Assignment Submission

Name: Mukund Suresh Sutar

Email: mukundmoto89@gmail.com

GitHub Link:
https://github.com/mukkund05/TicketBookingSystem

Assignment Name : Ticket Booking System

TASK – 1

1. Create the database named "TicketBookingSystem"

```
CREATE DATABASE TicketBookingSystem;
USE TicketBookingSystem;
```

2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. – Venu, Event, Customers, Booking

   1) Venue

```
CREATE TABLE Venue(
venue_id INT PRIMARY KEY,
venue_name VARCHAR(50),
address TEXT
);
```

2) Event

```
CREATE TABLE Event(
event_id INT PRIMARY KEY,
event_name varchar(50),
event_date DATE,
event_time TIME,
venue_id INT,
total_seats INT,
available_seats INT,
ticket_price DECIMAL,
event_type ENUM ('Movie', 'Sports', 'Concert'),
booking_id INT,
FOREIGN KEY (venue_id) REFERENCES Venue(venue_id)
);
```

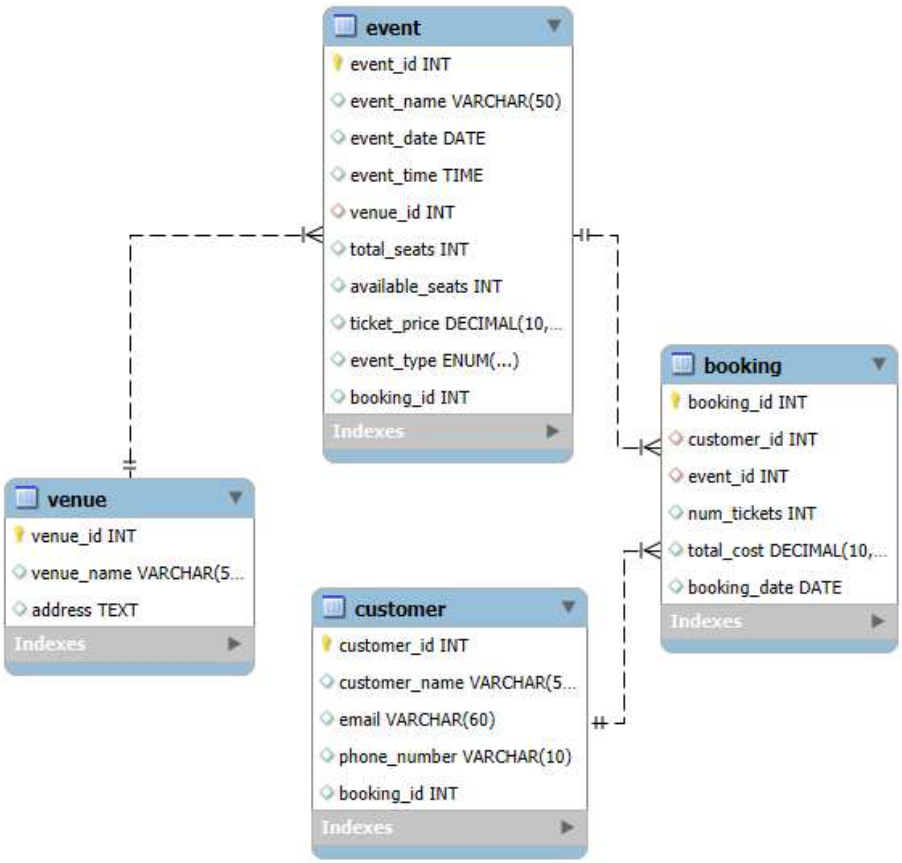
3) Customer

```
CREATE TABLE Customer(
customer_id INT PRIMARY KEY,
customer_name VARCHAR(50),
email VARCHAR(60),
phone_number VARCHAR(10),
booking_id INT
);
```

4) Booking

```
CREATE TABLE Booking(
booking_id INT PRIMARY KEY,
customer_id INT,
event_id INT,
num_tickets INT,
total_cost DECIMAL,
booking_date DATE,
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
FOREIGN KEY (event_id) REFERENCES Event(event_id)
);
```

3. Create an ERD (Entity Relationship Diagram) for the database.



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

```
FOREIGN KEY (venue_id) REFERENCES Venue(venue_id)

FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
FOREIGN KEY (event_id) REFERENCES Event(event_id)
```

TASK – 2

1. Write a SQL query to insert at least 10 sample records into each table.

```
INSERT INTO Venue (venue_id, venue_name, address) VALUES
(1, 'India Habitat Centre', 'Lodhi Road, New Delhi'),
(2, 'Jawaharlal Nehru Stadium', 'New Delhi'),
(3, 'Bandra Kurla Complex', 'Mumbai, Maharashtra'),
(4, 'Nehru Indoor Stadium', 'Chennai, Tamil Nadu'),
(5, 'Gachibowli Stadium', 'Hyderabad, Telangana'),
(6, 'Biswa Bangla Convention Centre', 'Kolkata, West Bengal'),
(7, 'Codissia Trade Fair Complex', 'Coimbatore, Tamil Nadu'),
(8, 'Sardar Patel Stadium', 'Ahmedabad, Gujarat'),
(9, 'The Lalit Ashok', 'Bangalore, Karnataka'),
(10, 'Chowdiah Memorial Hall', 'Bangalore, Karnataka');
```

```
INSERT INTO Event (
  event_id, event_name, event_date, event_time, venue_id,
  total_seats, available_seats, ticket_price, event_type, booking_id
) VALUES
(1, 'Indie Film Cup', '2025-05-05', '18:00:00', 1, 200, 50, 700.00, 'Movie', 101),
(2, 'IPL Final Match', '2025-05-20', '19:30:00', 2, 50000, 12000, 2200.00, 'Sports', 102),
(3, 'Classical Night', '2025-05-10', '20:00:00', 10, 800, 400, 1500.00, 'Concert', 103),
(4, 'FIFA World Cup Screening', '2025-05-15', '21:00:00', 3, 300, 0, 1800.00, 'Sports', 104),
(5, 'Bollywood Beats', '2025-05-18', '19:00:00', 5, 1500, 450, 1600.00, 'Concert', 105),
(6, 'Documentary Premiere', '2025-06-01', '17:00:00', 6, 250, 200, 850.00, 'Movie', 106),
(7, 'Music Cup Live', '2025-06-05', '20:30:00', 4, 5000, 1000, 2000.00, 'Concert', 107),
(8, 'Xtreme Wrestling', '2025-04-28', '18:30:00', 7, 10000, 3000, 950.00, 'Sports', 108),
(9, 'Zodiac Rock Fest', '2025-04-30', '19:00:00', 8, 12000, 12000, 1300.00, 'Concert', 109),
(10, 'Eco Documentary', '2025-05-02', '16:00:00', 9, 100, 0, 500.00, 'Movie', 110);
```

```
INSERT INTO Customer (
  customer_id, customer_name, email, phone_number, booking_id
) VALUES
(1, 'Aarav Mehta', 'aarav@gmail.com', '9123450000', 101),
(2, 'Sneha Kapoor', 'sneha@yahoo.com', '9876543210', 102),
(3, 'Raj Nair', 'raj.nair@live.com', '9988776655', 103),
(4, 'Priya Desai', 'priya@gmail.com', '9090909090', 104),
(5, 'Vikram Joshi', 'vikram@outlook.com', '9811122233', 105),
(6, 'Anjali Verma', 'anjali@gmail.com', '9870011223', 106),
(7, 'Karan Singh', 'karan@rediffmail.com', '9753124680', 107),
(8, 'Meera Iyer', 'meera.iyer@gmail.com', '9845098450', 108),
(9, 'Rohan Das', 'rohan.das@gmail.com', '9900990099', 109),
(10, 'Tanya Verma', 'tanya.verma@yahoo.in', '9823456000', 110);
```

```
INSERT INTO Booking (
  booking_id, customer_id, event_id, num_tickets, total_cost, booking_date
) VALUES
(101, 1, 1, 3, 2100.00, '2025-04-01'),
(102, 2, 2, 5, 11000.00, '2025-04-02'),
(103, 3, 3, 2, 3000.00, '2025-04-03'),
(104, 4, 4, 4, 7200.00, '2025-04-04'),
(105, 5, 5, 3, 4800.00, '2025-04-05'),
(106, 6, 6, 1, 850.00, '2025-04-06'),
(107, 7, 7, 6, 12000.00, '2025-04-07'),
(108, 8, 8, 2, 1900.00, '2025-04-08'),
(109, 9, 9, 4, 5200.00, '2025-04-08'),
(110, 10, 10, 1, 500.00, '2025-04-09');
```

2. Write a SQL query to list all Events.

```
select * from event;
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Indie Film Cup	2025-05-05	18:00:00	1	200	50	700	Movie	101
2	IPL Final Match	2025-05-20	19:30:00	2	50000	12000	2200	Sports	102
3	Classical Night	2025-05-10	20:00:00	10	800	400	1500	Concert	103
4	FIFA World Cup Screening	2025-05-15	21:00:00	3	300	0	1800	Sports	104
5	Bollywood Beats	2025-05-18	19:00:00	5	1500	450	1600	Concert	105
6	Documentary Premiere	2025-06-01	17:00:00	6	250	200	850	Movie	106
7	Music Cup Live	2025-06-05	20:30:00	4	5000	1000	2000	Concert	107
8	Xtreme Wrestling	2025-04-28	18:30:00	7	10000	3000	950	Sports	108
9	Zodiac Rock Fest	2025-04-30	19:00:00	8	12000	12000	1300	Concert	109
10	Eco Documentary	2025-05-02	16:00:00	9	100	0	500	Movie	110
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Write a SQL query to select events with available tickets.

```
select * from event
where available_seats > 0;
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Indie Film Cup	2025-05-05	18:00:00	1	200	50	700	Movie	101
2	IPL Final Match	2025-05-20	19:30:00	2	50000	12000	2200	Sports	102
3	Classical Night	2025-05-10	20:00:00	10	800	400	1500	Concert	103
5	Bollywood Beats	2025-05-18	19:00:00	5	1500	450	1600	Concert	105
6	Documentary Premiere	2025-06-01	17:00:00	6	250	200	850	Movie	106
7	Music Cup Live	2025-06-05	20:30:00	4	5000	1000	2000	Concert	107
8	Xtreme Wrestling	2025-04-28	18:30:00	7	10000	3000	950	Sports	108
9	Zodiac Rock Fest	2025-04-30	19:00:00	8	12000	12000	1300	Concert	109
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4. Write a SQL query to select events name partial match with ‘cup’.

```
select * from event
where event_name like "%cup%";
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
1	Indie Film Cup	2025-05-05	18:00:00	1	200	50	700	Movie	101
4	FIFA World Cup Screening	2025-05-15	21:00:00	3	300	0	1800	Sports	104
7	Music Cup Live	2025-06-05	20:30:00	4	5000	1000	2000	Concert	107
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5. Write a SQL query to select events with ticket price range is between 1000 to 2500.

