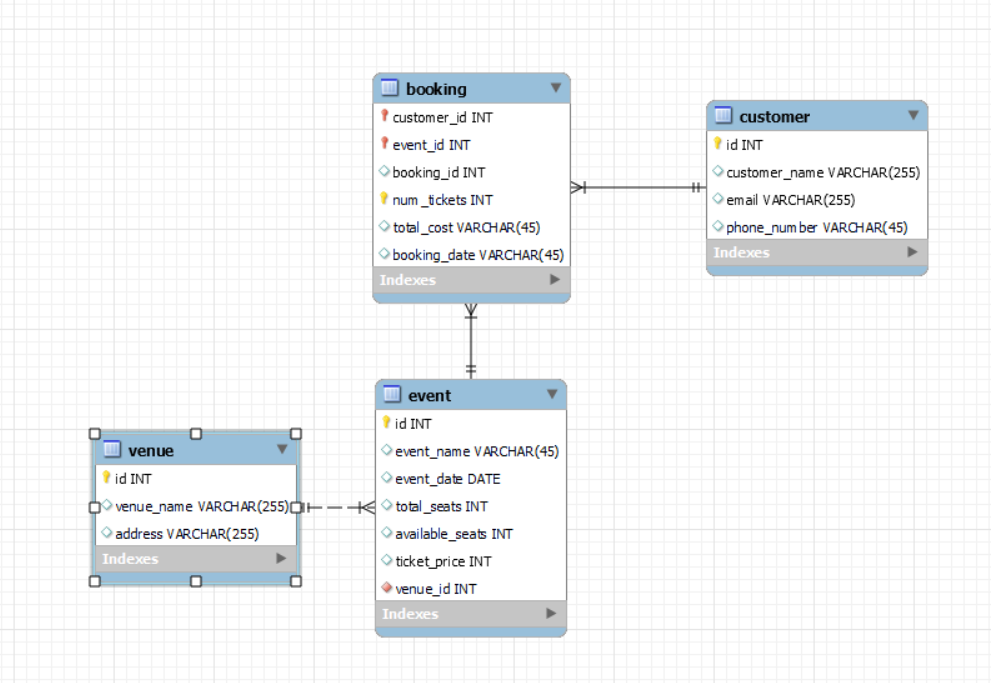
TICKET BOOKING – ASSIGNMENT 1



CREATE SCHEMA IF NOT EXISTS `ticketbooking\_feb\_hex\_24` DEFAULT CHARACTER SET utf8 ;

USE `ticketbooking\_feb\_hex\_24` ;

CREATE TABLE IF NOT EXISTS `ticketbooking\_feb\_hex\_24`.`venue` (

`id` INT NOT NULL AUTO\_INCREMENT,

`venue\_name` VARCHAR(45) NOT NULL,

`address` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `ticketbooking\_feb\_hex\_24`.`event` (

`id` INT NOT NULL AUTO\_INCREMENT,

`event\_name` VARCHAR(45) NULL,

`event\_date` DATE NULL,

`event\_time` TIME NULL,

`total\_seats` INT NULL,

`available\_seats` INT NULL,

`ticket\_price` DOUBLE NULL,

`event\_type` VARCHAR(45) NULL,

`venue\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_event\_venue\_idx` (`venue\_id` ASC),

CONSTRAINT `fk\_event\_venue`

FOREIGN KEY (`venue\_id`)

REFERENCES `ticketbooking\_feb\_hex\_24`.`venue` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `ticketbooking\_feb\_hex\_24`.`customer` (

`id` INT NOT NULL AUTO\_INCREMENT,

`customer\_name` VARCHAR(45) NULL,

`email` VARCHAR(45) NULL,

`phone\_number` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `ticketbooking\_feb\_hex\_24`.`booking` (

`event\_id` INT NOT NULL,

`customer\_id` INT NOT NULL,

`num\_tickets` INT NULL,

`total\_cost` DOUBLE NULL,

`booking\_date` DATE NULL,

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

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

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

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

FOREIGN KEY (`event\_id`)

REFERENCES `ticketbooking\_feb\_hex\_24`.`event` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

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

FOREIGN KEY (`customer\_id`)

REFERENCES `ticketbooking\_feb\_hex\_24`.`customer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

**Tasks 2: Select, Where, Between, AND, LIKE:**

1.use ticketbooking\_feb\_hex\_24;

