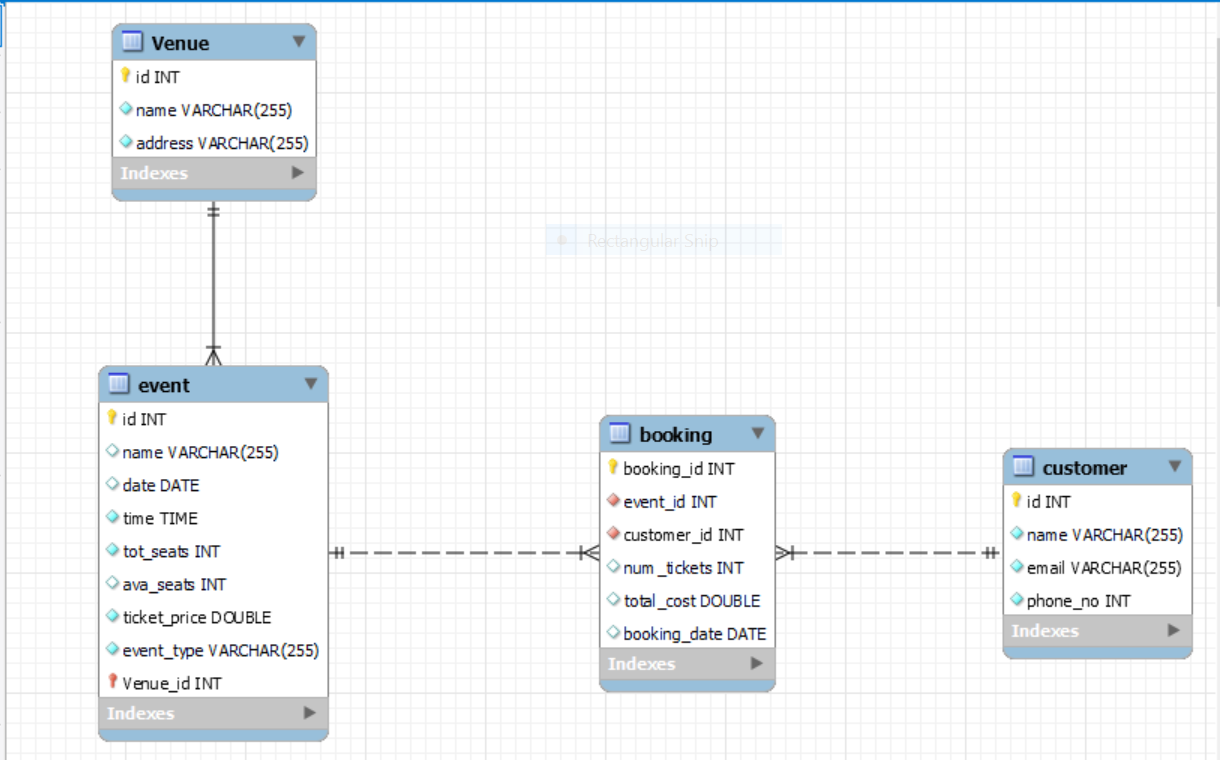
**Assignment 1-Ticket booking system**

Task 1: Database Design



CREATE SCHEMA IF NOT EXISTS `TicketBookingdb` DEFAULT CHARACTER SET utf8 ;

USE `TicketBookingdb` ;

-- -----------------------------------------------------

-- Table `TicketBookingdb`.`Venue`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `TicketBookingdb`.`Venue` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(255) NOT NULL,

`address` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `TicketBookingdb`.`event`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `TicketBookingdb`.`event` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(255) NULL,

`date` DATE NULL,

`time` TIME NOT NULL,

`tot\_seats` INT NOT NULL,

`ava\_seats` INT NULL,

`ticket\_price` DOUBLE NOT NULL,

`event\_type` VARCHAR(255) NOT NULL,

`Venue\_id` INT NOT NULL,

PRIMARY KEY (`id`, `Venue\_id`),

INDEX `fk\_event\_Venue\_idx` (`Venue\_id` ASC),

CONSTRAINT `fk\_event\_Venue`

FOREIGN KEY (`Venue\_id`)

REFERENCES `TicketBookingdb`.`Venue` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `TicketBookingdb`.`customer`

------------------------------------------------------

CREATE TABLE IF NOT EXISTS `TicketBookingdb`.`customer` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(255) NOT NULL,

`email` VARCHAR(255) NOT NULL,

`phone\_no` VARCHAR(10) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `TicketBookingdb`.`booking`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `TicketBookingdb`.`booking` (

