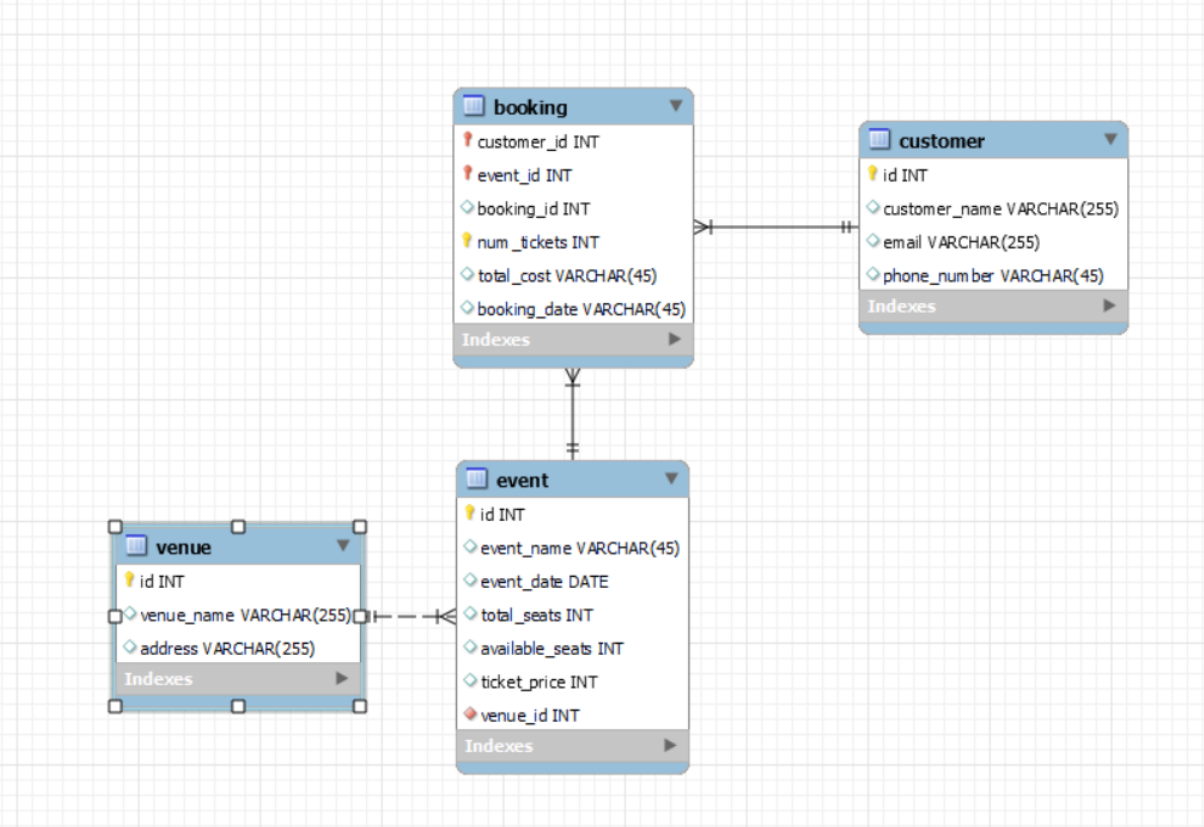
TICKET BOOKING – ASSIGNMENT 1




```
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 |
+--------------+------------------+
```

---

2. 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);
```

3. 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 |
```

4. 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    |
+------------------+
```

5. 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                    |
+----------------------------------+
```

6. 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 |
+------------+--------------------+
```

7. 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 |
+------------------+---------------+
```

8. 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

9. 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 |
+--------------+------------------+
```