```
select * from event
where ticket_price
between 1000 and 2500;
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
2	IPL Final Match	2025-05-20	19:30:00	2	50000	12000	2200	Sports	102
3	Classical Night	2025-05-10	20:00:00	10	800	400	1500	Concert	103
4	FIFA World Cup Screening	2025-05-15	21:00:00	3	300	0	1800	Sports	104
5	Bollywood Beats	2025-05-18	19:00:00	5	1500	450	1600	Concert	105
7	Music Cup Live	2025-06-05	20:30:00	4	5000	1000	2000	Concert	107
9	Zodiac Rock Fest	2025-04-30	19:00:00	8	12000	12000	1300	Concert	109
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6. Write a SQL query to retrieve events with dates falling within a specific range.

```
select * from event
where event_date
between "2025-05-01" and "2025-05-20"
order by event_date asc;
```

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id
10	Eco Documentary	2025-05-02	16:00:00	9	100	0	500	Movie	110
1	Indie Film Cup	2025-05-05	18:00:00	1	200	50	700	Movie	101
3	Classical Night	2025-05-10	20:00:00	10	800	400	1500	Concert	103
4	FIFA World Cup Screening	2025-05-15	21:00:00	3	300	0	1800	Sports	104
5	Bollywood Beats	2025-05-18	19:00:00	5	1500	450	1600	Concert	105
2	IPL Final Match	2025-05-20	19:30:00	2	50000	12000	2200	Sports	102
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.

```
select  event_name, available_seats from event
where available_seats > 0 AND event_type = "Concert";
```

	event_name	available_seats
▶	Classical Night	400
	Bollywood Beats	450
	Music Cup Live	1000
	Zodiac Rock Fest	12000

8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.

```
SELECT *
FROM Customer
ORDER BY customer_id
LIMIT 5 OFFSET 5;
```

	customer_id	customer_name	email	phone_number	booking_id
▶	6	Anjali Verma	anjali@gmail.com	9870011223	106
	7	Karan Singh	karan@rediffmail.com	9753124680	107
	8	Meera Iyer	meera.iyer@gmail.com	9845098450	108
	9	Rohan Das	rohan.das@gmail.com	9900990099	109
	10	Tanya Verma	tanya.verma@yahoo.in	9823456000	110
*	NULL	NULL	NULL	NULL	NULL

9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.

```
select c.booking_id, c.customer_name, b.num_tickets, b.total_cost
from customer c
join booking b
on c.booking_id = b.booking_id
where b.Num_tickets > 4;
```

	booking_id	customer_name	num_tickets	total_cost
▶	102	Sneha Kapoor	5	11000
	107	Karan Singh	6	12000
	110	Tanya Verma	5	500

10. Write a SQL query to retrieve customer information whose phone number end with ‘000’

```
select * from customer
where phone_number like "%000";
```

customer_id	customer_name	email	phone_number	booking_id
1	Aarav Mehta	aarav@gmail.com	9123450000	101
10	Tanya Verma	tanya.verma@yahoo.in	9823456000	110
NULL	NULL	NULL	NULL	NULL

11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.

```
select event_id, event_name, total_seats from event
where total_seats > 15000;
```

event_id	event_name	total_seats
2	IPL Final Match	50000
NULL	NULL	NULL

12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’

```
select event_name from event
where event_name not like "x%"
and event_name not like "y%"
and event_name not like "z%";
```

event_name
Indie Film Cup
IPL Final Match
Classical Night
FIFA World Cup Screening
Bollywood Beats
Documentary Premiere
Music Cup Live
Eco Documentary

13. Write a SQL query to List Events and Their Average Ticket Prices.

```
select event_name, avg(ticket_price) as averagePrice from event
group by event_name;
```

	event_name	averagePrice
▶	Indie Film Cup	700.0000
	IPL Final Match	2200.0000
	Classical Night	1500.0000
	FIFA World Cup Screening	1800.0000
	Bollywood Beats	1600.0000
	Documentary Premiere	850.0000
	Music Cup Live	2000.0000
	Xtreme Wrestling	950.0000
	Zodiac Rock Fest	1300.0000
	Eco Documentary	500.0000

14. Write a SQL query to Calculate the Total Revenue Generated by Events.

```
select e.event_id ,e.event_name, sum(b.total_cost) as TotalRevenue from event e
join booking b
on e.booking_id = b.booking_id
group by e.event_id;
```

	event_id	event_name	TotalRevenue
▶	1	Indie Film Cup	2100
	2	IPL Final Match	11000
	3	Classical Night	3000
	4	FIFA World Cup Screening	7200
	5	Bollywood Beats	4800
	6	Documentary Premiere	850
	7	Music Cup Live	12000
	8	Xtreme Wrestling	1900
	9	Zodiac Rock Fest	5200
	10	Eco Documentary	500

15. Write a SQL query to find the event with the highest ticket sales.

```
select e.event_id,e.event_name, b.num_tickets from event e
join booking b
on b.event_id = e.event_id
order by num_tickets desc
limit 1;
```

	event_id	event_name	num_tickets
▶	7	Music Cup Live	6

16. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.

```
select b.event_id, e.event_name ,sum(b.num_tickets) as TotalSoldTickets from booking b
join event e
on e.event_id = b.event_id
group by event_id;
```

event_id	event_name	TotalSoldTickets
1	Indie Film Cup	3
2	IPL Final Match	5
3	Classical Night	2
4	FIFA World Cup Screening	4
5	Bollywood Beats	3
6	Documentary Premiere	1
7	Music Cup Live	6
8	Xtreme Wrestling	2
9	Zodiac Rock Fest	4
10	Eco Documentary	5

17. Write a SQL query to Find Events with No Ticket Sales.

```
select b.event_id, e.event_name , b.num_tickets from booking b
join event e
on e.event_id = b.event_id
where b.num_tickets =0;
```

event_id	event_name	num_tickets

18. Write a SQL query to Find the User Who Has Booked the Most Tickets.

```
select c.customer_id, c.customer_name, b.num_tickets  from customer c
join booking b
on c.customer_id = b.customer_id
order by b.num_tickets desc
limit 1;
```

	customer_id	customer_name	num_tickets
▶	7	Karan Singh	6

19. Write a SQL query to List Events and the total number of tickets sold for each month.

```
select e.event_name, MONTH(b.booking_date) AS booking_month, sum(b.num_tickets) from event e
join booking b
on b.event_id = e.event_id
group by e.event_name, month(b.booking_date)
order by booking_month, e.event_name;
```

	event_name	booking_month	sum(b.num_tickets)
▶	Bollywood Beats	4	3
	Classical Night	4	2
	Documentary Premiere	4	1
	Eco Documentary	4	5
	FIFA World Cup Screening	4	4
	Indie Film Cup	4	3
	IPL Final Match	4	5
	Music Cup Live	4	6
	Xtreme Wrestling	4	2
	Zodiac Rock Fest	4	4

20. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.

```
select  e.venue_id, v.venue_name ,avg(e.ticket_price) from event e
join venue v
on v.venue_id = e.venue_id
group by venue_id;
```

	venue_id	venue_name	avg(e.ticket_price)
▶	1	India Habitat Centre	700.0000
	2	Jawaharlal Nehru Stadium	2200.0000
	10	Chowdiah Memorial Hall	1500.0000
	3	Bandra Kurla Complex	1800.0000
	5	Gachibowli Stadium	1600.0000
	6	Biswa Bangla Convention Centre	850.0000
	4	Nehru Indoor Stadium	2000.0000
	7	Codissia Trade Fair Complex	950.0000
	8	Sardar Patel Stadium	1300.0000
	9	The Lalit Ashok	500.0000

21. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.

```
select  e.event_type, sum(b.num_tickets) as totalSoldTickets from booking b
join event e
on e.booking_id = b.booking_id
group by e.event_type;
```

	event_type	totalSoldTickets
▶	Movie	9
	Sports	11
	Concert	15

22. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.

```
select e.event_name, sum(b.total_cost) as totalRevenue, year(b.booking_date) as bookingYear
from booking b
join event e
on e.booking_id = b.booking_id
group by year(b.booking_date), e.event_name;
```

	event_name	totalRevenue	bookingYear
▶	Indie Film Cup	2100	2025
	IPL Final Match	11000	2025
	Classical Night	3000	2025
	FIFA World Cup Screening	7200	2025
	Bollywood Beats	4800	2025
	Documentary Premiere	850	2025
	Music Cup Live	12000	2025
	Xtreme Wrestling	1900	2025
	Zodiac Rock Fest	5200	2025
	Eco Documentary	500	2025

23. Write a SQL query to list users who have booked tickets for multiple events.

```
select  c.customer_id, c.customer_name, COUNT(distinct b.event_id) as event_count
from  Customer c
join Booking b
on c.customer_id = b.customer_id
group by c.customer_id, c.customer_name
having COUNT(distinct b.event_id) > 1;
```

	customer_id	customer_name	event_count

24. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.

```
select c.customer_id, c.customer_name, sum(b.total_cost) as TotalRevenue from customer c
join booking b
on b.customer_id = c.customer_id
group by
customer_id, customer_name;
```

customer_id	customer_name	TotalRevenue
1	Aarav Mehta	2100
2	Sneha Kapoor	11000
3	Raj Nair	3000
4	Priya Desai	7200
5	Vikram Joshi	4800
6	Anjali Verma	850
7	Karan Singh	12000
8	Meera Iyer	1900
9	Rohan Das	5200
10	Tanya Verma	500

25. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.

```
select e.event_type, v.venue_name, avg(e.ticket_price) as AverageTicketPrice from event e
join venue v
on e.venue_id = v.venue_id
group by v.venue_name ,e.event_type;
```

event_type	venue_name	AverageTicketPrice
Movie	India Habitat Centre	700.0000
Sports	Jawaharlal Nehru Stadium	2200.0000
Concert	Chowdiah Memorial Hall	1500.0000
Sports	Bandra Kurla Complex	1800.0000
Concert	Gachibowli Stadium	1600.0000
Movie	Biswa Bangla Convention Centre	850.0000
Concert	Nehru Indoor Stadium	2000.0000
Sports	Codissia Trade Fair Complex	950.0000
Concert	Sardar Patel Stadium	1300.0000
Movie	The Lalit Ashok	500.0000

26. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.

```
select c.customer_id, c.customer_name, sum(b.num_tickets) as total_tickets_purchased
, b.booking_date from customer c
join booking b
on c.booking_id = b.booking_id
where b.booking_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
group by c.customer_id, c.customer_name
order by customer_id;
```

	customer_id	customer_name	total_tickets_purchased	booking_date
▶	1	Aarav Mehta	3	2025-04-01
	2	Sneha Kapoor	5	2025-04-02
	3	Raj Nair	2	2025-04-03
	4	Priya Desai	4	2025-04-04
	5	Vikram Joshi	3	2025-04-05
	6	Anjali Verma	1	2025-04-06
	7	Karan Singh	6	2025-04-07
	8	Meera Iyer	2	2025-04-08
	9	Rohan Das	4	2025-04-08
	10	Tanya Verma	5	2025-04-09

27. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.

```
select v.venue_id, v.venue_name, (select avg(e.ticket_price) from event e
where e.venue_id = v.venue_id) as avgTicketPrice
from venue v;
```

	venue_id	venue_name	avgTicketPrice
▶	1	India Habitat Centre	700.0000
	2	Jawaharlal Nehru Stadium	2200.0000
	3	Bandra Kurla Complex	1800.0000
	4	Nehru Indoor Stadium	2000.0000
	5	Gachibowli Stadium	1600.0000
	6	Biswa Bangla Convention Centre	850.0000
	7	Codissia Trade Fair Complex	950.0000
	8	Sardar Patel Stadium	1300.0000
	9	The Lalit Ashok	500.0000
	10	Chowdiah Memorial Hall	1500.0000

28. Find Events with More Than 50% of Tickets Sold using subquery.

```
SELECT e.event_id, e.event_name
FROM Event e
WHERE
(SELECT SUM(b.num_tickets) FROM Booking b WHERE b.event_id = e.event_id) / e.total_seats > 0.5;
```

	event_id	event_name
*	NULL	NULL

29. Calculate the Total Number of Tickets Sold for Each Event.

```
select e.event_id, e.event_name,
(select sum(b.num_tickets) from booking b
where e.booking_id = b.booking_id) as TotalSoldTickets
from event e;
```

	event_id	event_name	TotalSoldTickets
▶	1	Indie Film Cup	3
	2	IPL Final Match	5
	3	Classical Night	2
	4	FIFA World Cup Screening	4
	5	Bollywood Beats	3
	6	Documentary Premiere	1
	7	Music Cup Live	6
	8	Xtreme Wrestling	2
	9	Zodiac Rock Fest	4
	10	Eco Documentary	5

30. Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.

```
select c.customer_id, c.customer_name from customer c where
not exists  (select 1 from booking b where b.booking_id = c.booking_id);
```

	customer_id	customer_name
*	NULL	NULL

31. List Events with No Ticket Sales Using a NOT IN Subquery.

```
SELECT event_id, event_name
FROM Event
WHERE event_id NOT IN (SELECT event_id FROM Booking);
```

	event_id	event_name
*	NULL	NULL

32. Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause.

```
SELECT e.event_type, SUM(b.num_tickets) AS total_tickets_sold
FROM (SELECT event_id, event_type FROM Event) AS e
JOIN Booking b ON e.event_id = b.event_id
GROUP BY e.event_type;
```

	event_type	total_tickets_sold
▶	Movie	9
	Sports	11
	Concert	15

33. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause

```
SELECT event_name, ticket_price
FROM Event
WHERE ticket_price > (SELECT AVG(ticket_price) FROM Event);
```

event_name	ticket_price
IPL Final Match	2200
Classical Night	1500
FIFA World Cup Screening	1800
Bollywood Beats	1600
Music Cup Live	2000

34. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.

```
SELECT customer_name,
(SELECT SUM(total_cost) FROM Booking WHERE Booking.customer_id = Customer.customer_id) AS total_revenue
FROM Customer;
```

customer_name	total_revenue
Aarav Mehta	2100
Sneha Kapoor	11000
Raj Nair	3000
Priya Desai	7200
Vikram Joshi	4800
Anjali Verma	850
Karan Singh	12000
Meera Iyer	1900
Rohan Das	5200
Tanya Verma	500

35. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.

```
SELECT customer_name
FROM Customer
WHERE customer_id IN (SELECT customer_id FROM Booking WHERE event_id IN (SELECT event_id FROM Event WHERE venue_id = 1));
```

customer_name
Aarav Mehta

36. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY.

```
SELECT event_type, SUM(num_tickets) AS total_tickets_sold
FROM Event
JOIN Booking ON Event.event_id = Booking.event_id
GROUP BY event_type;
```

	event_type	total_tickets_sold
▶	Movie	9
	Sports	11
	Concert	15

37. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT.

```
SELECT customer_name
FROM Customer
WHERE customer_id IN (SELECT customer_id FROM Booking WHERE DATE_FORMAT(booking_date, '%Y-%m') = '2025-04');
```

	customer_name
▶	Aarav Mehta
	Sneha Kapoor
	Raj Nair
	Priya Desai
	Vikram Joshi
	Anjali Verma
	Karan Singh
	Meera Iyer
	Rohan Das
	Tanya Verma

38. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery

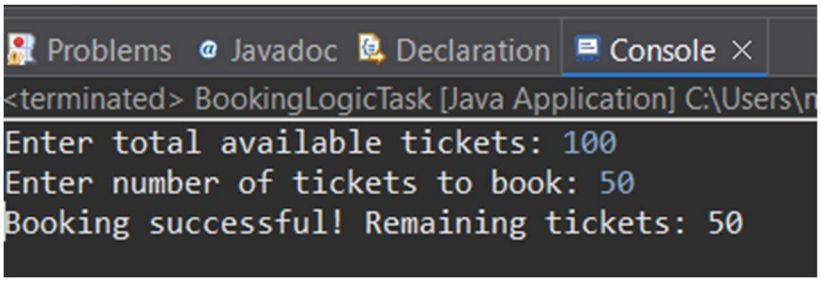
```
SELECT venue_name,
(SELECT AVG(ticket_price) FROM Event WHERE Event.venue_id = Venue.venue_id) AS avg_ticket_price
FROM Venue;
```

	venue_name	avg_ticket_price
▶	India Habitat Centre	700.0000
	Jawaharlal Nehru Stadium	2200.0000
	Bandra Kurla Complex	1800.0000
	Nehru Indoor Stadium	2000.0000
	Gachibowli Stadium	1600.0000
	Biswa Bangla Convention Centre	850.0000
	Codissia Trade Fair Complex	950.0000
	Sardar Patel Stadium	1300.0000
	The Lalit Ashok	500.0000
	Chowdiah Memorial Hall	1500.0000

Task 1: Conditional Statements

1. Write a program that takes the availableTicket and noOfBookingTicket as input.

```java
package com.hexaware.main;

import java.util.Scanner;

public class BookingLogicTask {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter total available tickets: ");
        int availableTickets = sc.nextInt();

        System.out.print("Enter number of tickets to book: ");
        int ticketsToBook = sc.nextInt();

        if (ticketsToBook <= availableTickets) {
            int remaining = availableTickets - ticketsToBook;
            System.out.println("Booking successful! Remaining tickets: " + remaining);
        } else {
            System.out.println("Booking failed! Not enough tickets available.");
        }

        sc.close();
    }
}
```



2. Use conditional statements (if-else) to determine if the ticket is available or not.

```java
package com.hexaware.main;

import java.util.Scanner;

public class TicketBookingNestedTask {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Available Ticket Types: Silver, Gold, Diamond");
        System.out.print("Enter ticket type: ");
        String ticketType = sc.nextLine().toLowerCase();

        System.out.print("Enter number of tickets to book: ");
        int numTickets = sc.nextInt();

        double ticketPrice = 0;

        if (ticketType.equals("silver")) {
            ticketPrice = 500;
        } else if (ticketType.equals("gold")) {
            ticketPrice = 1000;
        } else if (ticketType.equals("diamond")) {
            ticketPrice = 2000;
        } else {
            System.out.println("Invalid ticket type entered.");
            sc.close();
            return;
        }

        double totalCost = ticketPrice * numTickets;
        System.out.println("Booking Successful!");
        System.out.println("Total Cost: ₹" + totalCost);

        sc.close();
    }
}
```

```
Problems  Javadoc  Declaration  Console ×
<terminated> TicketBookingNestedTask [Java Application] C:\
Available Ticket Types: Silver, Gold, Diamond
Enter ticket type: Diamond
Enter number of tickets to book: 2
Booking Successful!
Total Cost: ₹4000.0
```

3. Display an appropriate message based on ticket availability.

```java
package com.hexaware.main;

import java.util.Scanner;

public class TicketBookingLoopingTask {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String continueBooking;

        do {
            System.out.println("\nAvailable Ticket Types: Silver, Gold, Diamond");
            System.out.print("Enter ticket type (or 'exit' to stop): ");
            String ticketType = sc.nextLine().toLowerCase();

            if (ticketType.equals("exit")) {
                break;
            }

            System.out.print("Enter number of tickets to book: ");
            int numTickets = sc.nextInt();
            sc.nextLine(); // consume newline

            double ticketPrice = 0;

            if (ticketType.equals("silver")) {
                ticketPrice = 500;
            } else if (ticketType.equals("gold")) {
                ticketPrice = 1000;
            } else if (ticketType.equals("diamond")) {
                ticketPrice = 2000;
            } else {
                System.out.println("Invalid ticket type entered.");
                continue;
            }

            double totalCost = ticketPrice * numTickets;
            System.out.println("Booking Successful!");
            System.out.println("Total Cost: ₹" + totalCost);

        } while (true);

        System.out.println("Thank you for using the Ticket Booking System!");
        sc.close();
    }
}
```

```
Problems  Javadoc  Declaration  Console ×
<terminated> TicketBookingLoopingTask [Java Application] C:\Users\mukku\.p2

Available Ticket Types: Silver, Gold, Diamond
Enter ticket type (or 'exit' to stop): Diamond
Enter number of tickets to book: 2
Booking Successful!
Total Cost: ₹4000.0

Available Ticket Types: Silver, Gold, Diamond
Enter ticket type (or 'exit' to stop): Silver
Enter number of tickets to book: 2
Booking Successful!
Total Cost: ₹1000.0

Available Ticket Types: Silver, Gold, Diamond
Enter ticket type (or 'exit' to stop): exit
Thank you for using the Ticket Booking System!
```

Create a Following classes with the following attributes and methods:

1. Event Class:

```java
package com.hexaware.entity;

public class Event {


    private String eventName;
    private String eventDate;
    private String eventTime;
    private String venueName;
    private int totalSeats;
    private int availableSeats;
    private double ticketPrice;
    private String eventType;

    public Event() {}

    public Event(String eventName, String eventDate, String eventTime, String venueName,
                 int totalSeats, int availableSeats, double ticketPrice, String eventType) {
        this.eventName = eventName;
        this.eventDate = eventDate;
        this.eventTime = eventTime;
        this.venueName = venueName;
        this.totalSeats = totalSeats;
        this.availableSeats = availableSeats;
        this.ticketPrice = ticketPrice;
        this.eventType = eventType;
    }

    public String getEventName() {
        return eventName;
    }

    public void setEventName(String eventName) {
        this.eventName = eventName;
    }

    public String getEventDate() {
        return eventDate;
    }

    public void setEventDate(String eventDate) {
        this.eventDate = eventDate;
    }

    public String getEventTime() {
        return eventTime;
    }

    public void setEventTime(String eventTime) {
        this.eventTime = eventTime;
    }

    public String getVenueName() {
        return venueName;
    }
```

```java
    public void setVenueName(String venueName) {
        this.venueName = venueName;
    }

    public int getTotalSeats() {
        return totalSeats;
    }

    public void setTotalSeats(int totalSeats) {
        this.totalSeats = totalSeats;
    }

    public int getAvailableSeats() {
        return availableSeats;
    }

    public void setAvailableSeats(int availableSeats) {
        this.availableSeats = availableSeats;
    }

    public double getTicketPrice() {
        return ticketPrice;
    }

    public void setTicketPrice(double ticketPrice) {
        this.ticketPrice = ticketPrice;
    }

    public String getEventType() {
        return eventType;
    }

    public void setEventType(String eventType) {
        this.eventType = eventType;
    }


    public double calculateTotalRevenue() {
        return (totalSeats - availableSeats) * ticketPrice;
    }

    public int getBookedNoOfTickets() {
        return totalSeats - availableSeats;
    }

    public void bookTickets(int numTickets) {
        if (numTickets <= availableSeats) {
            availableSeats -= numTickets;
        } else {
            System.out.println("Not enough seats available.");
        }
    }
```

```java
    public void cancelBooking(int numTickets) {
        availableSeats += numTickets;
    }

    public void displayEventDetails() {
        System.out.println("Event: " + eventName + " | Date: " + eventDate +
                " | Time: " + eventTime + " | Venue: " + venueName +
                " | Available Seats: " + availableSeats + "/" + totalSeats +
                " | Ticket Price: ₹" + ticketPrice + " | Type: " + eventType);
    }

}
```

