```
use ticketbooking_feb_hex_24;

-- Ticket Booking Assignment -- PART 4
--  5. Write a SQL query to Find Events with No Ticket Sales.

select *
from event
where id NOT IN (select e.id
from event e JOIN booking b ON e.id = b.event_id);

-- using manual mapping
select *
from event
where id NOT IN (select e.id
from event e, booking b where  e.id = b.event_id);

-- Task 4
-- 1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery
/*
projection: ticket price of event
criteria: venue
*/

select v.venue_name, AVG(e.ticket_price) as Average_Ticket_price
from venue v JOIN event e ON v.id=e.venue_id
group by v.venue_name;

/*
        venue_name     Average_Ticket_price
        chennai                    3500
```

```
        mumbai                          8000
        pondicherry     600
*/

-- Find Events with More Than 50% of Tickets Sold using subquery.
/*
        Analysis: If (total_seats-available seats) > (total_seats/2) -- this event shd be part of RS
            (320-270) > (320/2) -- this will not be displayed
*/

select *
from event
where (total_seats-available_seats) > (total_seats/2);

-- 3. Calculate the Total Number of Tickets Sold for Each Event.
/*
Analysis: tickets_sold = (total_seats-available seats)
*/

select event_name, SUM(total_seats-available_seats) as Tickets_Sold
from event
group by event_name;

-- 4. Find Customer Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery
/* Project : customer
condition: booking table */

select *
from customer
where id NOT IN (select distinct c.id
from customer c JOIN booking b ON c.id = b.customer_id);
```

```
/*
        7       frodo baggins   frodo@lotr.com          35454
*/

-- EXISTS and NOT-EXISTS
select *
from venue;
-- we want the above query to display results if and only if the below query returns atleast 1 record

select *
from event
where total_seats>27000; -- 1 row

select *
from venue
where EXISTS (select *
from event
where total_seats>29000);

-- EXISTS: for the outer query to run and show result, the inner query must return atleast 1 record.

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

select event_name, SUM(total_seats-available_seats) as Total_tickets_sold
from event
group by event_name;

select dt.event_name, SUM(dt.total_seats-dt.available_seats) as Total_tickets_sold
from (select * from event) as dt
group by event_name;
```

-- Display events with number of tickets_sold. consider those events where venue is in given list ['mumbai','chennai']

```
select event_name, SUM(total_seats-available_seats) as Total_tickets_sold
from (  select event_name,total_seats,available_seats
                from event e JOIN venue v ON e.venue_id=v.id
                where venue_name IN ('mumbai','chennai')) as dt
group by event_name;
```

```
select event_name, SUM(total_seats-available_seats) as Total_tickets_sold
from event e JOIN venue v ON e.venue_id=v.id
where venue_name IN ('mumbai','chennai')
group by event_name;
```

-- NOT IN , EXISTS-NOT EXISTS , Query in From statement - drived table/virtual

-- Calculate the Total Revenue Generated by Events for Each Customer Using a Correlated Subquery

```
select c.customer_name,SUM(total_cost)
from booking b JOIN customer c ON b.customer_id = c.id
group by c.customer_name;
```