`event\_id` INT NOT NULL,

`booking\_id` INT NOT NULL,

`customer\_id` INT NOT NULL,

`num\_tickets` INT NULL,

`total\_cost` DOUBLE NULL,

`booking\_date` DATETIME NULL,

`bookingcol` VARCHAR(45) NULL,

PRIMARY KEY (`event\_id`, `booking\_id`, `customer\_id`),

INDEX `fk\_event\_has\_customer\_customer1\_idx` (`customer\_id` ASC) ,

INDEX `fk\_event\_has\_customer\_event1\_idx` (`event\_id` ASC, `booking\_id` ASC) ,

CONSTRAINT `fk\_event\_has\_customer\_event1`

FOREIGN KEY (`event\_id` , `booking\_id`)

REFERENCES `TicketBookingdb`.`event` (`id` , `Venue\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_event\_has\_customer\_customer1`

FOREIGN KEY (`customer\_id`)

REFERENCES `TicketBookingdb`.`customer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-----INSERATIONS-----

use TicketBookingdb;

show tables;

insert into venue(name, address) values

('mumbai', 'marol andheri(w)'),

('chennai', 'IT Park'),

('pondicherry', 'state beach');

Insert into customer(name, email, phone\_no)

values

('harry potter', 'harry@gmail.com','45454545'),

('ronald weasley', 'ron@gmail.com', '45454545'),

('hermione granger', 'her@gmail.com', '45454545'),

('draco malfoy', 'drac@gmail.com', '45454545'),

('ginni weasley', 'ginni@gmail.com', '45454000');

insert into

event(name,date,time,tot\_seats, ava\_seats, ticket\_price,event\_type,venue\_id)

values

('Late Ms. Lata Mangeshkar Musical', '2021-09-12','20:00',320,270,600, 'concert', 3),

(CSK VS RCB, '2024-04-11', '19:30', 23000,3,3600, 'sports', 2),

('CSK vs RR', '2024-04-19', '19:30', 23000,10,2400, 'sports', 2),

('MI vs KKR', '2024-05-01', '15:30', 28000,100,2000, 'sports', 1);

insert into booking (event\_id,customer\_id,num\_tickets,total\_cost,booking\_date) values

(1,1,2,640,'2021-09-12'),

(1,4,3,960,'2021-09-12'),

(2,1,3,10800,'2024-04-11'),

(2,3,5,18000,'2024-04-10'),

(3,5,10,34000,'2024-04-15'),

(4,2,4,32000,'2024-05-01');

TASK 2

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

**select \* from event;**

/\*+----+----------------------------------+------------+----------+-----------+-----------+--------------+------------+----------+

| id | name | date | time | tot\_seats | ava\_seats | ticket\_price | event\_type | Venue\_id |

+----+----------------------------------+------------+----------+-----------+-----------+--------------+------------+----------+

| 1 | Late Ms. Lata Mangeshkar Musical | 2021-09-12 | 20:00:00 | 320 | 270 | 600 | concert | 3 |

| 2 | CSK vs RCB | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | 2 |

| 3 | CSK vs RR | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | 2 |

| 4 | MI vs KKR | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

+----+----------------------------------+------------+----------+-----------+-----------+--------------+------------+----------+\*/

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

**select event.name, ava\_seats from event where ava\_seats>0;**

/\*+----------------------------------+-----------+

| name | ava\_seats |

+----------------------------------+-----------+

| Late Ms. Lata Mangeshkar Musical | 270 |

| CSK vs RCB | 3 |

| CSK vs RR | 10 |

| MI vs KKR | 100 |

| Late Ms. Lata Mangeshkar Musical | 270 |

+----------------------------------+-----------+\*/

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

**select event.name**

**from event**

**where name like '%csk%';**

/\*+------------+

| name |

+------------+

| CSK vs RCB |

| CSK vs RR |\*/

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

**select name, ticket\_price from event**

**where ticket\_price between 600 and 2000;**

/\*+----------------------------------+--------------+

| name | ticket\_price |

+----------------------------------+--------------+

| Late Ms. Lata Mangeshkar Musical | 600 |\*/

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

**select name, date from event where date BETWEEN '2024-04-01' AND '2024-04-30';**

/\*+------------+------------+

| name | date |

+------------+------------+

| CSK vs RCB | 2024-04-11 |

| CSK vs RR | 2024-04-19 |

+------------+------------+\*/

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

**select name, ava\_seats from event where name LIKE '%RCB%';**

/\*+------------+-----------+

| name | ava\_seats |

+------------+-----------+

| CSK vs RCB | 3 |

+------------+-----------+\*/

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

**select \***

**From customer**

**limit 5,5;**

/\* Empty set\*/

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

**select \***

**from booking**

**where num\_tickets > 4;**

/\* +------------+----------+-------------+-------------+------------+--------------+

| booking\_id | event\_id | customer\_id | num\_tickets | total\_cost | booking\_date |