2. Venue Class

```java
package com.hexaware.entity;

public class Venue {
    private String venueName;
    private String address;

    public Venue() {}

    public Venue(String venueName, String address) {
        this.venueName = venueName;
        this.address = address;
    }

    public String getVenueName() { return venueName; }
    public void setVenueName(String venueName) { this.venueName = venueName; }

    public String getAddress() { return address; }
    public void setAddress(String address) { this.address = address; }

    public void displayVenueDetails() {
        System.out.println("Venue: " + venueName + " | Address: " + address);
    }
}
```

3 Customer Class

```java
package com.hexaware.entity;

public class Customer {
    private String customerName;
    private String email;
    private String phoneNumber;

    public Customer() {}

    public Customer(String customerName, String email, String phoneNumber) {
        this.customerName = customerName;
        this.email = email;
        this.phoneNumber = phoneNumber;
    }

    public String getCustomerName() { return customerName; }
    public void setCustomerName(String customerName) { this.customerName = customerName; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public String getPhoneNumber() { return phoneNumber; }
    public void setPhoneNumber(String phoneNumber) { this.phoneNumber = phoneNumber; }

    public void displayCustomerDetails() {
        System.out.println("Customer: " + customerName + " | Email: " + email +
                " | Phone: " + phoneNumber);
    }
}
```

4 Booking

```java
package com.hexaware.entity;

public class Booking {
    private double totalCost;
    private Event event;

    public Booking() {}

    public Booking(Event event) {
        this.event = event;
    }

    public double calculateBookingCost(int numTickets) {
        this.totalCost = event.getTicketPrice() * numTickets;
        return totalCost;
    }

    public void bookTickets(int numTickets) {
        event.bookTickets(numTickets);
    }

    public void cancelBooking(int numTickets) {
        event.cancelBooking(numTickets);
    }

    public int getAvailableNoOfTickets() {
        return event.getAvailableSeats();
    }

    public void getEventDetails() {
        event.displayEventDetails();
    }

    public double getTotalCost() {
        return totalCost;
    }
}
```

Task 5: Inheritance and polymorphism

• Create a subclass **Movie** that inherits from Event. Add the following attributes and methods:

```java
package com.hexaware.entity;

public class Movie extends Event {
    private String genre;
    private String actorName;
    private String actressName;

    public Movie() {}

    public Movie(String eventName, String eventDate, String eventTime, String venueName,
                 int totalSeats, int availableSeats, double ticketPrice, String eventType,
                 String genre, String actorName, String actressName) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, ticketPrice, eventType);
        this.genre = genre;
        this.actorName = actorName;
        this.actressName = actressName;
    }

    public void displayEventDetails() {
        super.displayEventDetails();
        System.out.println("Genre: " + genre + " | Actor: " + actorName + " | Actress: " + actressName);
    }
}
```

• Create another subclass **Concert** that inherits from Event. Add the following attributes and methods:

```java
package com.hexaware.entity;

public class Concert extends Event {
    private String artist;
    private String type;

    public Concert() {}

    public Concert(String eventName, String eventDate, String eventTime, String venueName,
                   int totalSeats, int availableSeats, double ticketPrice, String eventType,
                   String artist, String type) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, ticketPrice, eventType);
        this.artist = artist;
        this.type = type;
    }

    public void displayConcertDetails() {
        super.displayEventDetails();
        System.out.println("Artist: " + artist + " | Type: " + type);
    }
}
```

• Create another subclass **Sports** that inherits from Event. Add the following attributes and methods:

```java
package com.hexaware.entity;

public class Sports extends Event {
    private String sportName;
    private String teamsName;

    public Sports() {}

    public Sports(String eventName, String eventDate, String eventTime, String venueName,
                  int totalSeats, int availableSeats, double ticketPrice, String eventType,
                  String sportName, String teamsName) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, ticketPrice, eventType);
        this.sportName = sportName;
        this.teamsName = teamsName;
    }

    public void displaySportDetails() {
        super.displayEventDetails();
        System.out.println("Sport: " + sportName + " | Teams: " + teamsName);
    }
}
```

• Create a class TicketBookingSystem with the following methods:

```java
package com.hexaware.main;

import com.hexaware.entity.*;

public class TicketBookingSystem {

    public Event createEvent(String type) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter event name: ");
        String name = sc.nextLine();

        System.out.print("Enter event date (YYYY-MM-DD): ");
        String date = sc.nextLine();

        System.out.print("Enter event time (HH:MM): ");
        String time = sc.nextLine();

        System.out.print("Enter venue name: ");
        String venue = sc.nextLine();

        System.out.print("Enter total seats: ");
        int totalSeats = sc.nextInt();

        System.out.print("Enter available seats: ");
        int availableSeats = sc.nextInt();

        System.out.print("Enter ticket price: ");
        double price = sc.nextDouble();
        sc.nextLine(); // consume newline

        if (type.equalsIgnoreCase("movie")) {
            System.out.print("Enter genre: ");
            String genre = sc.nextLine();

            System.out.print("Enter actor name: ");
            String actor = sc.nextLine();

            System.out.print("Enter actress name: ");
            String actress = sc.nextLine();
```

```
            return new Movie(name, date, time, venue, totalSeats, availableSeats, price, "Movie", genre, actor, actress);
        } else if (type.equalsIgnoreCase("concert")) {
            System.out.print("Enter artist name: ");
            String artist = sc.nextLine();

            System.out.print("Enter concert type (Rock/Classical/etc): ");
            String concertType = sc.nextLine();

            return new Concert(name, date, time, venue, totalSeats, availableSeats, price, "Concert", artist, concertType);
        } else if (type.equalsIgnoreCase("sports")) {
            System.out.print("Enter sport name: ");
            String sport = sc.nextLine();

            System.out.print("Enter teams (e.g., India vs Pakistan): ");
            String teams = sc.nextLine();

            return new Sports(name, date, time, venue, totalSeats, availableSeats, price, "Sports", sport, teams);
        } else {
            System.out.println("Invalid event type.");
            return null;
        }
    }

    public void displayEventDetails(Event event) {
        event.displayEventDetails(); // Polymorphic call
    }

    public void bookTickets(Event event, int numTickets) {
        if (event.getAvailableSeats() >= numTickets) {
            event.bookTickets(numTickets);
            double cost = event.getTicketPrice() * numTickets;
            System.out.println("Tickets booked successfully. Total Cost: ₹" + cost);
        } else {
            System.out.println("Sorry, not enough seats available.");
        }
    }

    public void cancelTickets(Event event, int numTickets) {
        event.cancelBooking(numTickets);
        System.out.println(numTickets + " tickets cancelled successfully.");
    }
```

```
    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem();
        Scanner sc = new Scanner(System.in);
        Event event = null;

        while (true) {
            System.out.println("\n--- Ticket Booking Menu ---");
            System.out.println("1. Create Event");
            System.out.println("2. Display Event Details");
            System.out.println("3. Book Tickets");
            System.out.println("4. Cancel Tickets");
            System.out.println("5. Exit");
            System.out.print("Choose an option: ");
            int choice = sc.nextInt();
            sc.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter event type (Movie/Concert/Sports): ");
                    String type = sc.nextLine();
                    event = system.createEvent(type);
                    break;
                case 2:
                    if (event != null) system.displayEventDetails(event);
                    else System.out.println("No event created yet.");
                    break;
                case 3:
                    if (event != null) {
                        System.out.print("Enter number of tickets to book: ");
                        int tickets = sc.nextInt();
                        sc.nextLine();
                        system.bookTickets(event, tickets);
                    } else {
                        System.out.println("No event available to book.");
                    }
                    break;
                case 4:
                    if (event != null) {
                        System.out.print("Enter number of tickets to cancel: ");
                        int cancel = sc.nextInt();
                        sc.nextLine();
                        system.cancelTickets(event, cancel);
                    } else {
                        System.out.println("No event available to cancel.");
                    }
                    break;
                case 5:
                    System.out.println("Thank you for using the Ticket Booking System.");
                    sc.close();
                    return;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
```

Task 6: Abstraction

1. Event Abstraction:

```java
package com.hexaware.entity;

public abstract class Event {
    protected String eventName;
    protected String eventDate;
    protected String eventTime;
    protected String venueName;
    protected int totalSeats;
    protected int availableSeats;
    protected double ticketPrice;
    protected String eventType;

    public Event() {}

    public Event(String eventName, String eventDate, String eventTime, String venueName,
                 int totalSeats, int availableSeats, double ticketPrice, String eventType) {
        this.eventName = eventName;
        this.eventDate = eventDate;
        this.eventTime = eventTime;
        this.venueName = venueName;
        this.totalSeats = totalSeats;
        this.availableSeats = availableSeats;
        this.ticketPrice = ticketPrice;
        this.eventType = eventType;
    }

    public abstract void displayEventDetails();

    public double calculateTotalRevenue() {
        return (totalSeats - availableSeats) * ticketPrice;
    }

    public int getBookedNoOfTickets() {
        return totalSeats - availableSeats;
    }

    public void bookTickets(int numTickets) {
        if (numTickets <= availableSeats) {
            availableSeats -= numTickets;
        } else {
            System.out.println("Not enough seats available.");
        }
    }

    public void cancelBooking(int numTickets) {
        availableSeats += numTickets;
    }

    public int getAvailableSeats() {
        return availableSeats;
    }
```

```java
    public double getTicketPrice() {
        return ticketPrice;
    }

    public String getEventName() {
        return eventName;
    }

    public String getVenueName() {
        return venueName;
    }

    public String getEventType() {
        return eventType;
    }
}
```

