Ticket Booking Assignment

Task-2:

-- 2. Write a SQL query to list all Events.

```
select * from event;
```

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

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

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

```
select *
from event
where event_name LIKE '%cup%';
```

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

```
select * from event
where ticket_price between 500 and 2500;
```

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

```
select *
from event
where event_date BETWEEN '2024-04-11' AND '2024-05-01';
```

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

```
select * from event

where available_seats >0 aNd event_type like '%concert%';
```

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

```
select *

from customer

limit 3,2;
```

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

```
select
e.event_name,e.event_date,event_time,total_seats,available_seats,ticket_price,event_type

from event e,booking b

where e.id=b.event_id and num_tickets>4;
```

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

```
select *

from customer

where phone_number LIKE '%000'; # ends number with 000
```

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

```
select *

from event

where total_seats > 15000

order by total_seats ASC ;
```

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

```
select *

from event

where event_name NOT LIKE 'c%' AND event_name NOT LIKE 'x%';
```

Task-3:

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

```
select event_name ,avg(ticket_price)

from event

group by event_name;
```

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

```
select v.venue_name, avg(e.ticket_price)

from event e,venue v

where v.id=e.venue_id

group by v.venue_name;
```

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

```
select event_name,((total_seats-available_seats)*ticket_price) as Revenue
from event;
```

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

```
select event_name,MAX(total_seats-available_seats) as highest_ticket_sales
from event
group by event_name
order by highest_ticket_sales desc
limit 0,1;
```

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

```
select event_name,MAX(total_seats-available_seats) as total_sales
from event
group by event_name;
```

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

```
select event_name
from event
where total_seats=available_seats;
```

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

```
select customer_name,sum(num_tickets)
from customer c,booking b
where c.id=b.customer_id
group by customer_name
```

```
order by sum(num_tickets) desc

limit 0,1;
```

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

```
select venue_name,avg(ticket_price)

from venue v,event e

where v.id=e.venue_id

group by v.id;
```

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

```
select event_type,sum(total_seats-available_seats) as tickets_sold

from event

group by event_type;
```

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

```
select c.customer_name,count(id)

from customer c,booking b

where c.id=b.customer_id

group by c.customer_name

having count(id)>1;
```

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

```
select  c.customer_name,sum(b.total_cost)
from event e join booking b on e.id=b.event_id join customer c on c.id=b.customer_id
group by c.customer_name;
```

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

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

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

```
select c.customer_name, SUM(b.num_tickets) as Number_Of_tickets
from  event e  JOIN booking b  ON e.id = b.event_id JOIN  customer c ON c.id = b.customer_id
where b.booking_date between DATE_SUB('2024-04-30',INTERVAL 30 DAY) and  '2024-04-30'
group by c.customer_name;
```

Task-4:

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

```
select venue_id,AVG(ticket_price) as Avg_Price
from event
where venue_id IN (select id from venue)
group by venue_id;
```

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

```
select event_name
from event
where id IN ( select id
 from event
 where (total_seats - available_seats) > (total_seats/2));
```

-- 3. Calculate the Total Number of Tickets Sold for Each Event.

```
select event_name
from event
where ticket_price > (select avg(ticket_price) from event);
```

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

```
select * from customer
where not exists (select 1
from booking b
where b.customer_id = customer.id);
```

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

```
select * from event
where id NOT IN (select distinct event_id
from booking);
use ticketbooking_feb_hex_24;
```

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

```
select id,event_name from event where
ticket_price > (select
 avg(ticket_price) from event);
```

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

```
select * from customer
where id in(select customer_id from booking
where event_id in(select id from event
where venue_id in (select id from venue
 where venue_name='chennai')));
```

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

```
select event_type, sum(b.num_tickets)as total_tickets_booked from event
e,booking b where b.event_id=e.id
group by event_type;
```

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

```
select id,venue_name,(select avg(ticket_price)
 from event where venue.id=event.venue_id) as Avg_ticket_price from venue;
```