Assignment-5

Create Venue, Event, Customer and Booking class

Customers:

```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

    #getters
    @property
    def getfirstname(self):
        return self.customer_name

    @property
    def getlastname(self):
        return self.lastname

    @property
    def getemail(self):
        return self.email

    @property
    def getphone_number(self):
        return self.phone_number
```

```python
    @property
    def getaddress(self):
        return self.address

    #setters
    @getfirstname.setter
    def set_customer_name(self,firstname):
        self.firstname=firstname

    @getlastname.setter
    def setlastname(self, lastname):
        self.lastname = lastname
    @getemail.setter
    def set_email(self,email):
        self.email=email
    @getphone_number.setter
    def set_phone_number(self,phone_number):
        self.phone_number=phone_number

    @getaddress.setter
    def setaddress(self, address):
        self.address = address
```

Event:

```
import decimal
import enum


class event:
    def __init__(self,event_name,event_date,event_time,venue_name,total_seats,avaliable_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.avaliable_seats=avaliable_seats
        self.ticket_price=ticket_price
        self.event_type=event_type

    # getters
    @property
    def get_event_name(self):
        return self.event_name

    @property
    def get_event_date(self):
        return self.event_date
```

```
    @property
    def get_event_time(self):
        return self.event_time

    @property
    def get_venue_name(self):
        return self.venue_name

    @property
    def get_total_seats(self):
        return self.total_seats

    @property
    def get_avaliable_seats(self):
        return self.avaliable_seats

    @property
    def get_ticket_price(self):
        return self.ticket_price

    @property
    def get_event_type(self):
        return self.event_type
```

```
#setters
@get_event_date.setter
def set_event_name(self,event_name):
     self.event_name=event_name
@ get_event_date.setter
def set_event_date(self,event_date):
    self.event_date=event_date
@get_event_time.setter
def set_event_time(self,event_time):
    self.event_time=event_time
@get_venue_name.setter
def set_venue_name(self,venue_name):
    self.venue_name=venue_name
@get_total_seats.setter
def set_total_seats(self,total_seats):
    self.total_seats=total_seats
@get_avaliable_seats.setter
def set_avaliable_seats(self,avaliable_seats):
    self.avaliable_seats=avaliable_seats
@get_ticket_price.setter
def set_ticket_price(self,ticket_price):
    self.ticket_price=ticket_price
@get_event_type.setter
def set_event_type(self,event_type):
    self.event_type=event_type
```

```
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("avaliable_seats: ", self.avaliable_seats)
    print("ticket_price: ", self.ticket_price)
    print("event_type: ", self.event_type)
```

Venue:

```
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)

    # getters
    @property
    def get_venue_name(self):
        return self.venue_name
    @property
    def get_address(self):
        return self.address

    # setters
    @get_venue_name.setter
    def set_venue_name(self, venue_name):
        self.venue_name = venue_name
    @get_address.setter
    def set_address(self, address):
        self.address = address
```

Bookings:

```
class bookings:
    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

    @property
    def getevent_id(self):
        return self.event_id
    @property
    def getnum_tickets(self):
        return self.num_tickets
    @property
    def total_cost(self):
        return self.total_cost
    @property
    def booking_date(self):
        return self.booking_date

    @getevent_id.setter
    def setevent_id(self,event_id):
        self.event_id=event_id
    @getnum_tickets.setter
    def setnum_tickets(self,num_tickets):
        self.num_tickets=num_tickets
```

```
    @total_cost.setter
    def settotal_cost(self,total_cost):
        self.total_cost=total_cost

    @booking_date.setter
    def setbooking_date(self,booking_date):
        self.booking_date=booking_date
```

Exception:

throw the exception whenever needed and Handle in main method,

1. EventNotFoundException throw this exception when user try to book the tickets for Event not listed in the menu.

2. InvalidBookingIDException throw this exception when user entered the invalid bookingId when he tries to view the booking or cancel the booking.

```
class EventNotFoundException(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()

            if not event:
                raise EventNotFoundException(f"Event '{eventname}' not found in the menu.")



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