2. Concrete Event Classes

i. Movie

```java
package com.hexaware.entity;

public class Movie extends Event {
    private String genre;
    private String actorName;
    private String actressName;

    public Movie(String eventName, String eventDate, String eventTime, String venueName,
                 int totalSeats, int availableSeats, double ticketPrice, String eventType,
                 String genre, String actorName, String actressName) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, ticketPrice, eventType);
        this.genre = genre;
        this.actorName = actorName;
        this.actressName = actressName;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("🎬 Movie Event: " + eventName + " | Genre: " + genre +
            " | Actor: " + actorName + ", Actress: " + actressName +
            " | Date: " + eventDate + " | Time: " + eventTime +
            " | Venue: " + venueName + " | Price: ₹" + ticketPrice +
            " | Available Seats: " + availableSeats + "/" + totalSeats);
    }
}
```

ii. Concert

```java
package com.hexaware.entity;

public class Concert extends Event {
    private String artist;
    private String type;

    public Concert(String eventName, String eventDate, String eventTime, String venueName,
                   int totalSeats, int availableSeats, double ticketPrice, String eventType,
                   String artist, String type) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, ticketPrice, eventType);
        this.artist = artist;
        this.type = type;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("🎵 Concert Event: " + eventName + " | Artist: " + artist + ", Type: " + type +
            " | Date: " + eventDate + " | Time: " + eventTime +
            " | Venue: " + venueName + " | Price: ₹" + ticketPrice +
            " | Available Seats: " + availableSeats + "/" + totalSeats);
    }
}
```

iii. Sport

```java
package com.hexaware.entity;

public class Sports extends Event {
    private String sportName;
    private String teamsName;

    public Sports(String eventName, String eventDate, String eventTime, String venueName,
                  int totalSeats, int availableSeats, double ticketPrice, String eventType,
                  String sportName, String teamsName) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, ticketPrice, eventType);
        this.sportName = sportName;
        this.teamsName = teamsName;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("🏟 Sports Event: " + eventName + " | Sport: " + sportName +
            ", Teams: " + teamsName +
            " | Date: " + eventDate + " | Time: " + eventTime +
            " | Venue: " + venueName + " | Price: ₹" + ticketPrice +
            " | Available Seats: " + availableSeats + "/" + totalSeats);
    }
}
```

3. BookingSystem Abstraction:

```java
package com.hexaware.dao;

import com.hexaware.entity.Event;

public abstract class BookingSystem {
    public abstract Event createEvent();
    public abstract void displayEventDetails(Event event);
    public abstract void bookTickets(Event event, int numTickets);
    public abstract void cancelTickets(Event event, int numTickets);
    public abstract int getAvailableSeats(Event event);
}
```

4. Concrete TicketBookingSystem Class:

```java
package com.hexaware.main;

import com.hexaware.dao.BookingSystem;
import com.hexaware.entity.*;

import java.util.Scanner;

public class TicketBookingSystem extends BookingSystem {

    Scanner sc = new Scanner(System.in);

    @Override
    public Event createEvent() {
        System.out.print("Enter event type (Movie/Concert/Sports): ");
        String type = sc.nextLine();

        System.out.print("Enter event name: ");
        String name = sc.nextLine();

        System.out.print("Enter date (YYYY-MM-DD): ");
        String date = sc.nextLine();

        System.out.print("Enter time (HH:MM): ");
        String time = sc.nextLine();

        System.out.print("Enter venue: ");
        String venue = sc.nextLine();

        System.out.print("Enter total seats: ");
        int totalSeats = sc.nextInt();

        System.out.print("Enter available seats: ");
        int availableSeats = sc.nextInt();

        System.out.print("Enter ticket price: ");
        double price = sc.nextDouble();
        sc.nextLine(); // consume newline

        if (type.equalsIgnoreCase("movie")) {
            System.out.print("Enter genre: ");
            String genre = sc.nextLine();
            System.out.print("Enter actor name: ");
            String actor = sc.nextLine();
            System.out.print("Enter actress name: ");
            String actress = sc.nextLine();
            return new Movie(name, date, time, venue, totalSeats, availableSeats, price, "Movie", genre, actor, actress);
        } else if (type.equalsIgnoreCase("concert")) {
            System.out.print("Enter artist name: ");
            String artist = sc.nextLine();
            System.out.print("Enter concert type: ");
            String concertType = sc.nextLine();
            return new Concert(name, date, time, venue, totalSeats, availableSeats, price, "Concert", artist, concertType);
```

```
        } else if (type.equalsIgnoreCase("sports")) {
            System.out.print("Enter sport name: ");
            String sport = sc.nextLine();
            System.out.print("Enter teams name: ");
            String teams = sc.nextLine();
            return new Sports(name, date, time, venue, totalSeats, availableSeats, price, "Sports", sport, teams);
        }

        return null;
    }

    @Override
    public void displayEventDetails(Event event) {
        event.displayEventDetails(); // polymorphism
    }

    @Override
    public void bookTickets(Event event, int numTickets) {
        event.bookTickets(numTickets);
    }

    @Override
    public void cancelTickets(Event event, int numTickets) {
        event.cancelBooking(numTickets);
    }

    @Override
    public int getAvailableSeats(Event event) {
        return event.getAvailableSeats();
    }

    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem();
        Scanner sc = new Scanner(System.in);
        Event event = null;

        while (true) {
            System.out.println("\n--- Menu ---");
            System.out.println("1. Create Event");
            System.out.println("2. Display Event");
            System.out.println("3. Book Tickets");
            System.out.println("4. Cancel Tickets");
            System.out.println("5. Get Available Seats");
            System.out.println("6. Exit");
            System.out.print("Choose: ");
            int ch = sc.nextInt();
            sc.nextLine();
```

```
            switch (ch) {
                case 1:
                    event = system.createEvent();
                    break;
                case 2:
                    if (event != null) system.displayEventDetails(event);
                    else System.out.println("Create an event first!");
                    break;
                case 3:
                    if (event != null) {
                        System.out.print("Tickets to book: ");
                        int t = sc.nextInt();
                        sc.nextLine();
                        system.bookTickets(event, t);
                    }
                    break;
                case 4:
                    if (event != null) {
                        System.out.print("Tickets to cancel: ");
                        int c = sc.nextInt();
                        sc.nextLine();
                        system.cancelTickets(event, c);
                    }
                    break;
                case 5:
                    if (event != null)
                        System.out.println("Available: " + system.getAvailableSeats(event));
                    break;
                case 6:
                    System.out.println("Exiting.");
                    return;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
}
```

Task 7: Has A Relation / Association

Venue Class

```java
package com.hexaware.entity;

public class Venue {
    private String venueName;
    private String address;

    public Venue() {}

    public Venue(String venueName, String address) {
        this.venueName = venueName;
        this.address = address;
    }

    public String getVenueName() { return venueName; }
    public void setVenueName(String venueName) { this.venueName = venueName; }

    public String getAddress() { return address; }
    public void setAddress(String address) { this.address = address; }

    public void displayVenueDetails() {
        System.out.println("Venue: " + venueName + " | Address: " + address);
    }
}
```

Event Class:

```java
package com.hexaware.entity;

public abstract class Event {
    protected String eventName;
    protected String eventDate;
    protected String eventTime;
    protected Venue venue;
    protected int totalSeats;
    protected int availableSeats;
    protected double ticketPrice;
    protected String eventType;

    public Event() {}

    public Event(String eventName, String eventDate, String eventTime, Venue venue,
                 int totalSeats, int availableSeats, double ticketPrice, String eventType) {
        this.eventName = eventName;
        this.eventDate = eventDate;
        this.eventTime = eventTime;
        this.venue = venue;
        this.totalSeats = totalSeats;
        this.availableSeats = availableSeats;
        this.ticketPrice = ticketPrice;
        this.eventType = eventType;
    }

    public abstract void displayEventDetails();

    public double calculateTotalRevenue() {
        return (totalSeats - availableSeats) * ticketPrice;
    }

    public int getBookedNoOfTickets() {
        return totalSeats - availableSeats;
    }

    public void bookTickets(int numTickets) {
        if (numTickets <= availableSeats) {
            availableSeats -= numTickets;
        } else {
            System.out.println("Not enough seats available.");
        }
    }

    public void cancelBooking(int numTickets) {
        availableSeats += numTickets;
    }

    public int getAvailableSeats() {
        return availableSeats;
    }

    public double getTicketPrice() {
        return ticketPrice;
    }
```

```java
    public Venue getVenue() {
        return venue;
    }

    public String getEventName() {
        return eventName;
    }

    public String getEventType() {
        return eventType;
    }
}
```

Event sub classes:

```java
package com.hexaware.entity;

public class Sports extends Event {
    private String sportName;
    private String teamsName;

    public Sports(String eventName, String eventDate, String eventTime, Venue venue,
                  int totalSeats, int availableSeats, double ticketPrice, String eventType,
                  String sportName, String teamsName) {
        super(eventName, eventDate, eventTime, venue, totalSeats, availableSeats, ticketPrice, eventType);
        this.sportName = sportName;
        this.teamsName = teamsName;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("🏟 Sports Event: " + eventName +
            " | Sport: " + sportName +
            " | Teams: " + teamsName +
            " | Date: " + eventDate +
            " | Time: " + eventTime +
            " | Venue: " + venue.getVenueName() +
            " | Address: " + venue.getAddress() +
            " | Price: ₹" + ticketPrice +
            " | Available Seats: " + availableSeats + "/" + totalSeats);
    }
}
```

```java
package com.hexaware.entity;

public class Movie extends Event {
    private String genre;
    private String actorName;
    private String actressName;

    public Movie(String eventName, String eventDate, String eventTime, Venue venue,
                 int totalSeats, int availableSeats, double ticketPrice, String eventType,
                 String genre, String actorName, String actressName) {
        super(eventName, eventDate, eventTime, venue, totalSeats, availableSeats, ticketPrice, eventType);
        this.genre = genre;
        this.actorName = actorName;
        this.actressName = actressName;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("🎬 Movie Event: " + eventName +
            " | Genre: " + genre +
            " | Actor: " + actorName +
            " | Actress: " + actressName +
            " | Date: " + eventDate +
            " | Time: " + eventTime +
            " | Venue: " + venue.getVenueName() +
            " | Address: " + venue.getAddress() +
            " | Price: ₹" + ticketPrice +
            " | Available Seats: " + availableSeats + "/" + totalSeats);
    }
}
```

```java
package com.hexaware.entity;

public class Venue {
    private String venueName;
    private String address;

    public Venue() {}

    public Venue(String venueName, String address) {
        this.venueName = venueName;
        this.address = address;
    }

    public String getVenueName() { return venueName; }
    public void setVenueName(String venueName) { this.venueName = venueName; }

    public String getAddress() { return address; }
    public void setAddress(String address) { this.address = address; }

    public void displayVenueDetails() {
        System.out.println("Venue: " + venueName + " | Address: " + address);
    }
}
```