#insertions

insert into venue(venue\_name,address) values

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

('chennai', 'IT Park'),

('pondicherry ', 'state beach');

select \* from venue;

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

| id | venue\_name | address |

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

| 1 | mumbai | marol andheri(w) |

| 2 | chennai | IT Park |

| 3 | pondicherry | state beach |

| 4 | mumbai | marol andheri(w) |

| 5 | chennai | IT Park |

| 6 | pondicherry | state beach |

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

insert into customer(customer\_name,email,phone\_number)

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','45454545');

select \* from customer;

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

| id | customer\_name | email | phone\_number |

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

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

| 2 | ronald weasley | ron@gmail.com | 45454545 |

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

| 4 | draco malfoy | drac@gmail.com | 45454545 |

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

| 6 | harry potter | harry@gmail.com | 45454545 |

| 7 | ronald weasley | ron@gmail.com | 45454545 |

| 8 | hermione granger | her@gmail.com | 45454545 |

| 9 | draco malfoy | drac@gmail.com | 45454545 |

| 10 | ginni weasley | ginni@gmail.com | 45454545 |

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

insert into event(event\_name,event\_date,event\_time,total\_seats,available\_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',23000,10,3400,'sports',2),

('MI vs KKR', '2024-05-01','15:30',28000,100,8000,'sports',1);

select \* from event;

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

| id | event\_name | event\_date | event\_time | total\_seats | available\_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 | Conferece CUP | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

| 5 | Late Ms. Lata Mangeshkar Musical | 2021-09-12 | 20:00:00 | 320 | 270 | 600 | concert | 3 |

| 6 | CSK vs RCB | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | 2 |

| 7 | CSK vs RR | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | 2 |

| 8 | MI vs KKR | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

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

insert into booking values

(4,1,2,640,'2021-09-12'),

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

(5,1,3,10800,'2024-04-11'),

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

(6,5,10,34000,'2024-04-15'),

(7,2,4,32000,'2024-05-01');

2. write a sql query to list all events.

select \* from event;

update event set event\_name='conferece cup' where id=7;

3. write a sql query to select events with available tickets.

select e.id, e.event\_name, e.event\_date, e.event\_time, e.total\_seats,

e.available\_seats, e.ticket\_price, e.event\_type,v.venue\_name

from ticketbooking\_feb\_hex\_24.event e

join ticketbooking\_feb\_hex\_24.venue v on e.venue\_id = v.id

where e.available\_seats > 0;

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

| id | event\_name | event\_date | event\_time | total\_seats | available\_seats | ticket\_price | event\_type | venue\_name |

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

| 4 | conferece cup | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | mumbai |

| 8 | mi vs kkr | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | mumbai |

| 12 | mi vs kkr | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | mumbai |

| 2 | csk vs rcb | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | chennai |

| 3 | csk vs rr | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | chennai |

| 6 | csk vs rcb | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | chennai |

| 7 | csk vs rr | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | chennai |

| 10 | csk vs rcb | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | chennai |

| 11 | csk vs rr | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | chennai |

| 1 | late ms. lata mangeshkar musical | 2021-09-12 | 20:00:00 | 320 | 270 | 600 | concert | pondicherry |

| 5 | late ms. lata mangeshkar musical | 2021-09-12 | 20:00:00 | 320 | 270 | 600 | concert | pondicherry |

| 9 | late ms. lata mangeshkar musical | 2021-09-12 | 20:00:00 | 320 | 270 | 600 | concert | pondicherry |

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

4. write a sql query to select events name partial match with ‘cup’.

select \*

from event

where event\_name like '%cup%';

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

| id | event\_name | event\_date | event\_time | total\_seats | available\_seats | ticket\_price | event\_type | venue\_id |

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

| 7 | conferece cup | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

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

5. write a sql query to select events with ticket price range is between 1000 to 2500.

select e.id, e.event\_name, e.event\_date, e.event\_time, e.total\_seats,

e.available\_seats, e.ticket\_price,e.event\_type, v.venue\_name

from ticketbooking\_feb\_hex\_24.event e

join ticketbooking\_feb\_hex\_24.venue v on e.venue\_id = v.id

where e.ticket\_price between 1000 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';

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

| id | event\_name | event\_date | event\_time | total\_seats | available\_seats | ticket\_price | event\_type | venue\_id |

