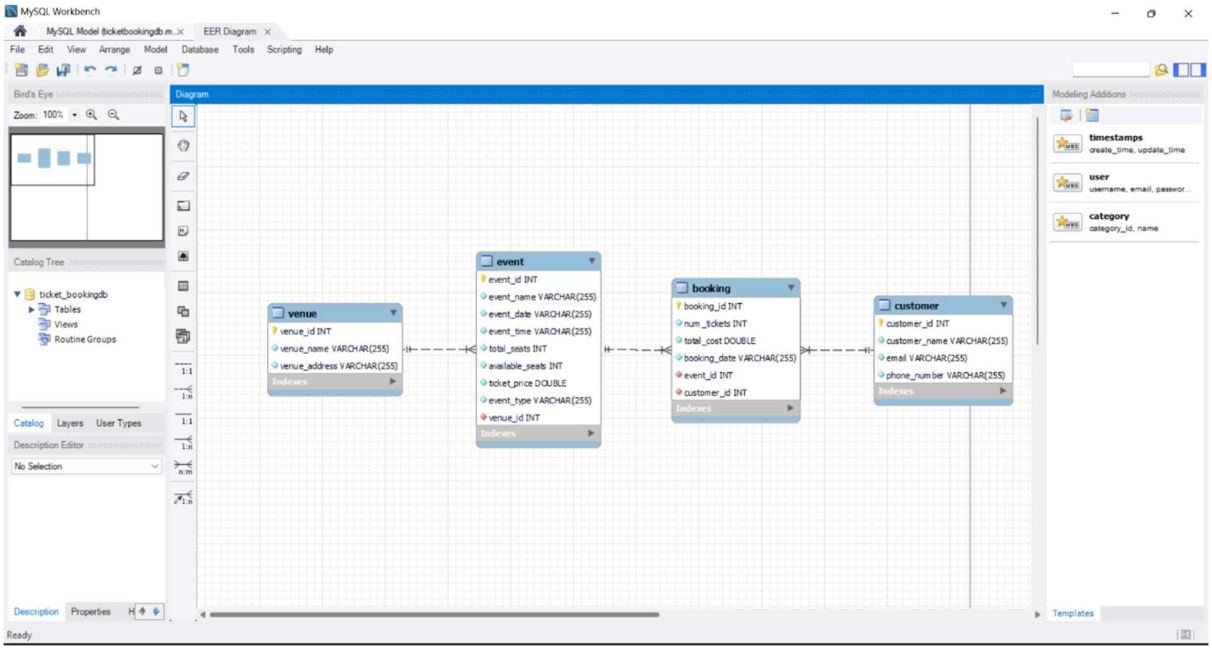
Assignment-1

name	Yalini Shree P V
serial no.	31
topic	ticket booking system

ER diagram for reference



sql code with outputs

```
-- mysql workbench forward engineering

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

-- schema ticket_bookingdb

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

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

-- schema ticket_bookingdb

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

create schema if not exists `ticket_bookingdb` default character set utf8 ;

use `ticket_bookingdb` ;

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

-- table `ticket_bookingdb`.`venue`
```

```
create table if not exists `ticket_bookingdb`.`venue` (
  `venue_id` int not null auto_increment,
  `venue_name` varchar(255) not null,
  `venue_address` varchar(255) not null,
  primary key (`venue_id`))
engine = innodb;
-- -----------------------------------------------------
-- table `ticket_bookingdb`.`event`
-- -----------------------------------------------------
create table if not exists `ticket_bookingdb`.`event` (
  `event_id` int not null auto_increment,
  `event_name` varchar(255) not null,
  `event_date` varchar(255) not null,
  `event_time` varchar(255) not null,
  `total_seats` int not null,
  `available_seats` int not null,
  `ticket_price` double not null,
  `event_type` varchar(255) not null,
  `venue_id` int not null,
  primary key (`event_id`),
  index `fk_event_venue_idx` (`venue_id` asc) ,
  constraint `fk_event_venue`
    foreign key (`venue_id`)
    references `ticket_bookingdb`.`venue` (`venue_id`)
    on delete no action
    on update no action)
engine = innodb;
-- -----------------------------------------------------
-- table `ticket_bookingdb`.`customer`
-- -----------------------------------------------------
create table if not exists `ticket_bookingdb`.`customer` (
```

```
  `customer_id` int not null auto_increment,
  `customer_name` varchar(255) not null,
  `email` varchar(255) not null,
  `phone_number` varchar(255) not null,
  primary key (`customer_id`))
engine = innodb;
-- -----------------------------------------------------
-- table `ticket_bookingdb`.`booking`
-- -----------------------------------------------------
create table if not exists `ticket_bookingdb`.`booking` (
  `booking_id` int not null auto_increment,
  `num_tickets` int not null,
  `total_cost` double not null,
  `booking_date` varchar(255) not null,
  `event_id` int not null,
  `customer_id` int not null,
  primary key (`booking_id`),
  index `fk_booking_event1_idx` (`event_id` asc) ,
  index `fk_booking_customer1_idx` (`customer_id` asc) ,
  constraint `fk_booking_event1`
    foreign key (`event_id`)
    references `ticket_bookingdb`.`event` (`event_id`)
    on delete no action
    on update no action,
  constraint `fk_booking_customer1`
    foreign key (`customer_id`)
    references `ticket_bookingdb`.`customer` (`customer_id`)
    on delete no action
    on update no action)
engine = innodb;
```

tasks 2: select, where, between, and, like:

-- 1. write a sql query to insert at least 10 sample records into each table.

```
use ticket_bookingdb;

insert into venue(venue_name,venue_address) values
('mumbai', 'marol andheri(w)'),
('chennai', 'it park'),
('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');

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);

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');

2. write a sql query to list all events.

select * from event;

31	late ms. lata mangeshkar musical concert 26	2021-09-12	20:00	320	270	600		
32	csk vs rcb	2024-04-11	19:30	23000	3	3600	sports	24
33	csk vs rr	2024-04-19	19:30	23000	10	3400	sports	24
34	mi vs kkr	2024-05-01	15:30	28000	100	8000	sports	25

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

select event_name

from event

where available_seats > 0;

late ms. lata mangeshkar musical

csk vs rcb

csk vs rr

mi vs kkr

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

select event_name

from event

where event_name like '%cup%';

nill

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

select event_name

from event

where ticket_price between 1000 and 2500;

nill

6. write a sql query to retrieve events with dates falling within a specific range.

select *

from event

where event_date between '2024-09-01' and '2024-12-31';

nill

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

select *

from event

where available_seats > 0 and event_name like '%concert%';

nill

8. write a sql query to retrieve users in batches of 5, starting from the 6th user.

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

select *

from booking

where num_tickets > 4;

66 5 18000 2024-04-10 32 29

67 10 34000 2024-04-15 33 28

10. write a sql query to retrieve customer information whose phone number end with '000'

select *

from customer

where phone_number like '%000';

nill

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 desc;

nill

12. write a sql query to select events name not start with x', y. 't

select *

from event

where event_name not like 'x%'

and event_name not like 'y%'

and event_name not like 't%';

31 late ms. lata mangeshkar musical concert 26 2021-09-12 20:00 320 270 600

32 csk vs rcb 2024-04-11 19:30 23000 3 3600 sports 24

33 csk vs rr2024-04-19 19:30 23000 10 3400 sports 24

34 mi vs kkr 2024-05-01 15:30 28000 100 8000 sports 25

tasks 3: aggregate functions, having, order by, groupby and joins:

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

select event_name, avg(ticket_price) as avg_ticket_price

from event

group by event_name;

csk vs rcb 3600

csk vs rr3400

late ms. lata mangeshkar musical 600

mi vs kkr 8000

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

select sum(total_cost) as total_revenue

from booking;

96400

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

select event_name, sum(num_tickets) as total_tickets_sold

from booking

join event on booking.event_id = event.event_id

group by event_name

order by total_tickets_sold desc

limit 1;

csk vs rr15

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

```
select event_name, sum(num_tickets) as total_tickets_sold
from booking
join event on booking.event_id = event.event_id
group by event_name;
```

csk vs rcb 8

csk vs rr10

late ms. lata mangeshkar musical 5

mi vs kkr 4

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

```
select event_name
from event
left join booking on event.event_id = booking.event_id
where booking.booking_id is null;
```

nill

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

```
select customer_name, sum(num_tickets) as total_tickets_booked
from booking
join customer on booking.customer_id = customer.customer_id
group by customer_name
order by total_tickets_booked desc
limit 1;
```

hermione granger 10

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

```
select month(booking_date) as month,sum(num_tickets)
from booking
join event on booking.event_id = event.event_id
```

group by month;

4 18

5 4

9 5

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

select venue_name, avg(ticket_price) as avg_ticket_price

from event

join venue on event.venue_id = venue.venue_id

group by venue_name;

chennai 8000

mumbai 3500

pondicherry 600

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

select event_type, sum(num_tickets) as total_tickets_sold

from booking

join event on booking.event_id = event.event_id

group by event_type;

concert 5

sports 22

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 booking

group by year

order by year;

2021 1600

2024 94800

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

select customer_name, count(distinct event_id) as num_events_booked

from booking

```
join customer on booking.customer_id = customer.customer_id
group by customer_name
having num_events_booked > 1;
harry potter	2
```

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

```
select customer_name, sum(total_cost) as total_revenue
from booking
join customer on booking.customer_id = customer.customer_id
group by customer_name;
draco malfoy	18000
ginni weasley	960
harry potter	11440
hermione granger	34000
ronald weasley	32000
```

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

```
select venue_name, event_type, avg(ticket_price) as avg_ticket_price
from event
join venue on event.venue_id = venue.venue_id
group by venue_name, event_type;
chennaisports	8000
mumbai	sports	3500
pondicherry	concert	600
```

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

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

2.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;
```

7,**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
```

8.**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;
```