Customer Class

```java
package com.hexaware.entity;

public class Customer {
    private String customerName;
    private String email;
    private String phoneNumber;

    public Customer() {}

    public Customer(String customerName, String email, String phoneNumber) {
        this.customerName = customerName;
        this.email = email;
        this.phoneNumber = phoneNumber;
    }

    public String getCustomerName() { return customerName; }
    public void setCustomerName(String customerName) { this.customerName = customerName; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public String getPhoneNumber() { return phoneNumber; }
    public void setPhoneNumber(String phoneNumber) { this.phoneNumber = phoneNumber; }

    public void displayCustomerDetails() {
        System.out.println("Customer: " + customerName + " | Email: " + email +
                " | Phone: " + phoneNumber);
    }
}
```

Create a class Booking with the following attributes:

```java
package com.hexaware.entity;

import java.time.LocalDateTime;

public class Booking {
    private static int counter = 1000;
    private int bookingId;
    private Customer[] customers;
    private Event event;
    private int numTickets;
    private double totalCost;
    private LocalDateTime bookingDate;

    public Booking() {}

    public Booking(Customer[] customers, Event event, int numTickets) {
        this.bookingId = counter++;
        this.customers = customers;
        this.event = event;
        this.numTickets = numTickets;
        this.totalCost = event.getTicketPrice() * numTickets;
        this.bookingDate = LocalDateTime.now();
        event.bookTickets(numTickets);
    }

    public void displayBookingDetails() {
        System.out.println("\n Booking ID: " + bookingId);
        event.displayEventDetails();
        System.out.println(" Tickets: " + numTickets + " | Total Cost: ₹" + totalCost);
        System.out.println(" Customers:");
        for (Customer c : customers) {
            c.displayCustomerDetails();
        }
        System.out.println(" Booking Date: " + bookingDate);
    }

    public int getBookingId() {
        return bookingId;
    }
}
```

BookingSystem Class to represent the Ticket booking system. Perform the following operation in.

```java
package com.hexaware.main;

import com.hexaware.entity.*;

public class BookingSystem {
    private Event[] events = new Event[10];
    private int eventCount = 0;
    private Booking[] bookings = new Booking[10];
    private int bookingCount = 0;
    Scanner sc = new Scanner(System.in);

    public Venue createVenue() {
        System.out.print("Enter venue name: ");
        String name = sc.nextLine();
        System.out.print("Enter venue address: ");
        String address = sc.nextLine();
        return new Venue(name, address);
    }

    public Event createEvent() {
        System.out.print("Enter event type (Movie/Concert/Sports): ");
        String type = sc.nextLine();

        System.out.print("Enter event name: ");
        String name = sc.nextLine();
        System.out.print("Enter event date: ");
        String date = sc.nextLine();
        System.out.print("Enter event time: ");
        String time = sc.nextLine();
        Venue venue = createVenue();

        System.out.print("Enter total seats: ");
        int totalSeats = sc.nextInt();
        System.out.print("Enter available seats: ");
        int availableSeats = sc.nextInt();
        System.out.print("Enter ticket price: ");
        double price = sc.nextDouble();
        sc.nextLine();
```

```java
        Event event = null;
        if (type.equalsIgnoreCase("movie")) {
            System.out.print("Enter genre: ");
            String genre = sc.nextLine();
            System.out.print("Enter actor name: ");
            String actor = sc.nextLine();
            System.out.print("Enter actress name: ");
            String actress = sc.nextLine();
            event = new Movie(name, date, time, venue, totalSeats, availableSeats, price, "Movie", genre, actor, actress);
        } else if (type.equalsIgnoreCase("concert")) {
            System.out.print("Enter artist: ");
            String artist = sc.nextLine();
            System.out.print("Enter type (Rock/Classical): ");
            String concertType = sc.nextLine();
            event = new Concert(name, date, time, venue, totalSeats, availableSeats, price, "Concert", artist, concertType);
        } else if (type.equalsIgnoreCase("sports")) {
            System.out.print("Enter sport name: ");
            String sport = sc.nextLine();
            System.out.print("Enter teams: ");
            String teams = sc.nextLine();
            event = new Sports(name, date, time, venue, totalSeats, availableSeats, price, "Sports", sport, teams);
        }

        if (event != null) {
            events[eventCount++] = event;
        }
        return event;
    }
```

```java
public void bookTickets(String eventName, int numTickets) {
    Event selectedEvent = null;
    for (Event e : events) {
        if (e != null && e.getEventName().equalsIgnoreCase(eventName)) {
            selectedEvent = e;
            break;
        }
    }

    if (selectedEvent == null) {
        System.out.println("Event not found.");
        return;
    }

    Customer[] customers = new Customer[numTickets];
    for (int i = 0; i < numTickets; i++) {
        System.out.println("Enter details for customer " + (i + 1));
        System.out.print("Name: ");
        String name = sc.nextLine();
        System.out.print("Email: ");
        String email = sc.nextLine();
        System.out.print("Phone: ");
        String phone = sc.nextLine();
        customers[i] = new Customer(name, email, phone);
    }

    Booking booking = new Booking(customers, selectedEvent, numTickets);
    bookings[bookingCount++] = booking;
    booking.displayBookingDetails();
}

public void showAllEvents() {
    System.out.println("\n All Events:");
    for (Event e : events) {
        if (e != null) {
            e.displayEventDetails();
            System.out.println("-------------------------");
        }
    }
}
```

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

IEventServiceProvider:

```java
package com.hexaware.service;

import com.hexaware.entity.*;
import java.util.List;

public interface IEventServiceProvider {
    Event createEvent(String type, Venue venue);
    List<Event> getEventDetails();
    int getAvailableNoOfTickets(String eventName);
}
```

IBookingSystemServiceProvider

```java
package com.hexaware.service;

import com.hexaware.entity.*;
import com.hexaware.exception.*;
import java.util.List;

public interface IBookingSystemServiceProvider {
    Booking bookTickets(String eventName, int numTickets, List<Customer> customers) throws EventNotFoundException;
    void cancelBooking(int bookingId) throws InvalidBookingIDException;
    Booking getBookingDetails(int bookingId) throws InvalidBookingIDException;
}
```

EventServiceProviderImpl:

```java
package com.hexaware.dao;

import com.hexaware.entity.*;
import com.hexaware.service.IEventServiceProvider;
import java.util.ArrayList;
import java.util.List;

public class EventServiceProviderImpl implements IEventServiceProvider {

    protected List<Event> events = new ArrayList<>();

    @Override
    public Event createEvent(String type, Venue venue) {
        return null; // Created in main class
    }

    @Override
    public List<Event> getEventDetails() {
        return events;
    }

    @Override
    public int getAvailableNoOfTickets(String eventName) {
        for (Event e : events) {
            if (e != null && e.getEventName().equalsIgnoreCase(eventName)) {
                return e.getAvailableSeats();
            }
        }
        return -1;
    }

    public void addEvent(Event event) {
        events.add(event);
    }

    public Event findEventByName(String name) {
        for (Event e : events) {
            if (e != null && e.getEventName().equalsIgnoreCase(name)) {
                return e;
            }
        }
        return null;
    }
```

BookingSystemServiceProviderImpl:

```java
package com.hexaware.dao;

import com.hexaware.entity.*;
import com.hexaware.exception.*;
import com.hexaware.service.IBookingSystemServiceProvider;
import java.util.ArrayList;
import java.util.List;

public class BookingSystemServiceProviderImpl extends EventServiceProviderImpl implements IBookingSystemServiceProvider {

    private static int bookingIdCounter = 1001;
    private List<Booking> bookings = new ArrayList<>();

    @Override
    public Booking bookTickets(String eventName, int numTickets, List<Customer> customers) throws EventNotFoundException {
        Event event = findEventByName(eventName);
        if (event == null) {
            throw new EventNotFoundException("Event not found: " + eventName);
        }
        if (event.getAvailableSeats() < numTickets) {
            System.out.println("Not enough seats.");
            return null;
        }

        Booking booking = new Booking(bookingIdCounter++, customers, event, numTickets);
        bookings.add(booking);
        return booking;
    }

    @Override
    public void cancelBooking(int bookingId) throws InvalidBookingIDException {
        for (Booking b : bookings) {
            if (b != null && b.getBookingId() == bookingId) {
                b.getEventDetails().cancelBooking(b.getBookedNoOfTickets());
                System.out.println("Booking cancelled.");
                return;
            }
        }
        throw new InvalidBookingIDException("Booking ID not found: " + bookingId);
    }
```

```java
    @Override
    public Booking getBookingDetails(int bookingId) throws InvalidBookingIDException {
        for (Booking b : bookings) {
            if (b != null && b.getBookingId() == bookingId) {
                return b;
            }
        }
        throw new InvalidBookingIDException("Invalid booking ID: " + bookingId);
    }
}
```

EventServiceProviderImpl:

```java
package com.hexaware.dao;

import com.hexaware.bean.Concert;
import com.hexaware.bean.Event;
import com.hexaware.bean.Movie;
import com.hexaware.bean.Sports;
import com.hexaware.bean.Venue;
import com.hexaware.exception.DatabaseException;
import com.hexaware.service.IEventServiceProvider;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.HashSet;
import java.util.Set;

public class EventServiceProviderImpl implements IEventServiceProvider {
    protected Set<Event> events = new HashSet<>();

    @Override
    public Event createEvent(String eventName, String date, String time, int totalSeats,
                             double ticketPrice, String eventType, Venue venue) throws DatabaseException {
        LocalDate eventDate = LocalDate.parse(date);
        LocalTime eventTime = LocalTime.parse(time);
        Event event = null;
        if (eventType.equalsIgnoreCase("Movie")) {
            event = new Movie(eventName, eventDate, eventTime, venue, totalSeats, totalSeats, ticketPrice, eventType, "", "", "");
        } else if (eventType.equalsIgnoreCase("Concert")) {
            event = new Concert(eventName, eventDate, eventTime, venue, totalSeats, totalSeats, ticketPrice, eventType, "", "");
        } else {
            event = new Sports(eventName, eventDate, eventTime, venue, totalSeats, totalSeats, ticketPrice, eventType, "", "");
        }
        events.add(event);
        return event;
    }

    @Override
    public Set<Event> getEventDetails() throws DatabaseException {
        return events;
    }

    @Override
    public int getAvailableNoOfTickets(String eventName) throws DatabaseException {
        for (Event e : events) {
            if (e.getEventName().equalsIgnoreCase(eventName)) {
                return e.getAvailableSeats();
            }
        }
        return -1;
    }

    public Event findEventByName(String name) {
        for (Event e : events) {
            if (e.getEventName().equalsIgnoreCase(name)) {
                return e;
            }
        }
        return null;
    }
```

Task 9: Exception Handling

EventNotFoundException

```java
package com.hexaware.exception;

public class EventNotFoundException extends Exception {
    public EventNotFoundException(String message) {
        super(message);
    }
}
```

InvalidBookingIDException

```java
package com.hexaware.exception;

public class InvalidBookingIDException extends Exception {
    public InvalidBookingIDException(String message) {
        super(message);
    }
}
```

DatabaseException

```java
package com.hexaware.exception;

public class DatabaseException extends Exception {
    public DatabaseException(String message) {
        super(message);
    }

    public DatabaseException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

Task 10: Collection

1. From the previous task change the Booking class attribute customers to List of customers and

BookingSystem class attribute events to List of events and perform the same operation.

2. From the previous task change all list type of attribute to type Set in Booking and BookingSystem

class and perform the same operation.

• Avoid adding duplicate Account object to the set.

• Create Comparator<Event> object to sort the event based on event name and location in

alphabetical order.

3. From the previous task change all list type of attribute to type Map object in Booking and

BookingSystem class and perform the same operation.