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

| 5 | csk vs rcb | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | 2 |

| 6 | csk vs rr | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | 2 |

| 7 | conferece cup | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

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

7. write a sql query to retrieve events with available tickets that also have "concert" in their

name.

select e.id, e.event\_name, e.event\_date,e.event\_time,e.total\_seats,

e.available\_seats,e.ticket\_price, e.event\_type,v.venue\_name

from ticketbooking\_feb\_hex\_24.event e

join ticketbooking\_feb\_hex\_24.venue v on e.venue\_id = v.id

where e.available\_seats > 0

and e.event\_name 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;

select \*

from customer

limit 5,5;

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

| id | customer\_name | email | phone\_number |

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

| 4 | draco malfoy | drac@gmail.com | 45454545 |

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

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

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

| id | customer\_name | email | phone\_number |

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

| 6 | harry potter | harry@gmail.com | 45454545 |

| 7 | ronald weasley | ron@gmail.com | 45454545 |

| 8 | hermione granger | her@gmail.com | 45454545 |

| 9 | draco malfoy | drac@gmail.com | 45454545 |

| 10 | ginni weasley | ginni@gmail.com | 45454545 |

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

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

select b.event\_id, b.customer\_id, b.num\_tickets, b.total\_cost, b.booking\_date

from ticketbooking\_feb\_hex\_24.booking b

where b.num\_tickets > 4;

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

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

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

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

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

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

| 6 | 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\_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 ;

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

| id | event\_name | event\_date | event\_time | total\_seats | available\_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 |

| 6 | csk vs rcb | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | 2 |

| 7 | csk vs rr | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | 2 |

| 10 | csk vs rcb | 2024-04-11 | 19:30:00 | 23000 | 3 | 3600 | sports | 2 |

| 11 | csk vs rr | 2024-04-19 | 19:30:00 | 23000 | 10 | 3400 | sports | 2 |

| 4 | conferece cup | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

| 8 | mi vs kkr | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

| 12 | mi vs kkr | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

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

-- 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%';

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

| id | event\_name | event\_date | event\_time | total\_seats | available\_seats | ticket\_price | event\_type | venue\_id |

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

| 1 | late ms. lata mangeshkar musical | 2021-09-12 | 20:00:00 | 320 | 270 | 600 | concert | 3 |

| 5 | late ms. lata mangeshkar musical | 2021-09-12 | 20:00:00 | 320 | 270 | 600 | concert | 3 |

| 8 | mi vs kkr | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

| 9 | late ms. lata mangeshkar musical | 2021-09-12 | 20:00:00 | 320 | 270 | 600 | concert | 3 |

| 12 | mi vs kkr | 2024-05-01 | 15:30:00 | 28000 | 100 | 8000 | sports | 1 |

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

**task 3: aggregate functions, having, order by, groupby and joins:**

1. write a sql query to list events and their average ticket prices.

select e.id,e.event\_name,

avg(b.ticket\_price) as average\_ticket\_price

from ticketbooking\_feb\_hex\_24.event e

join ticketbooking\_feb\_hex\_24.booking b on e.id = b.event\_id

group by e.id, e.event\_name;

2. write a sql query to calculate the total revenue generated by events.

select

sum(b.total\_cost) as total\_revenue

from

ticketbooking\_feb\_hex\_24.booking b;

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

| total\_revenue |

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

| 192800 |

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

3. write a sql query to find the event with the highest ticket sales.

select

event\_id,

sum(num\_tickets) as total\_tickets\_sold

from

ticketbooking\_feb\_hex\_24.booking

group by

event\_id

order by

total\_tickets\_sold desc

limit 1;

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

| event\_id | total\_tickets\_sold |

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

| 3 | 10 |

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

4. write a sql query to calculate the total number of tickets sold for each event.

select

event\_id,

sum(num\_tickets) as total\_tickets\_sold

from

ticketbooking\_feb\_hex\_24.booking

group by

event\_id;

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

| event\_id | total\_tickets\_sold |

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

| 1 | 5 |

| 2 | 8 |

| 3 | 10 |

| 4 | 9 |

| 5 | 8 |

| 6 | 10 |

| 7 | 4 |

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

5. write a sql query to find events with no ticket sales.

