"

# 1. Event Class (Abstract Class)
class Event(ABC):
    def __init__(self, event_name, event_date, event_time, venue_name,
total_seats, ticket_price, event_type):
        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 = total_seats
        self._ticket_price = ticket_price
        self._event_type = event_type

    @abstractmethod
    def display_event_details(self):
        pass

    def calculate_total_revenue(self):
        return (self._total_seats - self._available_seats) * self._ticket_price

    def get_booked_no_of_tickets(self):
        return self._total_seats - self._available_seats

    def book_tickets(self, num_tickets):
        if self._available_seats >= num_tickets:
            self._available_seats -= num_tickets
            return True
        else:
            return False

    def cancel_booking(self, num_tickets):
        if self._total_seats - self._available_seats >= num_tickets:
            self._available_seats += num_tickets
            return True
        else:
            return False

    def display_event_info(self):
        return f"Event: {self._event_name}, Date: {self._event_date}, Time:
{self._event_time}, " \
               f"Venue: {self._venue_name}, Available Seats:
{self._available_seats}"

    @property
    def event_name(self): return self._event_name
    @property
    def available_seats(self): return self._available_seats
    @property
    def ticket_price(self): return self._ticket_price

# 2. Venue Class
class Venue:
    def __init__(self, venue_name, address):
```

```
        self._venue_name = venue_name
        self._address = address

    def display_venue_details(self):
        return f"Venue: {self._venue_name}, Address: {self._address}"

# 3. Customer Class
class Customer:
    def __init__(self, customer_name, email, phone_number):
        self._customer_name = customer_name
        self._email = email
        self._phone_number = phone_number

    def display_customer_details(self):
        return f"Customer Name: {self._customer_name}, Email: {self._email},
Phone: {self._phone_number}"

# 4. Booking Class
class Booking:
    def __init__(self, event, customer):
        self._event = event
        self._customer = customer
        self._total_cost = 0

    def calculate_booking_cost(self, num_tickets):
        if self._event.book_tickets(num_tickets):
            self._total_cost = num_tickets * self._event.ticket_price
            return f"Booking Successful. Total Cost: {self._total_cost}"
        else:
            return "Not enough available seats for the booking."

    def cancel_booking(self, num_tickets):
        if self._event.cancel_booking(num_tickets):
            self._total_cost = 0
            return "Booking cancelled successfully."
        else:
            return "Unable to cancel booking."

# 5. Event Subclasses
class Movie(Event):
    def __init__(self, event_name, event_date, event_time, venue_name,
total_seats, ticket_price, genre, actor_name, actress_name):
        super().__init__(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, EventType.MOVIE)
        self._genre = genre
```

```
        self._actor_name = actor_name
        self._actress_name = actress_name

    def display_event_details(self):
        return f"Movie: {self._event_name}, Genre: {self._genre}, Actor:
{self._actor_name}, Actress: {self._actress_name}"

class Concert(Event):
    def __init__(self, event_name, event_date, event_time, venue_name,
total_seats, ticket_price, artist, type):
        super().__init__(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, EventType.CONCERT)
        self._artist = artist
        self._type = type

    def display_event_details(self):
        return f"Concert: {self._event_name}, Artist: {self._artist}, Type:
{self._type}"

class Sports(Event):
    def __init__(self, event_name, event_date, event_time, venue_name,
total_seats, ticket_price, sport_name, teams_name):
        super().__init__(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, EventType.SPORTS)
        self._sport_name = sport_name
        self._teams_name = teams_name

    def display_event_details(self):
        return f"Sport Event: {self._event_name}, Sport: {self._sport_name},
Teams: {self._teams_name}"

# 6. TicketBookingSystem Class
class TicketBookingSystem:
    def __init__(self):
        self.events = []

    def create_event(self, event_name, event_date, event_time, total_seats,
ticket_price, event_type, venue_name):
        if event_type == "Movie":
            event = Movie(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, "Action", "Actor", "Actress")
        elif event_type == "Concert":
            event = Concert(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, "Artist", "Theatrical")
        elif event_type == "Sports":
```

```
            event = Sports(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, "Cricket", "India vs Pakistan")
        else:
            print("Invalid event type.")
            return None

        self.events.append(event)
        return event

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

    def book_tickets(self, event, num_tickets):
        if event.book_tickets(num_tickets):
            return f"Successfully booked {num_tickets} tickets for
{event.event_name}"
        else:
            return "Not enough available seats."

    def cancel_tickets(self, event, num_tickets):
        if event.cancel_booking(num_tickets):
            return f"Successfully cancelled {num_tickets} tickets."
        else:
            return "Unable to cancel booking."

