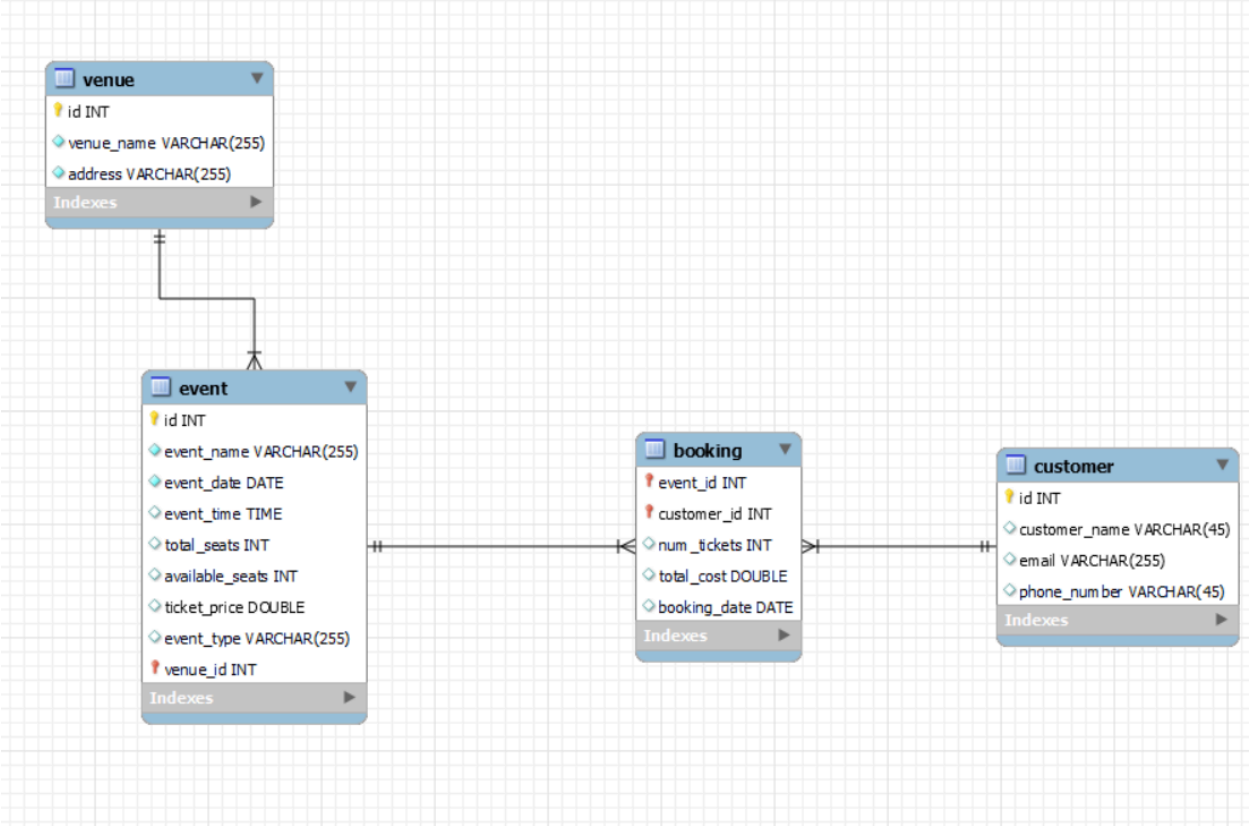
Ticket Booking case study

ER DIAGRAM :