+------------+----------+-------------+-------------+------------+--------------+

| 10 | 2 | 3 | 5 | 18000 | 2024-04-10 |

| 11 | 3 | 5 | 10 | 34000 | 2024-04-15 |

| 19 | 2 | 3 | 5 | 18000 | 2024-04-10 |

| 20 | 3 | 5 | 10 | 34000 | 2024-04-15 |

+------------+----------+-------------+-------------+------------+--------------+\*/

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

**select \***

**from customer**

**where phone\_no LIKE '%000';**

/\*+----+---------------+-----------------+----------+

| id | name | email | phone\_no |

+----+---------------+-----------------+----------+

| 5 | ginni weasley | ginni@gmail.com | 45454000 |

+----+---------------+-----------------+----------+\*/

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

**select name, tot\_seats**

**from event where tot\_seats>15000**

**order by tot\_seats desc;**

/\*+------------+-----------+

| name | tot\_seats |

+------------+-----------+

| MI vs KKR | 28000 |

| CSK vs RCB | 23000 |

| CSK vs RR | 23000 |

+------------+-----------+\*/

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

**Select name**

**from event**

**where name NOT LIKE 'x%' AND name NOT LIKE 'y%' AND name NOT LIKE 'z%';**

/\* +----------------------------------+

| name |

+----------------------------------+

| Late Ms. Lata Mangeshkar Musical |

| CSK vs RCB |

| CSK vs RR |

| MI vs KKR |

+----------------------------------+\*/

**TASK 3:**

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

**select e.name, AVG(b.total\_cost)**

**from event e, booking b where e.id=b.event\_id**

**group by e.name;**

/\*+----------------------------------+-------------------+

| name | avg(b.total\_cost) |

+----------------------------------+-------------------+

| CSK vs RCB | 14400 |

| CSK vs RR | 34000 |

| Late Ms. Lata Mangeshkar Musical | 800 |

| MI vs KKR | 32000 |

+----------------------------------+-------------------+\*/

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

**select e.name, sum(b.total\_cost)**

**from event e**

**JOIN booking b ON e.id=b.event\_id**

**group by e.name;**

/\* +----------------------------------+-------------------+

| name | sum(b.total\_cost) |

+----------------------------------+-------------------+

| CSK vs RCB | 57600 |

| CSK vs RR | 68000 |

| Late Ms. Lata Mangeshkar Musical | 3200 |

| MI vs KKR | 64000 |

+----------------------------------+-------------------+\*/

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

**select e.name,e.tot\_seats-e.ava\_seats as 'sold\_seats'**

**from event e**

**order by e.tot\_seats-e.ava\_seats desc**

**limit 1;**

/\*+-----------+------------+

| name | sold\_seats |

+-----------+------------+

| MI vs KKR | 27900 |

+-----------+------------+\*/

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

**select e.name,e.tot\_seats-e.ava\_seats as 'sold\_seats' from event e;**

/\*+----------------------------------+------------+

| name | sold\_seats |

+----------------------------------+------------+

| Late Ms. Lata Mangeshkar Musical | 50 |

| CSK vs RCB | 22997 |

| CSK vs RR | 22990 |

| MI vs KKR | 27900 |

+----------------------------------+------------+\*/

--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);**

/\*null\*/

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

**select c.name, sum(b.num\_tickets)as total\_tickets**

**from event e, booking b, customer c**

**where e.id=b.event\_id and b.customer\_id=c.id**

**group by c.id**

**order by total\_tickets desc limit 1;**

/\*+---------------+---------------+

| name | total\_tickets |

+---------------+---------------+

| ginni weasley | 10 |

+---------------+---------------+\*/

--7. Write a SQL query to List Events and the total number of tickets sold for each month.

--8. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.

**select v.name, avg(ticket\_price) as average\_price from event e, venue v where v.id=e.venue\_id group by v.name;**

+-------------+---------------+

| name | average\_price |

+-------------+---------------+

| chennai | 3500 |

| mumbai | 8000 |

| pondicherry | 600 |

+-------------+---------------+

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

select e.event\_type, SUM(num\_tickets) from event e JOIN booking b ON e.id=b.event\_id group by event\_type;

+------------+------------------+

| event\_type | SUM(num\_tickets) |

+------------+------------------+

| concert | 5|

| sports | 22|

+------------+------------------+\*/

--10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.

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

**select c.name, c.email from booking b, customer c where c.id = b.customer\_id group by c.id having count(c.id) >1;**

/\*+------------------+-----------------+

| name | email |

+------------------+-----------------+

| harry potter | harry@gmail.com |\*/

--12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.