```
public class Booking {
    private int bookingId;
    private List<Customer> customers;
    private Event event;
    private int numTickets;
    private double totalCost;
    private LocalDateTime bookingDate;
```

```
public class BookingSystem {
    private Event[] events = new Event[10];
    private int eventCount = 0;
    private List<Booking> bookings = new ArrayList<>();
    Scanner sc = new Scanner(System.in);
```

```
public class EventServiceProviderImpl implements IEventServiceProvider

    protected List<Event> events = new ArrayList<>();
```

```java
public void addEvent(Event event) {
    events.add(event);
}
```

IBookingSystemRepository

```java
package com.hexaware.service;

import com.hexaware.bean.Booking;
import com.hexaware.bean.Customer;
import com.hexaware.bean.Event;
import com.hexaware.bean.Venue;
import com.hexaware.exception.DatabaseException;
import java.sql.SQLException;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.List;
import java.util.Set;


public interface IBookingSystemRepository {
    Event createEvent(String eventName, LocalDate date, LocalTime time, int totalSeats, double ticketPrice,
                      String eventType, Venue venue, String actor, String actress) throws SQLException, DatabaseException;
    Set<Event> getEventDetails() throws SQLException, DatabaseException;
    int getAvailableNoOfTickets(String eventName) throws SQLException, DatabaseException;
    Booking bookTickets(String eventName, int numTickets, List<Customer> customers) throws SQLException, DatabaseException;
    void cancelBooking(int bookingId) throws SQLException, DatabaseException;
    Booking getBookingDetails(int bookingId) throws SQLException, DatabaseException;
}
```

BookingSystemRepositoryImpl

```java
package com.hexaware.dao;

import com.hexaware.bean.*;
import com.hexaware.exception.DatabaseException;
import com.hexaware.service.IBookingSystemRepository;
import com.hexaware.util.DBUtil;
import java.sql.*;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;


public class BookingSystemRepositoryImpl implements IBookingSystemRepository {

    private int insertOrGetVenueId(Venue venue, Connection conn) throws SQLException {
        String selectSql = "SELECT venue_id FROM Venue WHERE venue_name = ?";
        PreparedStatement checkStmt = conn.prepareStatement(selectSql);
        checkStmt.setString(1, venue.getVenueName());
        ResultSet rs = checkStmt.executeQuery();

        if (rs.next()) {
            return rs.getInt("venue_id");
        }

        String insertSql = "INSERT INTO Venue (venue_name, address) VALUES (?, ?)";
        PreparedStatement insertStmt = conn.prepareStatement(insertSql, Statement.RETURN_GENERATED_KEYS);
        insertStmt.setString(1, venue.getVenueName());
        insertStmt.setString(2, venue.getAddress());
        insertStmt.executeUpdate();

        ResultSet keys = insertStmt.getGeneratedKeys();
        if (keys.next()) {
            return keys.getInt(1);
        }

        throw new SQLException("Failed to insert or retrieve venue ID.");
    }

    @Override
    public Event createEvent(String eventName, LocalDate date, LocalTime time, int totalSeats,
                             double ticketPrice, String eventType, Venue venue, String actor, String actress)
                             throws SQLException, DatabaseException {
        String sql = "INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type, actor, actress) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        try (Connection conn = DBUtil.getConnection()) {
            int venueId = insertOrGetVenueId(venue, conn);
            PreparedStatement pstmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
            pstmt.setString(1, eventName);
            pstmt.setDate(2, java.sql.Date.valueOf(date));
            pstmt.setTime(3, java.sql.Time.valueOf(time));
            pstmt.setInt(4, venueId);
            pstmt.setInt(5, totalSeats);
            pstmt.setInt(6, totalSeats);
```

```java
            pstmt.setString(8, eventType);
            if ("Movie".equalsIgnoreCase(eventType)) {
                pstmt.setString(9, actor != null ? actor : "");
                pstmt.setString(10, actress != null ? actress : "");
            } else {
                pstmt.setString(9, "");
                pstmt.setString(10, "");
            }
            pstmt.executeUpdate();

            ResultSet keys = pstmt.getGeneratedKeys();
            int eventId = keys.next() ? keys.getInt(1) : -1;

            Event event;
            if (eventType.equalsIgnoreCase("Movie")) {
                event = new Movie(eventName, date, time, venue, totalSeats, totalSeats, ticketPrice, eventType, "", actor, actress);
            } else if (eventType.equalsIgnoreCase("Concert")) {
                event = new Concert(eventName, date, time, venue, totalSeats, totalSeats, ticketPrice, eventType, "", "");
            } else {
                event = new Sports(eventName, date, time, venue, totalSeats, totalSeats, ticketPrice, eventType, "", "");
            }
            event.setEventId(eventId);
            return event;
        } catch (SQLException e) {
            throw new DatabaseException("Error inserting event", e);
        }
    }

    @Override
    public Set<Event> getEventDetails() throws SQLException, DatabaseException {
        Set<Event> events = new HashSet<>();
        String sql = "SELECT e.*, v.venue_name, v.address FROM Event e JOIN Venue v ON e.venue_id = v.venue_id";
        try (Connection conn = DBUtil.getConnection()) {
            PreparedStatement pstmt = conn.prepareStatement(sql);
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                Venue venue = new Venue(rs.getInt("venue_id"), rs.getString("venue_name"), rs.getString("address"));
                Event event;
                String eventType = rs.getString("event_type");
                if (eventType.equalsIgnoreCase("Movie")) {
                    event = new Movie(
                        rs.getString("event_name"),
                        rs.getDate("event_date").toLocalDate(),
                        rs.getTime("event_time").toLocalTime(),
                        venue,
                        rs.getInt("total_seats"),
                        rs.getInt("available_seats"),
                        rs.getDouble("ticket_price"),
                        eventType,
                        rs.getString("actor"),
                        rs.getString("actress"),
                        ""
                    );
                } else if (eventType.equalsIgnoreCase("Concert")) {
                    event = new Concert(
                        rs.getString("event_name"),
                        rs.getDate("event_date").toLocalDate(),
```

```
                    rs.getDate("event_date").toLocalDate(),
                    rs.getTime("event_time").toLocalTime(),
                    venue,
                    rs.getInt("total_seats"),
                    rs.getInt("available_seats"),
                    rs.getDouble("ticket_price"),
                    eventType,
                    "",
                    ""
                );
            } else {
                event = new Sports(
                    rs.getString("event_name"),
                    rs.getDate("event_date").toLocalDate(),
                    rs.getTime("event_time").toLocalTime(),
                    venue,
                    rs.getInt("total_seats"),
                    rs.getInt("available_seats"),
                    rs.getDouble("ticket_price"),
                    eventType,
                    "",
                    ""
                );
            }
            event.setEventId(rs.getInt("event_id"));
            events.add(event);
        }
    } catch (SQLException e) {
        throw new DatabaseException("Error fetching events", e);
    }
    return events;
}

@Override
public int getAvailableNoOfTickets(String eventName) throws SQLException, DatabaseException {
    String sql = "SELECT available_seats FROM Event WHERE event_name = ?";
    try (Connection conn = DBUtil.getConnection()) {
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, eventName);
        ResultSet rs = pstmt.executeQuery();
        if (rs.next()) {
            return rs.getInt("available_seats");
        }
    } catch (SQLException e) {
        throw new DatabaseException("Error retrieving available seats", e);
    }
    return -1;
}

@Override
public Booking bookTickets(String eventName, int numTickets, List<Customer> customers) throws SQLException, DatabaseException {
    Connection conn = null;
```

```
    Connection conn = null;
    try {
        conn = DBUtil.getConnection();
        conn.setAutoCommit(false);

        Event event = null;
        for (Event e : getEventDetails()) {
            if (e.getEventName().equalsIgnoreCase(eventName)) {
                event = e;
                break;
            }
        }
        if (event == null) {
            throw new DatabaseException("Event not found: " + eventName);
        }
        if (event.getAvailableSeats() < numTickets) {
            throw new DatabaseException("Not enough available seats for event: " + eventName);
        }

        String bookingSQL = "INSERT INTO Booking (event_id, num_tickets, total_cost, booking_date) VALUES (?, ?, ?, ?)";
        PreparedStatement bookingStmt = conn.prepareStatement(bookingSQL, Statement.RETURN_GENERATED_KEYS);
        bookingStmt.setInt(1, event.getEventId());
        bookingStmt.setInt(2, numTickets);
        bookingStmt.setDouble(3, event.getTicketPrice() * numTickets);
        bookingStmt.setTimestamp(4, Timestamp.valueOf(LocalDateTime.now()));
        bookingStmt.executeUpdate();

        ResultSet keys = bookingStmt.getGeneratedKeys();
        if (!keys.next()) throw new DatabaseException("Failed to retrieve booking ID.");
        int bookingId = keys.getInt(1);

        String customerSQL = "INSERT INTO Customer (customer_name, email, phone_number, booking_id) VALUES (?, ?, ?, ?)";
        PreparedStatement customerStmt = conn.prepareStatement(customerSQL, Statement.RETURN_GENERATED_KEYS);
        for (Customer c : customers) {
            customerStmt.setString(1, c.getCustomerName());
            customerStmt.setString(2, c.getEmail());
            customerStmt.setString(3, c.getPhoneNumber());
            customerStmt.setInt(4, bookingId);
            customerStmt.executeUpdate();
            ResultSet customerKeys = customerStmt.getGeneratedKeys();
            if (customerKeys.next()) {
                c.setCustomerId(customerKeys.getInt(1));
            }
        }

        String updateSeatsSQL = "UPDATE Event SET available_seats = available_seats - ? WHERE event_id = ?";
        PreparedStatement updateStmt = conn.prepareStatement(updateSeatsSQL);
        updateStmt.setInt(1, numTickets);
        updateStmt.setInt(2, event.getEventId());
        updateStmt.executeUpdate();

        conn.commit();
```

```
            return new Booking(bookingId, new HashSet<>(customers), event, numTickets, event.getTicketPrice() * numTickets, LocalDateTime.now());
        } catch (SQLException e) {
            if (conn != null) conn.rollback();
            throw new DatabaseException("Database error during booking", e);
        } finally {
            if (conn != null) {
                conn.setAutoCommit(true);
                conn.close();
            }
        }
    }

    @Override
    public void cancelBooking(int bookingId) throws SQLException, DatabaseException {
        Connection conn = null;
        try {
            conn = DBUtil.getConnection();
            conn.setAutoCommit(false);

