create database ticketbookingsystem;

use TicketBookingSystem;

drop database TicketBookingSystem;

CREATE TABLE Venue (

venue\_id INT AUTO\_INCREMENT PRIMARY KEY ,

venue\_name VARCHAR(255),

address VARCHAR(255)

);

CREATE TABLE Event (

event\_id INT AUTO\_INCREMENT PRIMARY KEY,

event\_name VARCHAR(255),

event\_date DATE,

event\_time TIME,

venue\_id INT,

total\_seats INT,

available\_seats INT,

ticket\_price DECIMAL(10, 2),

event\_type VARCHAR(50) CHECK (event\_type IN ('Movie', 'Sports', 'Concert')),

booking\_id INT

);

CREATE TABLE Customer (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(255),

email VARCHAR(255),

phone\_number VARCHAR(20),

booking\_id INT

);

CREATE TABLE Booking (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

event\_id INT,

num\_tickets INT,

total\_cost DECIMAL(10, 2),

booking\_date DATE

);

use ticketbookingsystem;

ALTER TABLE Event

ADD FOREIGN KEY (venue\_id) REFERENCES venue(venue\_id);

ALTER TABLE Event

ADD FOREIGN KEY (booking\_id) REFERENCES Booking(booking\_id);

ALTER TABLE Customer

ADD FOREIGN KEY (booking\_id) REFERENCES Booking(booking\_id);

ALTER TABLE Booking

ADD FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id);

ALTER TABLE Booking

ADD FOREIGN KEY (event\_id) REFERENCES Event(event\_id);

INSERT INTO Venue

(venue\_id, venue\_name, address)

values

(1,'rock ','btm '),

(2,'band ','rajiv nagar '),

(3,'kill ','marthalli '),

(4,'comedy ','hosur '),

(5,'skippy ','nagar ');

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, 'show ', '2024-02-01', '18:30:00', 1, 100, 100, 10.50, 'Movie', 1),

(2, 'Basketball ', '2024-02-05', '20:00:00', 2, 500, 500, 25.00, 'Sports', 2),

(3, 'Rock ', '2024-03-10', '19:00:00', 3, 200, 200, 35.00, 'Concert', 3),

(4, 'Comedy ', '2024-04-15', '21:15:00', 4, 150, 150, 20.00, 'Concert', 4),

(5, 'Football Match', '2024-05-20', '17:45:00', 5, 300, 300, 30.00, 'Sports', 5);

INSERT INTO CUSTOMER

(CUSTOMER\_ID ,CUSTOMER\_NAME,email,phone\_number,BOOKING\_ID)

VALUES

(1,'ramesh','ramesh@GMAIL.COM',9783838388,1),

(2,'suresh','suresh@GMAIL.COM',6369693469,2),

(3,'ram','ram@GMAIL.COM',8943798944,3),

(4,'kabir','kabir@GMAIL.COM',78735883788,4),

(5,'singh','singh@GMAIL.COM',570375975,5);

INSERT INTO BOOKING

(BOOKING\_ID,CUSTOMER\_ID,EVENT\_ID,NUM\_TICKETS,TOTAL\_COST,BOOKING\_DATE)

VALUES

(1,1,1,1,9000.00,'2024-03-20'),

(2,2,2,2,1500.00,'2024-06-02'),

(3,3,3,3,10500.00,'2024-01-28'),

(4,4,4,4,4500.00,'2024-04-18'),

(5,5,5,5,3000.00,'2024-04-05');

SELECT \* FROM BOOKING;

SELECT \* FROM customer;

SELECT \* FROM event;

SELECT \* FROM venu;

SELECT \*

FROM Event

WHERE available\_seats > 0;

SELECT \*

FROM Event

WHERE event\_name LIKE '%cup%';

SELECT \*

FROM Event

WHERE ticket\_price BETWEEN 1000 AND 2500;

SELECT \*

FROM Event

WHERE event\_date BETWEEN '2024-04-11' AND '2024-04-21';