```
class InvalidBookingIDException(Exception):
    pass

    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
```

```
if __name__ == "__main__":


    ticket = TicketBookingSystem1()
    ticket.book_tickets_menu()
    ticket.booking_details_menu()
```

Abstract method:

Create IBookingSystemRepository interface/abstract class which include following methods to interact with database.

• create_event(event_name: str, date:str, time:str, total_seats: int, ticket_price: float, event_type: str, venu: Venu): Create a new event with the specified details and event type (movie, sport or concert) and return event object and should store in database.

• getEventDetails(): return array of event details from the database.

• getAvailableNoOfTickets(): return the total available tickets from the database.

• calculate_booking_cost(num_tickets): Calculate and set the total cost of the booking.

• book_tickets(eventname:str, num_tickets, listOfCustomer): Book a specified number of tickets for an event. for each tickets customer object should be created and stored in array also should update the attributes of Booking class and stored in database.

• cancel_booking(booking_id): Cancel the booking and update the available seats and stored in database.

• get_booking_details(booking_id): get the booking details from database.

```python
from abc import ABC, abstractmethod

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

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

DButil:

Create DBUtil class and add the following method.

• static getDBConn():Connection Establish a connection to the database and return Connection reference

```
import mysql.connector
class dbutil:
    @staticmethod
    def dbconn():
        con=mysql.connector.connect(
                    host="localhost",
                    user="root",
                    password="root",
                    port="3306",
                    database="TicketBookingSystem"
                    )
        print(con)
        cur=con.cursor()

if __name__ == "__main__":
    # Create an instance of the DBUtil class
    db_util = dbutil()
    dbutil.dbconn()
```

Create TicketBookingSystem class and perform following operations:

• Create a simple user interface in a main method that allows users to interact with the ticket booking system by entering commands such as "create_event", "book_tickets", "cancel_tickets", "get_available_seats,", "get_event_details," and "exit."

Ticket Booking System App:

```
class TicketBookingSystem:
    while True:
        print("1.create Event")
        print("2.book_tickets")
        print("3.cancel tickets")
        print("4.get avaliable tickets")
        print("5.get event details")
        print("6.exit")
        choice=input("select from above options: ")
        if choice=="1":
            b.create_event()
        elif choice=="2":
            num_tickets=input("enter the num_tickets")
            b.book_tickets(num_tickets)
        elif choice=="3":
            b.cancel_tickets()
        elif choice=="4":
            b.get_avaliable_tickets()
        elif choice=="5":
            b.get_event_details()
        elif choice=="6":
            print("exiting the system\n Thank you")
            break;
        else:
            print("invalid option chosen, chose from above options")
```

Create IBookingSystemRepository interface/abstract class which include following methods to interact with database.

• create_event(event_name: str, date:str, time:str, total_seats: int, ticket_price: float, event_type: str, venu: Venu): Create a new event with the specified details and event type (movie, sport or concert) and return event object and should store in database.

• getEventDetails(): return array of event details from the database.

• getAvailableNoOfTickets(): return the total available tickets from the database.

• calculate_booking_cost(num_tickets): Calculate and set the total cost of the booking.

• book_tickets(eventname:str, num_tickets, listOfCustomer): Book a specified number of tickets for an event. for each tickets customer object should be created and stored in array also should update the attributes of Booking class and stored in database.

• cancel_booking(booking_id): Cancel the booking and update the available seats and stored in database.

• get_booking_details(booking_id): get the booking details from database.

DataBase:

```
import functools
from datetime import *
import mysql.connector

from abstract_methods import BookingSystemRepositoryImpl


class EventNotFoundException(Exception):
    pass

class InvalidBookingIDException(Exception):
    pass

con=mysql.connector.connect(
            host="localhost",
            user="root",
            password="root",
            port="3306",
            database="TicketBookingSystem"
            )
cur=con.cursor()
```

Create event:

```
class BookingSystemRepository(BookingSystemRepositoryImpl):

    def create_event(self):
        #event_id=self.generate_unique_event_id()
        event_id=input("enter the id")
        event_name = input("Enter event name: ")
        date=self.get_current_date()
        time=self.get_current_time()
       # venue_id = input("Enter venue id: ")
        total_seats = int(input("Enter total seats: "))
        ticket_price = float(input("Enter ticket price: "))
        event_type = input("Enter event type (movie, sport, concert): ")
        eve1={
            'event_id':event_id,
            'event_name': event_name,
            'event_date':date,
            'event_time':time,
            #'venue-id':venue_id,
            'total_Seats':total_seats,
            'ticket_price':ticket_price,
            'event_type':event_type
        }
```

```
    query = "insert into event(event_id,event_name,event_date,event_time,total_seats,ticket_price,event_type)  " \
            "values(%s,%s,%s,%s,%s,%s,%s)"
    values = (eve1['event_id'],eve1['event_name'],eve1['event_date'],eve1['event_time'],
              eve1['total_Seats'], eve1['ticket_price'],eve1['event_type'])
    cur.execute(query, values)
    cur.execute("select * from event")
    user = cur.fetchall()
    con.commit()
    for i in user:
        print(i)
def get_current_date(self):
    return date.today()
def get_current_time(self):
    return datetime.now().time()
```