#db scripts

-- MySQL Workbench Forward Engineering

-- ---
-- Schema ticketbooking_feb_hex_24
-- ---

-- ---
-- Schema ticketbooking_feb_hex_24
-- ---
CREATE SCHEMA IF NOT EXISTS `ticketbooking_feb_hex_24` DEFAULT CHARACTER SET utf8 ;
USE `ticketbooking_feb_hex_24` ;

```
-- -----------------------------------------------------
-- Table `ticketbooking_feb_hex_24`.`venue`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ticketbooking_feb_hex_24`.`venue` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `venue_name` VARCHAR(255) NOT NULL,
  `address` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `ticketbooking_feb_hex_24`.`event`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ticketbooking_feb_hex_24`.`event` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `event_name` VARCHAR(255) NOT NULL,
  `event_date` DATE NOT NULL,
  `event_time` TIME NULL,
  `total_seats` INT NULL,
  `available_seats` INT NULL,
  `ticket_price` DOUBLE NULL,
  `event_type` VARCHAR(255) NULL,
  `venue_id` INT NOT NULL,
  PRIMARY KEY (`id`, `venue_id`),
  INDEX `fk_event_venue1_idx` (`venue_id` ASC) ,
  CONSTRAINT `fk_event_venue1`
    FOREIGN KEY (`venue_id`)
    REFERENCES `ticketbooking_feb_hex_24`.`venue` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `ticketbooking_feb_hex_24`.`customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ticketbooking_feb_hex_24`.`customer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `customer_name` VARCHAR(45) NULL,
  `email` VARCHAR(255) NULL,
  `phone_number` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

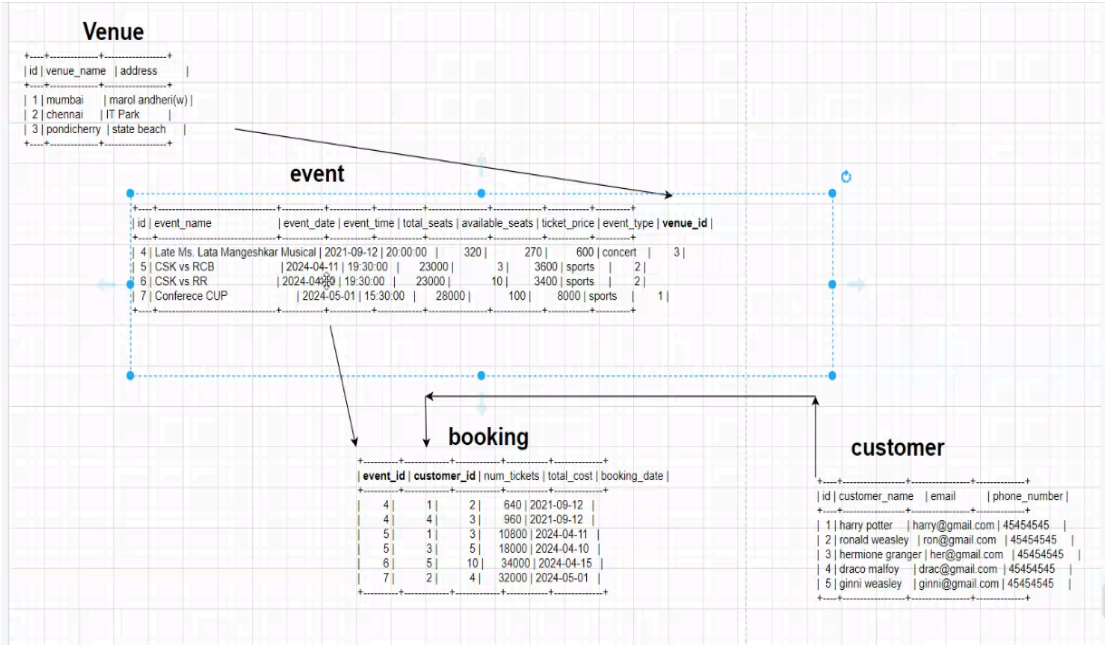
```
-- -----------------------------------------------------
-- Table `ticketbooking_feb_hex_24`.`booking`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ticketbooking_feb_hex_24`.`booking` (
  `event_id` INT NOT NULL,
  `customer_id` INT NOT NULL,
  `num_tickets` INT NULL,
  `total_cost` DOUBLE NULL,
  `booking_date` DATE NULL,
  INDEX `fk_event_has_customer_customer1_idx` (`customer_id` ASC) ,
  INDEX `fk_event_has_customer_event_idx` (`event_id` ASC) ,
  PRIMARY KEY (`event_id`, `customer_id`),
  CONSTRAINT `fk_event_has_customer_event`
    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;
```

Query Insertions

```
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;

insert into customer(customer_name , email,phone_number)
values
('harry potter','harry@gmail.com','45454545000'),
('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;

insert into
event(event_name,event_date,event_time,total_seats,available_seats,ticket_price,event_type,v
enue_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
event(event_name,event_date,event_time,total_seats,available_seats,ticket_price,event_type,v
enue_id)
values ('CSK vs MI', '2024-06-01','18:30',29000,90,8000,'sports',2);
select * from event;

insert into booking 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');
```

Reference Image:



TASK 2:

#SQL Queries - Task 2
#insertions - done
-- 2. Write a SQL query to list all Events.