select

e.id,

e.event\_name

from

ticketbooking\_feb\_hex\_24.event e

left join

ticketbooking\_feb\_hex\_24.booking b on e.id = b.event\_id

where

b.event\_id is null;

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

| id | event\_name |

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

| 8 | mi vs kkr |

| 9 | late ms. lata mangeshkar musical |

| 10 | csk vs rcb |

| 11 | csk vs rr |

| 12 | mi vs kkr |

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

6. write a sql query to find the user who has booked the most tickets.

select

customer\_id,

sum(num\_tickets) as total\_tickets\_booked

from

ticketbooking\_feb\_hex\_24.booking

group by

customer\_id

order by

total\_tickets\_booked desc

limit 1;

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

| customer\_id | total\_tickets\_booked |

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

| 5 | 20 |

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

7. write a sql query to list events and the total number of tickets sold for each month.

select

month(booking\_date) as month,

year(booking\_date) as year,

event\_id,

sum(num\_tickets) as total\_tickets\_sold

from

ticketbooking\_feb\_hex\_24.booking

group by

year, month, event\_id;

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

| month | year | event\_id | total\_tickets\_sold |

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

| 9 | 2021 | 1 | 5 |

| 9 | 2021 | 4 | 5 |

| 4 | 2024 | 2 | 8 |

| 4 | 2024 | 3 | 10 |

| 4 | 2024 | 5 | 8 |

| 4 | 2024 | 6 | 10 |

| 5 | 2024 | 4 | 4 |

| 5 | 2024 | 7 | 4 |

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

8. write a sql query to calculate the average ticket price for events in each venue.

select

e.venue\_id,

avg(b.ticket\_price) as average\_ticket\_price

from

ticketbooking\_feb\_hex\_24.event e

join

ticketbooking\_feb\_hex\_24.booking b on e.id = b.event\_id

group by

e.venue\_id;

9. 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 total\_tickets\_sold

from

ticketbooking\_feb\_hex\_24.event e

join

ticketbooking\_feb\_hex\_24.booking b on e.id = b.event\_id

group by

e.event\_type;

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

| event\_type | total\_tickets\_sold |

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

| concert | 13 |

| sports | 41 |

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

10. write a sql query to calculate the total revenue generated by events in each year.

select

year(booking\_date) as year,

sum(total\_cost) as total\_revenue

from

ticketbooking\_feb\_hex\_24.booking

group by

year;

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

| year | total\_revenue |

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

| 2021 | 3200 |

| 2024 | 189600 |

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

11. write a sql query to list users who have booked tickets for multiple events.

select

customer\_id

from

ticketbooking\_feb\_hex\_24.booking

group by

customer\_id

having

count(distinct event\_id) > 1;

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

| customer\_id |

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

| 1 |

| 2 |

| 3 |

| 4 |

| 5 |

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

12. write a sql query to calculate the total revenue generated by events for each user

select

customer\_id,

sum(total\_cost) as total\_revenue

from

ticketbooking\_feb\_hex\_24.booking

group by

customer\_id;

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

| customer\_id | total\_revenue |

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

| 1 | 22880 |

| 2 | 64000 |

| 3 | 36000 |

| 4 | 1920 |

| 5 | 68000 |

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

.

13. write a sql query to calculate the average ticket price for events in each category and venue.

select

e.event\_type,

e.venue\_id,

avg(b.ticket\_price) as average\_ticket\_price

from

ticketbooking\_feb\_hex\_24.event e

join

ticketbooking\_feb\_hex\_24.booking b on e.id = b.event\_id

group by

e.event\_type, e.venue\_id;

14. write a sql query to list users and the total number of tickets they've purchased in the last 30 days.

select

customer\_id,

sum(num\_tickets) as total\_tickets\_purchased

from

ticketbooking\_feb\_hex\_24.booking

where

booking\_date >= date\_sub(curdate(), interval 30 day)

group by

customer\_id;

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

| customer\_id | total\_tickets\_purchased |

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

| 1 | 6 |

| 2 | 8 |

| 3 | 10 |

| 5 | 20 |

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

tasks 4: subquery and its types

1. calculate the average ticket price for events in each venue using a subquery.

select v.venue\_name, (select avg(ticket\_price)

from event where venue\_id = v.id) as avg\_ticket\_price

from venue v;