Output:

```
1.create Event
2.book_tickets
3.cancel tickets
4.get avaliable tickets
5.get event details
6.exit
select from above options: 1
enter the id12
Enter event name: ipl
Enter total seats: 500
Enter ticket price: 1500
Enter event type (movie, sport, concert): sport
(1, 'Bollywood Night Extravaganza', datetime.date(2024, 6, 17), datetime.timedelta(seconds=66600), 1, 50, 47, Decimal('250.00'), 'Concert', 101)
(2, 'Cricket Tournament Finals world cup', datetime.date(2024, 5, 20), datetime.timedelta(seconds=54000), 2, 15500, 15480, Decimal('2400.00'), 'Sports', 102)
(3, 'Classic Movie Marathon', datetime.date(2024, 7, 21), datetime.timedelta(seconds=43200), 3, 100, 95, Decimal('50.00'), 'Movie', 103)
(4, 'Live Stand-Up Comedy Show', datetime.date(2024, 5, 22), datetime.timedelta(seconds=68400), 4, 50, 45, Decimal('120.00'), 'Concert', 104)
(5, 'Football Championship', datetime.date(2024, 5, 1), datetime.timedelta(seconds=59400), 5, 100, 100, Decimal('80.00'), 'Sports', None)
(6, 'Melodious Sufi Night', datetime.date(2024, 8, 10), datetime.timedelta(seconds=72000), 6, 50, 49, Decimal('200.00'), 'Concert', 105)
(7, 'Outdoor Movie Night', datetime.date(2024, 6, 25), datetime.timedelta(seconds=70200), 7, 50, 45, Decimal('75.00'), 'Movie', 106)
(8, 'Basketball Showdown', datetime.date(2024, 7, 15), datetime.timedelta(seconds=50400), 8, 150, 145, Decimal('60.00'), 'Sports', 107)
(9, 'Folk Music Festival', datetime.date(2024, 9, 3), datetime.timedelta(seconds=61200), 9, 50, 47, Decimal('150.00'), 'Concert', 108)
(10, 'Tennis Championship asian cup', datetime.date(2024, 9, 11), datetime.timedelta(seconds=45000), 10, 200, 190, Decimal('1000.00'), 'Sports', 109)
(11, 'cricket', datetime.date(2024, 2, 1), datetime.timedelta(seconds=39596), None, 500, None, Decimal('1500.00'), 'sport', None)
(12, 'ipl', datetime.date(2024, 2, 3), datetime.timedelta(seconds=52592), None, 500, None, Decimal('1500.00'), 'sport', None)
```

Get event details:

```
def get_event_details(self):
    return self.all_events()
def all_events(self):
    query="select * from event"
    cur.execute(query)
    user = cur.fetchall()
    con.commit()
    for i in user:
        print(i)
```