```
select * from event;
/*
+----------+---------------------------------+------------+------------+-------------+-----------------+--------------+-
-----------+----------+
| 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 | MI vs KKR                       | 2024-05-01 | 15:30:00   |       28000 |             100 |         8000
| sports     |        1 |
|        5 | CSK vs MI                       | 2024-06-01 | 18:30:00   |       29000 |              90 |         8000
| sports     |        2 |
+----------+---------------------------------+------------+------------+-------------+-----------------+--------------+-
-----------+----------+
*/
```

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

```
select * from event where available_seats>0;
/*
+----------+---------------------------------+------------+------------+-------------+-----------------+--------------+-
-----------+----------+
| 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 | MI vs KKR                       | 2024-05-01 | 15:30:00   |       28000 |             100 |         8000
| sports     |        1 |
```

```
|       5 | Conference Cup                  | 2024-06-01 | 18:30:00   |       29000 |             90 |
8000 | sports     |        2 |
+---------+---------------------------------+------------+------------+-------------+----------------+-------------+-
----------+----------+
*/
update event set event_name='Conference Cup' where id= 5;
-- 4. Write a SQL query to select events name partial match with ‘cup’.
select * from event where event_name LIKE '%cup%';

/*
+---------+---------------+------------+------------+-------------+----------------+-------------+------------+------
----+
| event_id | event_name     | event_date | event_time | total_seats | available_seats | ticket_price
| event_type | venue_id |
+---------+---------------+------------+------------+-------------+----------------+-------------+------------+------
----+
|       5 | Conference Cup | 2024-06-01 | 18:30:00   |       29000 |             90 |         8000 | sports
|       2 |
+---------+---------------+------------+------------+-------------+----------------+-------------+------------+------
----+
*/
-- 5. Write a SQL query to select events with ticket price range is between 1000 to 2500.
select * from event where 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';
/*
+---------+------------+------------+------------+-------------+----------------+-------------+------------+----------
+
| event_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 |
|       4 | MI vs KKR  | 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 * from event where available_seats >0 AND event_type='concert';
/*
+----------+---------------------------------+------------+------------+-------------+-----------------+--------------+-----------+----------+
| 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 |
+----------+---------------------------------+------------+------------+-------------+-----------------+--------------+-----------+----------+
*/
```

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

```
select *
from customer
limit 3,2;

select *
from customer
limit 5,5; #records 6-10
```

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

```
select * from booking where num_tickets >4;
```

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

```
select *
from customer
where phone_number LIKE '%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 ;
```

-- 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%' and event_name NOT
LIKE 'y%';
```

#Level 2:Multi table Queries using Manual Technique

```
-- display list of events hosted by venue 'chennai'
select e.id, e.event_name,e.event_date, e.event_time,e.total_seats
from event e, venue v
where v.id= e.venue_id and v.venue_name='chennai';
```

```
-- select customers that have booked tickets for event 'csk vs rcb ' game with id =5

select c.customer_name,c.email,c.phone_number
from customer c , booking b
where c.id =b.customer_id AND b.customer_id=5;
```

```
-- display event details that have booking total_cost >1000
select e.id, e.event_name,e.event_date, e.event_time,e.total_seats,e.ticket_price,e.event_type
from event e , booking b
where e.id = b.event_id AND b.total_cost > 1000;
```

```
/*
        Display the names of venues visited by customer with email 'harry@gmail.com'
*/
select v.venue_name,v.address, c.customer_name
from venue v , booking b , event e, customer c
where v.id=e.venue_id AND
e.id=b.event_id AND
b.customer_id = c.id AND
c.email= 'harry@gmail.com';
```

TASK 3:

```
-- Task  3: Aggregate functions, Having, Order By, GroupBy and Joins:

-- 1. Write a SQL query to List  venues and Their Average Ticket Prices.
-- 8. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.

select e.venue_id ,v.venue_name, avg(e.ticket_price)
from venue v, event e
where v.id=e.venue_id
group by e.venue_id;
```

#note: We can join multiple tables like venue and fetch extra info from there like venue_name.