# 7. Main Method
def main():
    ticket_system = TicketBookingSystem()

    while True:
        print("\nTicket Booking System")
        print("1. Create Event")
        print("2. Book Tickets")
        print("3. Cancel Tickets")
        print("4. Get Available Seats")
        print("5. Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            event_type = input("Enter event type (Movie/Concert/Sports):
").strip()
            event_name = input("Enter event name: ").strip()

            # Fix: Convert string to date and time objects
            event_date_str = input("Enter event date (YYYY-MM-DD): ").strip()
```

```
            event_time_str = input("Enter event time (HH:MM): ").strip()
            try:
                event_date = datetime.strptime(event_date_str, "%Y-%m-%d").date()
                event_time = datetime.strptime(event_time_str, "%H:%M").time()
            except ValueError:
                print("Invalid date or time format.")
                continue

            venue_name = input("Enter venue name: ").strip()
            total_seats = int(input("Enter total number of seats: ").strip())
            ticket_price = float(input("Enter ticket price: ").strip())

            event = ticket_system.create_event(event_name, event_date,
event_time, total_seats, ticket_price, event_type, venue_name)
            if event:
                print("Event Created Successfully!")
                ticket_system.display_event_details(event)

        elif choice == "2":
            event_name = input("Enter event name to book tickets: ").strip()
            num_tickets = int(input("Enter number of tickets to book: ").strip())

            event = next((e for e in ticket_system.events if e.event_name ==
event_name), None)
            if event:
                print(ticket_system.book_tickets(event, num_tickets))
            else:
                print("Event not found!")

        elif choice == "3":
            event_name = input("Enter event name to cancel tickets: ").strip()
            num_tickets = int(input("Enter number of tickets to cancel:
").strip())

            event = next((e for e in ticket_system.events if e.event_name ==
event_name), None)
            if event:
                print(ticket_system.cancel_tickets(event, num_tickets))
            else:
                print("Event not found!")

        elif choice == "4":
            event_name = input("Enter event name to get available seats:
").strip()
```

```
            event = next((e for e in ticket_system.events if e.event_name ==
event_name), None)
            if event:
                print(f"Available seats for {event_name}:
{event.available_seats}")
            else:
                print("Event not found!")

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

        else:
            print("Invalid choice! Please enter a valid option.")

if __name__ == "__main__":
    main()
```

output:

```
Ticket Booking System
1. Create Event
2. Book Tickets
3. Cancel Tickets
4. Get Available Seats
5. Exit
Enter your choice: 2
Enter event name to book tickets: fire
Enter number of tickets to book: 89
Event not found!
```

Task 6:

```
from abc import ABC, abstractmethod
from datetime import datetime
from enum import Enum

class EventType(Enum):
    MOVIE = "Movie"
    CONCERT = "Concert"
    SPORTS = "Sports"

class Event(ABC):
    def __init__(self, event_name, event_date, event_time, venue_name,
total_seats, ticket_price, event_type):
```

```
        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 = total_seats
        self._ticket_price = ticket_price
        self._event_type = event_type

    @abstractmethod
    def display_event_details(self):
        pass

    def book_tickets(self, num_tickets):
        if self._available_seats >= num_tickets:
            self._available_seats -= num_tickets
            return True
        return False

    def cancel_tickets(self, num_tickets):
        if (self._total_seats - self._available_seats) >= num_tickets:
            self._available_seats += num_tickets
            return True
        return False

    def get_available_seats(self):
        return self._available_seats

    @property
    def event_name(self):
        return self._event_name
class Movie(Event):
    def __init__(self, event_name, event_date, event_time, venue_name,
total_seats, ticket_price, genre, actor, actress):
        super().__init__(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, EventType.MOVIE)
        self._genre = genre
        self._actor = actor
        self._actress = actress

    def display_event_details(self):
        return f"[Movie] {self._event_name} - Genre: {self._genre}, Lead:
{self._actor}, Actress: {self._actress}"

class Concert(Event):
```

```
    def __init__(self, event_name, event_date, event_time, venue_name,
total_seats, ticket_price, artist, concert_type):
        super().__init__(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, EventType.CONCERT)
        self._artist = artist
        self._concert_type = concert_type