Output:

```
1.create Event
2.book_tickets
3.cancel tickets
4.get avaliable tickets
5.get event details
6.exit
select from above options: 5
(1, 'Bollywood Night Extravaganza', datetime.date(2024, 6, 17), datetime.timedelta(seconds=66600), 1, 50, 47, Decimal('250.00'), 'Concert', 101)
(2, 'Cricket Tournament Finals world cup', datetime.date(2024, 5, 20), datetime.timedelta(seconds=54000), 2, 15500, 15480, Decimal('2400.00'), 'Sports', 102)
(3, 'Classic Movie Marathon', datetime.date(2024, 7, 21), datetime.timedelta(seconds=43200), 3, 100, 95, Decimal('50.00'), 'Movie', 103)
(4, 'Live Stand-Up Comedy Show', datetime.date(2024, 5, 22), datetime.timedelta(seconds=68400), 4, 50, 45, Decimal('120.00'), 'Concert', 104)
(5, 'Football Championship', datetime.date(2024, 5, 1), datetime.timedelta(seconds=59400), 5, 100, 100, Decimal('80.00'), 'Sports', None)
(6, 'Melodious Sufi Night', datetime.date(2024, 8, 10), datetime.timedelta(seconds=72000), 6, 50, 49, Decimal('200.00'), 'Concert', 105)
(7, 'Outdoor Movie Night', datetime.date(2024, 6, 25), datetime.timedelta(seconds=70200), 7, 50, 45, Decimal('75.00'), 'Movie', 106)
(8, 'Basketball Showdown', datetime.date(2024, 7, 15), datetime.timedelta(seconds=50400), 8, 150, 145, Decimal('60.00'), 'Sports', 107)
(9, 'Folk Music Festival', datetime.date(2024, 9, 3), datetime.timedelta(seconds=61200), 9, 50, 47, Decimal('150.00'), 'Concert', 108)
(10, 'Tennis Championship asian cup', datetime.date(2024, 9, 11), datetime.timedelta(seconds=45000), 10, 200, 190, Decimal('1000.00'), 'Sports', 109)
(11, 'ipl', datetime.date(2024, 2, 4), datetime.timedelta(seconds=36553), None, 500, None, Decimal('1500.00'), 'sports', None)
1.create Event
```

Get available tickets:

```
def get_avaliable_tickets(self):
    return self.avaliable_tickets()
def avaliable_tickets(self):

    query="select avaliable_seats,event_name from event"
    cur.execute(query)
    user = cur.fetchall()
    con.commit()
    for i in user:
        print(i)
```

Output:

```
1.create Event
2.book_tickets
3.cancel tickets
4.get avaliable tickets
5.get event details
6.exit
select from above options: 4
(47, 'Bollywood Night Extravaganza')
(15480, 'Cricket Tournament Finals world cup')
(95, 'Classic Movie Marathon')
(45, 'Live Stand-Up Comedy Show')
(100, 'Football Championship')
(49, 'Melodious Sufi Night')
(45, 'Outdoor Movie Night')
(145, 'Basketball Showdown')
(47, 'Folk Music Festival')
(190, 'Tennis Championship asian cup')
(None, 'ipl')
```

Create customers:

```
def create_customer(self):
    customer_id=self.unique_customer_id()
    customer_name=input("enter your name")
    email=input("enter your email")
    phone=input("enter your phone number")
    booking_id=self.unique_booking_id()
    cust={
        'customer_id':customer_id,
        'customer_name':customer_name,
        'email':email,
        'phone':phone,
        'booking_id':booking_id
    }
    query="insert into customer values(%s,%s,%s,%s,%s)"
    values=(cust['customer_id'],cust['customer_name'],cust['email'],cust['phone'],cust['booking_id'])
    cur.execute(query,values)
    cur.fetchall()
```

```
def get_all_customers(self):
    query="select * from customer"
    cur.execute(query)
    return cur.fetchall()

def unique_customer_id(self):
    return len(self.get_all_customers())+1
```

Book tickets:

```
def book_tickets(self,num_tickets:int):

    self.get_event_details()

    event_name = (input("enter the event name"))
    try:
        query="select event_name from event where event_name=%s"
        cur.execute(query,(event_name,))
        event=cur.fetchone()
        if not event:
            print(f"Error: {event_name} not found")
            return None
        self.create_customer()

        customer_id = self.unique_customer_id()
        event_id = input("enter the event id")
        num_ticket = num_tickets
        query = "select ticket_price from event where event_id=%s "
        cur.execute(query, (event_id,))
        res = cur.fetchone()
        res = res[0]
        total_cost = int(num_tickets) * res
        booking_date = self.get_current_date()
        booking_id = self.unique_booking_id()
```

```
    book = {
        'booking_id': booking_id,
        'customer_id': customer_id,
        'event_id': event_id,
        'num_tickets': num_ticket,
        'total_cost': total_cost,
        'booking_date': booking_date
    }
    sql = "insert into booking values(%s,%s,%s,%s,%s,%s)"
    values = (book['booking_id'], book['customer_id'], book['event_id'],
              book['num_tickets'], book['total_cost'],book['booking_date'])
    cur.execute(sql, values)
    cur.fetchall()
    sql1="select avaliable_seats from event where event_name= %s"
    cur.execute(sql1,(event_name,))
    results = cur.fetchone()
    results=results[0]
    results=int(results)
    if int(num_tickets) < results:
        results= int(num_tickets)-results
        print(f"Booked {num_tickets} tickets. Available seats: {results}")
    else:
        print("Not enough available seats for the requested number of tickets.")
finally:
    con.commit()
    con.commit()
```

ystemRepository › book_tickets() › try

```
def get_all_bookings(self):
    query = "select * from booking"
    cur.execute(query)
    return cur.fetchall()

def unique_booking_id(self):
    return len(self.get_all_bookings()) + 1
```