SELECT \*

FROM Event

WHERE available\_seats > 0

AND event\_type = 'Concert';

SELECT \*

FROM customer

ORDER BY customer\_id

LIMIT 5 OFFSET 5;

SELECT \*

FROM Booking

WHERE num\_tickets > 4;

SELECT \*

FROM Customer

WHERE phone\_number LIKE '%000';

SELECT \*

FROM Event

WHERE total\_seats > 15000

ORDER BY total\_seats DESC;

SELECT \*

FROM Event

WHERE event\_name NOT LIKE 'x%'

AND event\_name NOT LIKE 'y%'

AND event\_name NOT LIKE 'z%';

/\* task\_3 \*/

SELECT

EVENT\_ID,EVENT\_NAME,

AVG(TICKET\_PRICE)

FROM EVENT

GROUP BY EVENT\_ID,

EVENT\_NAME;

SELECT

SUM(total\_cost) AS total\_revenue

FROM

Booking;

SELECT EVENT\_ID,

EVENT\_NAME,

SUM(NUM\_TICKETS) AS HIGHEST\_TICKET\_SALES

FROM BOOKING

GROUP BY

EVENT\_ID,EVENT\_NAME

ORDER BY

HIGHEST\_TICKET\_SALES DESC

LIMIT 1;

SELECT EVENT\_ID,

SUM(NUM\_TICKETS) AS TOTAL\_TICKET\_SOLD

FROM BOOKING

GROUP BY EVENT\_ID;

SELECT

event\_name

FROM

Event e

LEFT JOIN

Booking b ON event\_name = event\_name

WHERE

event\_name IS NULL;

SELECT

c.customer\_id,

c.customer\_name,

SUM(b.num\_tickets) AS total\_tickets\_booked

FROM

Customer c

JOIN

Booking b ON c.customer\_id = b.customer\_id

GROUP BY

c.customer\_id, c.customer\_name

ORDER BY

total\_tickets\_booked DESC

LIMIT 1;

SELECT

e.event\_id,

e.event\_name,

MONTH(b.booking\_date) AS booking\_month,

SUM(b.num\_tickets) AS total\_tickets\_sold

FROM

Event e

JOIN

Booking b ON e.event\_id = b.event\_id

GROUP BY

e.event\_id, e.event\_name, booking\_month

ORDER BY

booking\_month, e.event\_id;

SELECT

v.venue\_id,

v.venue\_name,

AVG(e.ticket\_price) AS average\_ticket\_price

FROM

Venu v

JOIN

Event e ON v.venue\_id = e.venue\_id

GROUP BY

v.venue\_id, v.venue\_name;

SELECT

e.event\_type,

SUM(b.num\_tickets) AS total\_tickets\_sold

FROM

Event e

JOIN

Booking b ON e.event\_id = b.event\_id

GROUP BY

e.event\_type;

SELECT

YEAR(b.booking\_date) AS booking\_year,

SUM(b.total\_cost) AS total\_revenue

FROM

Booking b

GROUP BY

booking\_year;

SELECT

c.customer\_id,

c.customer\_name,

COUNT(DISTINCT b.event\_id) AS num\_events\_booked

FROM

Customer c

JOIN

Booking b ON c.customer\_id = b.customer\_id

GROUP BY

c.customer\_id, c.customer\_name

HAVING

num\_events\_booked > 1;

SELECT

c.customer\_id,

c.customer\_name,

SUM(b.total\_cost) AS total\_revenue

FROM

Customer c

JOIN

Booking b ON c.customer\_id = b.customer\_id

GROUP BY

c.customer\_id, c.customer\_name;

SELECT

e.event\_type,

v.venue\_name,

AVG(e.ticket\_price) AS average\_ticket\_price

FROM

Event e

JOIN

Venu v ON e.venue\_id = v.venue\_id

GROUP BY

e.event\_type, v.venue\_name;