    def display_event_details(self):
        return f"[Concert] {self._event_name} - Artist: {self._artist}, Type:
{self._concert_type}"

class Sports(Event):
    def __init__(self, event_name, event_date, event_time, venue_name,
total_seats, ticket_price, sport_name, teams):
        super().__init__(event_name, event_date, event_time, venue_name,
total_seats, ticket_price, EventType.SPORTS)
        self._sport_name = sport_name
        self._teams = teams

    def display_event_details(self):
        return f"[Sports] {self._event_name} - Sport: {self._sport_name}, Teams:
{self._teams}"
class BookingSystem(ABC):
    @abstractmethod
    def create_event(self, *args, **kwargs):
        pass

    @abstractmethod
    def book_tickets(self, event_name, num_tickets):
        pass

    @abstractmethod
    def cancel_tickets(self, event_name, num_tickets):
        pass

    @abstractmethod
    def get_available_seats(self, event_name):
        pass
class TicketBookingSystem(BookingSystem):
    def __init__(self):
        self.events = []

    def create_event(self, event_type, *args):
        if event_type == "Movie":
            event = Movie(*args)
```

```
        elif event_type == "Concert":
            event = Concert(*args)
        elif event_type == "Sports":
            event = Sports(*args)
        else:
            print("Invalid event type.")
            return None

        self.events.append(event)
        return event

    def find_event(self, event_name):
        for event in self.events:
            if event.event_name == event_name:
                return event
        return None

    def book_tickets(self, event_name, num_tickets):
        event = self.find_event(event_name)
        if event:
            if event.book_tickets(num_tickets):
                return "Tickets booked successfully."
            else:
                return "Not enough seats available."
        return "Event not found."

    def cancel_tickets(self, event_name, num_tickets):
        event = self.find_event(event_name)
        if event:
            if event.cancel_tickets(num_tickets):
                return "Tickets cancelled successfully."
            else:
                return "Cannot cancel tickets. Check your booking count."
        return "Event not found."

    def get_available_seats(self, event_name):
        event = self.find_event(event_name)
        if event:
            return f"Available seats: {event.get_available_seats()}"
        return "Event not found."
def main():
    system = TicketBookingSystem()

    while True:
        print("\n--- Ticket Booking System ---")
```

```
        print("1. create_event")
        print("2. book_tickets")
        print("3. cancel_tickets")
        print("4. get_available_seats")
        print("5. exit")

        choice = input("Enter command: ").strip()

        if choice == "create_event":
            event_type = input("Event type (Movie/Concert/Sports): ")
            event_name = input("Event name: ")
            date_str = input("Event date (YYYY-MM-DD): ")
            time_str = input("Event time (HH:MM): ")
            venue = input("Venue name: ")
            seats = int(input("Total seats: "))
            price = float(input("Ticket price: "))

            try:
                event_date = datetime.strptime(date_str, "%Y-%m-%d").date()
                event_time = datetime.strptime(time_str, "%H:%M").time()
            except ValueError:
                print("Invalid date/time format.")
                continue

            if event_type == "Movie":
                genre = input("Genre: ")
                actor = input("Lead Actor: ")
                actress = input("Lead Actress: ")
                event = system.create_event("Movie", event_name, event_date,
event_time, venue, seats, price, genre, actor, actress)
            elif event_type == "Concert":
                artist = input("Artist: ")
                concert_type = input("Concert type: ")
                event = system.create_event("Concert", event_name, event_date,
event_time, venue, seats, price, artist, concert_type)
            elif event_type == "Sports":
                sport_name = input("Sport name: ")
                teams = input("Teams playing: ")
                event = system.create_event("Sports", event_name, event_date,
event_time, venue, seats, price, sport_name, teams)
            else:
                print("Invalid event type.")
                continue

            if event:
```

```
                print("Event created successfully.")
                print(event.display_event_details())

        elif choice == "book_tickets":
            name = input("Event name: ")
            num = int(input("Number of tickets: "))
            print(system.book_tickets(name, num))

        elif choice == "cancel_tickets":
            name = input("Event name: ")
            num = int(input("Number of tickets to cancel: "))
            print(system.cancel_tickets(name, num))

        elif choice == "get_available_seats":
            name = input("Event name: ")
            print(system.get_available_seats(name))

        elif choice == "exit":
            print("Exiting system.")
            break

        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    main()
```