**select c.name, SUM(b.total\_cost) from customer c, booking b where c.id=b.customer\_id**

**group by c.name;**

/\*

+------------------+-------------------+

| name | SUM(b.total\_cost) |

+------------------+-------------------+

| draco malfoy | 1920 |

| ginni weasley | 68000 |

| harry potter | 22880 |

| hermione granger | 36000 |

| ronald weasley | 64000 |

+------------------+-------------------+\*/

--13. 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) from event e, venue v where e.venue\_id=v.id group by e.event\_type, v.name;**

+------------+---------------------+

| event\_type | AVG(e.ticket\_price) |

+------------+---------------------+

| concert | 600 |

| sports | 4650 |

+------------+---------------------+

**TASK 4:**

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

**select v.name, AVG(e.ticket\_price) from venue v, event e where e.venue\_id=v.id group by v.name;**

/\*+-------------+---------------------+

| name | AVG(e.ticket\_price) |

+-------------+---------------------+

| chennai | 3500 |

| mumbai | 8500 |

| pondicherry | 600 |

+-------------+---------------------+\*/

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

**select \* from event WHERE tot\_seats-ava\_seats>tot\_seats/2;**

/\*+----+------------+------------+----------+-----------+-----------+--------------+------------+----------+

| id | name | date | time | tot\_seats | ava\_seats | ticket\_price | event\_type | Venue\_id |

+----+------------+------------+----------+-----------+-----------+--------------+------------+----------+

| 2 | CSK vs RCB | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | 2 |

| 3 | CSK vs RR | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | 2 |

| 4 | MI vs KKR | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

+----+------------+------------+----------+-----------+-----------+--------------+------------+----------+\*/

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

**select name, SUM(tot\_seats-ava\_seats) as ticket\_sold from event group by name;**

/\*+----------------------------------+-------------+

| name | ticket\_sold |

+----------------------------------+-------------+

| CSK vs RCB | 22997 |

| CSK vs RR | 22990 |

| Late Ms. Lata Mangeshkar Musical | 50 |

| MI vs KKR | 27900 |

+----------------------------------+-------------+\*/

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

**select \* from customer where id not in (select customer\_id from booking);**

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

**select \* from event where id not in (select event\_id from booking);**

/\*empty set\*/

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

**select name, sum(tot\_seats-ava\_seats) from (select \* from event) as dt group by event\_type;**

/\*+----------------------------------+--------------------------+

| name | sum(tot\_seats-ava\_seats) |

+----------------------------------+--------------------------+

| Late Ms. Lata Mangeshkar Musical | 50 |

| CSK vs RCB | 73887 |

+----------------------------------+--------------------------+\*/

--7. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.

**select \* from event where ticket\_price>(select avg(ticket\_price) from event);**

/\*+----+-----------+------------+----------+-----------+-----------+--------------+------------+----------+

| id | name | date | time | tot\_seats | ava\_seats | ticket\_price | event\_type | Venue\_id |

+----+-----------+------------+----------+-----------+-----------+--------------+------------+----------+

| 4 | MI vs KKR | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

+----+-----------+------------+----------+-----------+-----------+--------------+------------+----------+\*/

--8. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.

**select c.name, SUM(b.total\_cost) from customer c, booking b where c.id=b.customer\_id group by c.name;**

/\*+------------------+-------------------+

| name | SUM(b.total\_cost) |

+------------------+-------------------+

| draco malfoy | 1920 |

| ginni weasley | 68000 |

| harry potter | 22880 |

| hermione granger | 36000 |

| ronald weasley | 64000 |

+------------------+-------------------+\*/

--9. 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 name='chennai')));**

/\*+----+------------------+-----------------+----------+

| id | name | email | phone\_no |

+----+------------------+-----------------+----------+

| 1 | harry potter | harry@gmail.com | 45454545 |

| 3 | hermione granger | her@gmail.com | 45454545 |

| 5 | ginni weasley | ginni@gmail.com | 45454545 |

+----+------------------+-----------------+----------+\*/

--10. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY.

**select event\_type, SUM(tot\_seats-ava\_seats) from event group by event\_type;**

/\*+------------+--------------------------+

| event\_type | SUM(tot\_seats-ava\_seats) |

+------------+--------------------------+

| concert | 50 |

| sports | 73887 |

+------------+--------------------------+\*/

--11. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE\_FORMAT.

--12. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery

**select v.name, AVG(e.ticket\_price) from venue v JOIN event e ON e.venue\_id=v.id group by v.name;**

+-------------+---------------------+

| name | AVG(e.ticket\_price) |

+-------------+---------------------+

| chennai | 3500 |

| mumbai | 8000 |

| pondicherry | 600 |

+-------------+---------------------+