Output:

Book_tickets:

```
1.create Event
2.book_tickets
3.cancel tickets
4.get avaliable tickets
5.get event details
6.exit
select from above options: 2
enter the num_tickets5
(1, 'Bollywood Night Extravaganza', datetime.date(2024, 6, 17), datetime.timedelta(seconds=66600), 1, 50, 47, Decimal('250.00'), 'Concert', 101)
(2, 'Cricket Tournament Finals world cup', datetime.date(2024, 5, 20), datetime.timedelta(seconds=54000), 2, 15500, 15480, Decimal('2400.00'), 'Sports', 102)
(3, 'Classic Movie Marathon', datetime.date(2024, 7, 21), datetime.timedelta(seconds=43200), 3, 100, 95, Decimal('50.00'), 'Movie', 103)
(4, 'Live Stand-Up Comedy Show', datetime.date(2024, 5, 22), datetime.timedelta(seconds=68400), 4, 50, 45, Decimal('120.00'), 'Concert', 104)
(5, 'Football Championship', datetime.date(2024, 5, 1), datetime.timedelta(seconds=59400), 5, 100, 100, Decimal('80.00'), 'Sports', None)
(6, 'Melodious Sufi Night', datetime.date(2024, 8, 10), datetime.timedelta(seconds=72000), 6, 50, 49, Decimal('200.00'), 'Concert', 105)
(7, 'Outdoor Movie Night', datetime.date(2024, 6, 25), datetime.timedelta(seconds=70200), 7, 50, 45, Decimal('75.00'), 'Movie', 106)
(8, 'Basketball Showdown', datetime.date(2024, 7, 15), datetime.timedelta(seconds=50400), 8, 150, 145, Decimal('60.00'), 'Sports', 107)
(9, 'Folk Music Festival', datetime.date(2024, 9, 3), datetime.timedelta(seconds=61200), 9, 50, 47, Decimal('150.00'), 'Concert', 108)
(10, 'Tennis Championship asian cup', datetime.date(2024, 9, 11), datetime.timedelta(seconds=45000), 10, 200, 190, Decimal('1000.00'), 'Sports', 109)
(11, 'ipl', datetime.date(2024, 2, 4), datetime.timedelta(seconds=36553), None, 500, None, Decimal('1500.00'), 'sports', None)
enter the event nameFootball Championship
enter your nameshiv
enter your emailshiv@gmail.com
enter your phone number125879
enter the event id5
Booked 5 tickets. Available seats: -95
```

Customer table:

```
(1, 'Aditya Verma', 'aditya@gmail.com', '1234567890', 101)
(2, 'Sneha Patel', 'sneha@gmail.com', '1234567891', 102)
(3, 'Aryan Singh', 'aryan@gmail.com', '1234567892', 103)
(4, 'Kavita Gupta', 'kavita@gmail.com', '1234567893', 104)
(5, 'Rahul Sharma', 'rahul@gmail.com', '1234567894', None)
(6, 'Priya Mishra', 'priya@gmail.com', '1234567895', 105)
(7, 'Vikram Malhotra', 'vikram@gmail.com', '1234567896', 106)
(8, 'Anjali sharma', 'anjali@gmail.com', '1234567000', 107)
(9, 'Neha Kapoor', 'neha@gmail.com', '1234567898', 108)
(10, 'Ravi Verma', 'ravi@gmail.com', '1134567000', 109)
(11, 'akash devaki', 'akash@gmail.com', '1478523691', None)
(12, 'sai teja', 'sai@gmail.com', '1478523692', None)
(13, 'ram', 'ram@gmail.com', '123457', 10)
(14, 'shiv', 'shiv@gmail.com', '125879', 11)
```