            String fetchSQL = "SELECT event_id, num_tickets FROM Booking WHERE booking_id = ?";
            PreparedStatement fetchStmt = conn.prepareStatement(fetchSQL);
            fetchStmt.setInt(1, bookingId);
            ResultSet rs = fetchStmt.executeQuery();

            if (!rs.next()) {
                throw new DatabaseException("Booking ID not found: " + bookingId);
            }

            int eventId = rs.getInt("event_id");
            int tickets = rs.getInt("num_tickets");

            PreparedStatement delCustomers = conn.prepareStatement("DELETE FROM Customer WHERE booking_id = ?");
            delCustomers.setInt(1, bookingId);
            delCustomers.executeUpdate();

            PreparedStatement delBooking = conn.prepareStatement("DELETE FROM Booking WHERE booking_id = ?");
            delBooking.setInt(1, bookingId);
            delBooking.executeUpdate();

            PreparedStatement updateSeats = conn.prepareStatement("UPDATE Event SET available_seats = available_seats + ? WHERE event_id = ?");
            updateSeats.setInt(1, tickets);
            updateSeats.setInt(2, eventId);
            updateSeats.executeUpdate();

            conn.commit();
        } catch (SQLException e) {
            if (conn != null) conn.rollback();
            throw new DatabaseException("Database error during cancellation", e);
        } finally {
            if (conn != null) {
                conn.setAutoCommit(true);
                conn.close();
            }
        }
    }
```

```
    @Override
    public Booking getBookingDetails(int bookingId) throws SQLException, DatabaseException {
        Connection conn = DBUtil.getConnection();
        try {
            String bookingSQL = "SELECT b.*, e.*, v.venue_name, v.address FROM Booking b JOIN Event e ON b.event_id = e.event_id JOIN Venue v ON e.venue_id = v.venue_id WHERE b.booking_id = ?";
            PreparedStatement pstmt = conn.prepareStatement(bookingSQL);
            pstmt.setInt(1, bookingId);
            ResultSet rs = pstmt.executeQuery();

            if (!rs.next()) return null;

            Venue venue = new Venue(rs.getInt("venue_id"), rs.getString("venue_name"), rs.getString("address"));
            Event event;
            String eventType = rs.getString("event_type");
            if (eventType.equalsIgnoreCase("Movie")) {
                event = new Movie(
                    rs.getString("event_name"),
                    rs.getDate("event_date").toLocalDate(),
                    rs.getTime("event_time").toLocalTime(),
                    venue,
                    rs.getInt("total_seats"),
                    rs.getInt("available_seats"),
                    rs.getDouble("ticket_price"),
                    eventType,
                    rs.getString("actor"),
                    rs.getString("actress"),
                    ""
                );
            } else if (eventType.equalsIgnoreCase("Concert")) {
                event = new Concert(
                    rs.getString("event_name"),
                    rs.getDate("event_date").toLocalDate(),
                    rs.getTime("event_time").toLocalTime(),
                    venue,
                    rs.getInt("total_seats"),
                    rs.getInt("available_seats"),
                    rs.getDouble("ticket_price"),
                    eventType,
                    "",
                    ""
                );
            } else {
                event = new Sports(
                    rs.getString("event_name"),
                    rs.getDate("event_date").toLocalDate(),
                    rs.getTime("event_time").toLocalTime(),
                    venue,
                    rs.getInt("total_seats"),
                    rs.getInt("available_seats"),
                    rs.getDouble("ticket_price"),
                    eventType,
                    "",
                    ""
                );
            }
            event.setEventId(rs.getInt("event_id"));
```

```
                );
            }
            event.setEventId(rs.getInt("event_id"));

            int numTickets = rs.getInt("num_tickets");
            double cost = rs.getDouble("total_cost");
            LocalDateTime bookingDate = rs.getTimestamp("booking_date").toLocalDateTime();

            String customerSQL = "SELECT * FROM Customer WHERE booking_id = ?";
            PreparedStatement custStmt = conn.prepareStatement(customerSQL);
            custStmt.setInt(1, bookingId);
            ResultSet crs = custStmt.executeQuery();

            Set<Customer> customers = new HashSet<>();
            while (crs.next()) {
                Customer c = new Customer(
                    crs.getInt("customer_id"),
                    crs.getString("customer_name"),
                    crs.getString("email"),
                    crs.getString("phone_number")
                );
                customers.add(c);
            }

            return new Booking(bookingId, customers, event, numTickets, cost, bookingDate);
        } catch (SQLException e) {
            throw new DatabaseException("Error retrieving booking", e);
        } finally {
            conn.close();
        }
    }
}
```

DBPropertyUtil

```java
package com.hexaware.util;

import java.io.IOException;
import java.util.Properties;


public class DBPropertyUtil {

    public static String getConnectionString(String propertyFileName) {
        Properties props = new Properties();
        try (var input = DBPropertyUtil.class.getClassLoader().getResourceAsStream(propertyFileName)) {
            if (input == null) {
                throw new IOException("Property file '" + propertyFileName + "' not found in the classpath. Ensure it is placed in src/main/resources or src.");
            }
            props.load(input);
            String url = props.getProperty("db.url");
            String user = props.getProperty("db.user");
            String password = props.getProperty("db.password");
            if (url == null || user == null || password == null) {
                throw new IOException("Missing required properties in '" + propertyFileName + "'. Ensure db.url, db.user, and db.password are defined.");
            }
            return url + "?user=" + user + "&password=" + password;
        } catch (IOException e) {
            throw new RuntimeException("Error reading database properties: " + e.getMessage(), e);
        }
    }
}
```

DBUtil

```java
package com.hexaware.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


public class DBUtil {

    public static Connection getConnection() throws SQLException {
        String connectionString = DBPropertyUtil.getConnectionString("db.properties");
        return DriverManager.getConnection(connectionString);
    }
}
```

Final Working Menu:

```
--- MENU ---
1. Create Event
2. Show Events
3. Book Tickets
4. Cancel Booking
5. View Booking
6. Exit
Choose option:
```

Create Event:

```
Choose option: 1
Enter event type (Movie/Concert/Sports): Movie
Event name: Chaava
Date (YYYY-MM-DD): 2025-04-18
Time (HH:MM): 12:30
Venue name: PVR
Venue address: Mg Road
Total seats: 500
Ticket price: 500
Genre: Action
Actor: Vicky
Actress: Rashmika
☑ Event created successfully.
```

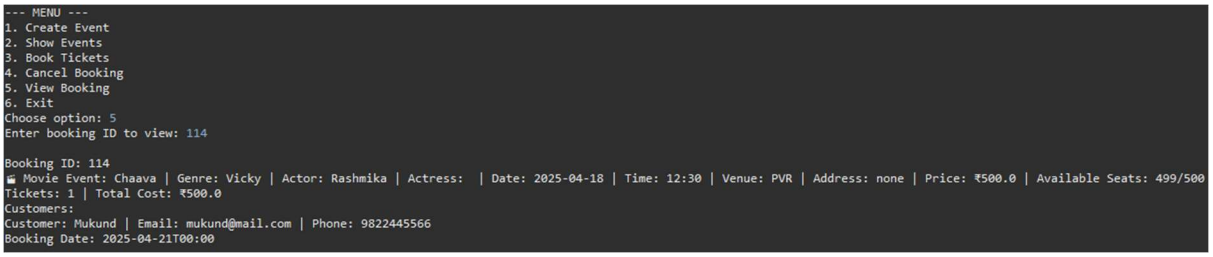
Show Events:

```
Choose option: 2
♫ Concert Event: Bollywood Beats | Artist:  | Type:  | Date: 2025-05-18 | Time: 19:00 | Venue: Gachibowli Stadium | Address: Hyderabad, Telangana | Price: ₹1600.0 | Available Seats: 450/1500
🎬 Movie Event: Chaava | Genre: Vicky | Actor: Rashmika | Actress:  | Date: 2025-04-18 | Time: 12:30 | Venue: PVR | Address: none | Price: ₹500.0 | Available Seats: 500/500
♫ Concert Event: Classical Night | Artist:  | Type:  | Date: 2025-05-10 | Time: 20:00 | Venue: Chowdiah Memorial Hall | Address: Bangalore, Karnataka | Price: ₹1500.0 | Available Seats: 400/800
🎬 Movie Event: Documentary Premiere | Genre: null | Actor: null | Actress:  | Date: 2025-06-01 | Time: 17:00 | Venue: Biswa Bangla Convention Centre | Address: Kolkata, West Bengal | Price: ₹850.0 | Av
🎬 Movie Event: Eco Documentary | Genre: null | Actor: null | Actress:  | Date: 2025-05-02 | Time: 16:00 | Venue: The Lalit Ashok | Address: Bangalore, Karnataka | Price: ₹500.0 | Available Seats: 0/100
🏏 Sports Event: FIFA World Cup Screening | Sport:  | Teams:  | Date: 2025-05-15 | Time: 21:00 | Venue: Bandra Kurla Complex | Address: Mumbai, Maharashtra | Price: ₹1800.0 | Available Seats: 0/300
🏏 Sports Event: IPL Final Match | Sport:  | Teams:  | Date: 2025-05-20 | Time: 19:30 | Venue: Jawaharlal Nehru Stadium | Address: New Delhi | Price: ₹2200.0 | Available Seats: 12000/50000
🎬 Movie Event: Indie Film Cup | Genre: null | Actor: null | Actress:  | Date: 2025-05-05 | Time: 18:00 | Venue: India Habitat Centre | Address: Lodhi Road, New Delhi | Price: ₹700.0 | Available Seats:
♫ Concert Event: Music Cup Live | Artist:  | Type:  | Date: 2025-06-05 | Time: 20:30 | Venue: Nehru Indoor Stadium | Address: Chennai, Tamil Nadu | Price: ₹2000.0 | Available Seats: 1000/5000
🎬 Movie Event: Pushpa 2 | Genre: Alluarjun | Actor: rashmika | Actress:  | Date: 2025-04-12 | Time: 12:30 | Venue: PVR | Address: none | Price: ₹100.0 | Available Seats: 399/400
🏏 Sports Event: Xtreme Wrestling | Sport:  | Teams:  | Date: 2025-04-28 | Time: 18:30 | Venue: Codissia Trade Fair Complex | Address: Coimbatore, Tamil Nadu | Price: ₹950.0 | Available Seats: 3000/1000
♫ Concert Event: Zodiac Rock Fest | Artist:  | Type:  | Date: 2025-04-30 | Time: 19:00 | Venue: Sardar Patel Stadium | Address: Ahmedabad, Gujarat | Price: ₹1300.0 | Available Seats: 12000/12000
```

Book Tickets

```
--- MENU ---
1. Create Event
2. Show Events
3. Book Tickets
4. Cancel Booking
5. View Booking
6. Exit
Choose option: 3
Event to book (or 'Exit' to stop): chaava
Tickets: 1
Ticket Category (Silver/Gold/Diamond): Diamond
Enter details for Customer 1
Name: Mukund
Email: mukund@mail.com
Phone: 9822445566
☑ Booking successful. ID: 114
```

View Booking



Cancel Booking