SELECT

c.customer\_id,

c.customer\_name,

SUM(b.num\_tickets) AS total\_tickets\_purchased

FROM

Customer c

JOIN

Booking b ON c.customer\_id = b.customer\_id

WHERE

b.booking\_date >= CURDATE() - INTERVAL 30 DAY

GROUP BY

c.customer\_id, c.customer\_name;

SELECT

v.venue\_id,

v.venue\_name,

COALESCE(avg\_ticket\_price, 0) AS average\_ticket\_price

FROM

Venu v

LEFT JOIN

(SELECT

venue\_id,

AVG(ticket\_price) AS avg\_ticket\_price

FROM

Event

GROUP BY

venue\_id) subquery

ON

v.venue\_id = subquery.venue\_id;

SELECT

e.event\_id,

e.event\_name,

e.total\_seats,

COUNT(b.booking\_id) AS tickets\_sold

FROM

Event e

LEFT JOIN

Booking b ON e.event\_id = b.event\_id

GROUP BY

e.event\_id, e.event\_name, e.total\_seats

HAVING

COUNT(b.booking\_id) > 0.5 \* e.total\_seats;

SELECT

e.event\_id,

e.event\_name,

e.total\_seats,

COALESCE(SUM(b.num\_tickets), 0) AS total\_tickets\_sold

FROM

Event e

LEFT JOIN

Booking b ON e.event\_id = b.event\_id

GROUP BY

e.event\_id, e.event\_name, e.total\_seats;

SELECT

c.customer\_id,

c.customer\_name

FROM

Customer c

WHERE

NOT EXISTS (

SELECT 1

FROM Booking b

WHERE b.customer\_id = c.customer\_id

);

SELECT

event\_id,

event\_name

FROM

Event

WHERE

event\_id NOT IN (

SELECT DISTINCT event\_id

FROM Booking

);

SELECT

e.event\_type,

COALESCE(total\_tickets\_sold, 0) AS total\_tickets\_sold

FROM

(SELECT

event\_type,

SUM(num\_tickets) AS total\_tickets\_sold

FROM

Event e

LEFT JOIN

Booking b ON e.event\_id = b.event\_id

GROUP BY

event\_type) AS e

RIGHT JOIN

(SELECT DISTINCT event\_type FROM Event) AS et ON e.event\_type = et.event\_type;

SELECT

event\_id,

event\_name,

ticket\_price

FROM

Event

WHERE

ticket\_price > (

SELECT COALESCE(AVG(ticket\_price), 0)

FROM Event

);

SELECT

c.customer\_id,

c.customer\_name,

(

SELECT COALESCE(SUM(b.total\_cost), 0)

FROM Booking b

WHERE b.customer\_id = c.customer\_id

) AS total\_revenue

FROM

Customer c;

SELECT

c.customer\_id,

c.customer\_name

FROM

Customer c

WHERE

EXISTS (

SELECT 1

FROM Booking b

JOIN Event e ON b.event\_id = e.event\_id

WHERE b.customer\_id = c.customer\_id

AND e.venue\_id = <your\_venue\_id>

);

SELECT

e.event\_type,

COALESCE((

SELECT SUM(b.num\_tickets)

FROM Booking b

WHERE b.event\_id IN (

SELECT event\_id

FROM Event e1

WHERE e1.event\_type = e.event\_type

)

), 0) AS total\_tickets\_sold

FROM

Event e

GROUP BY

e.event\_type;

SELECT

c.customer\_id,

c.customer\_name,

DATE\_FORMAT(b.booking\_date, '%Y-%m') AS booking\_month

FROM

Customer c

JOIN

Booking b ON c.customer\_id = b.customer\_id

WHERE

EXISTS (

SELECT 1

FROM Booking b1

WHERE

b1.customer\_id = c.customer\_id

AND DATE\_FORMAT(b1.booking\_date, '%Y-%m') = DATE\_FORMAT(b.booking\_date, '%Y-%m')

);