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

| venue\_name | avg\_ticket\_price |

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

| mumbai | 8000 |

| chennai | 3500 |

| pondicherry | 600 |

| mumbai | NULL |

| chennai | NULL |

| pondicherry | NULL |

| mumbai | NULL |

| chennai | NULL |

| pondicherry | NULL |

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

1. find events with more than 50% of tickets sold using subquery.

select e.event\_name

from event e

where (select sum(num\_tickets)

from booking

where event\_id = e.id) > (e.total\_seats \* 0.5);

1. calculate the total number of tickets sold for each event.

select e.event\_name,

coalesce(sum(b.num\_tickets), 0) as total\_tickets\_sold

from event e

left join booking b on e.id = b.event\_id

group by e.id;

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

| event\_name | total\_tickets\_sold |

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

| Late Ms. Lata Mangeshkar Musical | 5 |

| CSK vs RCB | 8 |

| CSK vs RR | 10 |

| Conferece CUP | 9 |

1. find users who have not booked any tickets using a not exists subquery.

select c.customer\_name

from customer c

where not exists (select 1

from booking b

where b.customer\_id = c.id);

-> WHERE b.customer\_id = c.id);

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

| customer\_name |

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

| harry potter |

| ronald weasley |

| hermione granger |

| draco malfoy |

| ginni weasley |

| harry potter |

| ronald weasley |

| hermione granger |

| draco malfoy |

| ginni weasley |

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

1. list events with no ticket sales using a not in subquery.

select event\_name

from event

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

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

| event\_name |

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

| MI vs KKR |

| Late Ms. Lata Mangeshkar Musical |

| CSK vs RCB |

| CSK vs RR |

| MI vs KKR |

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

1. calculate the total number of tickets sold for each event type using a subquery in the from clause.

select e.event\_type,

coalesce(sum(t.total\_tickets), 0) as total\_tickets\_sold

from event e

left join (select event\_id,

sum(num\_tickets) as total\_tickets

from booking

group by event\_id) t on e.id = t.event\_id

group by e.event\_type;

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

| event\_type | total\_tickets\_sold |

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

| concert | 13 |

| sports | 41 |

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

1. 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 |

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

| Conferece CUP | 8000 |

| MI vs KKR | 8000 |

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

1. calculate the total revenue generated by events for each user using a correlated subquery.

select c.customer\_name

from customer c

where exists (select 1

from booking b

join event e on b.event\_id = e.id

where e.venue\_id = <venue\_id> and b.customer\_id = c.id);

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

| customer\_name | total\_revenue |

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

| harry potter | 22880 |

| ronald weasley | 64000 |

| hermione granger | 36000 |

| draco malfoy | 1920 |

| ginni weasley | 68000 |

1. list users who have booked tickets for events in a given venue using a subquery in the where clause.

SELECT c.customer\_name

FROM customer c

WHERE EXISTS (SELECT 1

FROM booking b

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

WHERE e.venue\_id = 1 AND b.customer\_id = 2);

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

| customer\_name |

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

| harry potter |

| ronald weasley |

| hermione granger |

| draco malfoy |

| ginni weasley |

10. calculate the total number of tickets sold for each event category using a subquery with group by.

select e.event\_type,

coalesce(sum(b.num\_tickets), 0) as total\_tickets\_sold

from event e

left join booking b on e.id = b.event\_id

group by e.event\_type;

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

| event\_type | total\_tickets\_sold |

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

| concert | 13 |

| sports | 41 |

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

11. find users who have booked tickets for events in each month using a subquery with date\_format.

select c.customer\_name,

date\_format(b.booking\_date, '%y-%m') as booking\_month

from customer c

join booking b on c.id = b.customer\_id

group by c.id, booking\_month;

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

| customer\_name | booking\_month |

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

| harry potter | 2021-09 |

| harry potter | 2024-04 |

| ronald weasley | 2024-05 |

| hermione granger | 2024-04 |

| draco malfoy | 2021-09 |

| ginni weasley | 2024-04 |

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

12. calculate the average ticket price for events in each venue using a subquery

select v.venue\_name,

avg(e.ticket\_price) as avg\_ticket\_price

from venue v

join event e on v.id = e.venue\_id

group by v.id;

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

| venue\_name | avg\_ticket\_price |

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

| mumbai | 8000 |

| chennai | 3500 |

| pondicherry | 600 |

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