Booking Table:

```
(10, 13, 5, 5, Decimal('400.00'), datetime.date(2024, 2, 4))
(11, 14, 5, 5, Decimal('400.00'), datetime.date(2024, 2, 4))
(101, 1, 1, 3, Decimal('750.00'), datetime.date(2024, 1, 10))
(102, 2, 2, 20, Decimal('48000.00'), datetime.date(2024, 1, 12))
(103, 3, 3, 5, Decimal('250.00'), datetime.date(2024, 1, 15))
(104, 4, 4, 5, Decimal('600.00'), datetime.date(2024, 1, 17))
(105, 6, 6, 1, Decimal('200.00'), datetime.date(2024, 2, 20))
(106, 7, 7, 5, Decimal('375.00'), datetime.date(2024, 2, 25))
(107, 8, 8, 5, Decimal('300.00'), datetime.date(2024, 3, 5))
(108, 9, 9, 3, Decimal('450.00'), datetime.date(2024, 3, 7))
(109, 10, 10, 10, Decimal('10000.00'), datetime.date(2024, 3, 10))
```

cancel_tickets:

```
def cancel_tickets(self):
    try:
        avaliable_seats=100
        booking_id=int(input("enter the booking_id"))
        query="select booking_id from booking where booking_id=%s"
        cur.execute(query,(booking_id,))
        book=cur.fetchone()
        if not book:
            print(f"Error {booking_id} not found")
            return None
        sql = "select num_tickets from booking where booking_id=%s "
        cur.execute(sql, (booking_id,))
        num_tickets = cur.fetchone()
        num_tickets = num_tickets[0]
        avaliable_seats += num_tickets
        query="delete from booking where booking_id=%s "
        cur.execute(query,(booking_id,))
        qu = "delete from customer where booking_id=%s "
        cur.execute(qu, (booking_id,))


        print(f"after canceling {num_tickets} the avaliable tickets are {avaliable_seats}")
    finally:
        con.commit()
        con.close()
```

Output:

```
1.create Event
2.book_tickets
3.cancel tickets
4.get avaliable tickets
5.get event details
6.exit
select from above options: 3
enter the booking_id10
after canceling 5 the avaliable tickets are 105
```

Booking_table:

```
(11, 14, 5, 5, Decimal('400.00'), datetime.date(2024, 2, 4))
(101, 1, 1, 3, Decimal('750.00'), datetime.date(2024, 1, 10))
(102, 2, 2, 20, Decimal('48000.00'), datetime.date(2024, 1, 12))
(103, 3, 3, 5, Decimal('250.00'), datetime.date(2024, 1, 15))
(104, 4, 4, 5, Decimal('600.00'), datetime.date(2024, 1, 17))
(105, 6, 6, 1, Decimal('200.00'), datetime.date(2024, 2, 20))
(106, 7, 7, 5, Decimal('375.00'), datetime.date(2024, 2, 25))
(107, 8, 8, 5, Decimal('300.00'), datetime.date(2024, 3, 5))
(108, 9, 9, 3, Decimal('450.00'), datetime.date(2024, 3, 7))
(109, 10, 10, 10, Decimal('10000.00'), datetime.date(2024, 3, 10))
```

customer table:

```
(1, 'Aditya Verma', 'aditya@gmail.com', '1234567890', 101)
(2, 'Sneha Patel', 'sneha@gmail.com', '1234567891', 102)
(3, 'Aryan Singh', 'aryan@gmail.com', '1234567892', 103)
(4, 'Kavita Gupta', 'kavita@gmail.com', '1234567893', 104)
(5, 'Rahul Sharma', 'rahul@gmail.com', '1234567894', None)
(6, 'Priya Mishra', 'priya@gmail.com', '1234567895', 105)
(7, 'Vikram Malhotra', 'vikram@gmail.com', '1234567896', 106)
(8, 'Anjali sharma', 'anjali@gmail.com', '1234567000', 107)
(9, 'Neha Kapoor', 'neha@gmail.com', '1234567898', 108)
(10, 'Ravi Verma', 'ravi@gmail.com', '1134567000', 109)
(11, 'akash devaki', 'akash@gmail.com', '1478523691', None)
(12, 'sai teja', 'sai@gmail.com', '1478523692', None)
(14, 'shiv', 'shiv@gmail.com', '125879', 11)
```

Exit:

```
1.create Event
2.book_tickets
3.cancel tickets
4.get avaliable tickets
5.get event details
6.exit
select from above options: 6
exiting the system
 Thank you
```

Submitted By:

Devaki Akash