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

```
select sum((total_seats-available_seats)*ticket_price) #We can perform arithmetic ops in select statement
from event;
```

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

```
select event_name ,max((total_seats-available_seats)*ticket_price) as total_sales # why we are using max means if we have two same names we take max of it
from event
group by event_name
order by total_sales Desc
limit 0 , 1;
```

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

```
select event_name ,(total_seats - available_seats) as total_tickets
from event
group by event_name;
```

-- 5. Write a SQL query to Find Events with No Ticket Sales.

```
select event_name
from event
where available_seats = total_seats;
```

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

#plan : find the tickers booked by the customer . the find the most

```
select customer_name , sum(b.num_tickets) as tickets_booked
from booking b , customer c
where b.customer_id = c.id
group by customer_name
order by tickets_booked desc
limit 0 ,1;
```

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

```
select e.event_name , sum(b.num_tickets) as total_tickets  , Month(b.booking_date) as Month
from booking b , event e
where e.id=b.event_id
group by  MONTH(b.booking_date);
```

-- 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_sales
from booking b, event e
where e.id =b.event_id
group by event_type;
```

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

```
select sum(b.total_cost) as total_revenue ,year(b.booking_date) as booking_year
from event e join booking b on e.id = b.event_id
group by year(b.booking_date);
```

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

```
-- one way
select  c.customer_name ,count(customer_id ) as event_booked
from booking b, customer c
where c.id=b.customer_id
group by customer_id
Order by event_booked desc
limit 0,1;

-- another way of using having

select c.customer_name , count(c.id) as events_booked
from event e,customer c, booking b
where e.id = b.event_id AND
b.customer_id = c.id
group by c.customer_name
having events_booked>1;

/*
 | harry potter     |             2 |
*/
```

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

```
 select c.customer_name , sum(b.total_cost) as total_revenue
from event e , booking b , customer c
where e.id=b.event_id AND
b.customer_id = c.id
group by c.customer_name
order by total_revenue DESC;
```

-- 13. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.

```
select   e.event_type ,avg((total_seats-available_seats)*ticket_price) as Average_price
```

```
from event e , venue v
where v.id =e.venue_id
group by e.event_type ;-- each category

select e.event_type , v.venue_name ,avg((total_seats-available_seats)*ticket_price) as
Average_price
from event e , venue v
where v.id =e.venue_id
group by  e.event_type,v.venue_name ;-- each category and  venue

-- 14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased
in the Last 30 Days
select c.customer_name ,sum(b.num_tickets ) as total_tickets
from customer c join booking b
on b.customer_id = c.id
where b.booking_date BETWEEN DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY) AND
'2024-01-01'
group by customer_name;

-- now() gives today's date with time
-- we can use also now () .In current_date() gives only date
```

TASK 4:

```
-- Task 4: Subquery and its types
1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery
select id ,avg(ticket_price) as Average_price
from event where venue_id in ( select id from venue)
group by venue_id;

2. Find Events with More Than 50% of Tickets Sold using subquery.
select event_name
from event
where id in ( select id from event
where (total_seats - available_seats) > (total_seats/2));

3. Find Events having ticket price more than average ticket price of all events
select event_name
from event where  ticket_price > (select avg(ticket_price) from event );

 4. Find Customers Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.
 #inserting values to check the query
  insert into customer(customer_name,email,phone_number)
```

```
values ('severus snape', 'sev@gmail.com','56556');
select * from customer;
```

 # here i am displaying both exists and not exists for understanding

```
select customer_name
from customer where not exists(select  distinct c.customer_name
from customer c join booking b on c.id = b.customer_id );-- false means prints null

select customer_name
from customer where exists(select  distinct c.customer_name
from customer c join booking b on c.id = b.customer_id ); -- true means prints all the value
```

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

```
select id , event_name
from event where id not in (select id  from event where (total_seats - available_seats) > 0 );
```

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

```
select event_type , total_tickets from
(select event_type ,(total_seats- available_seats) AS Total_tickets
from event
group by event_type) as t;
```

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

```
select id , event_name , ticket_price
from event where ticket_price > (select avg(ticket_price ) from event);
```

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

-- A correlated query is a specific type of subquery that is evaluated repeatedly, once for each row processed by the outer query.

-- The inner query is executed once for each row processed by the outer query.

-- no need to use group by because we are processing with each id

```
select c.id , c.customer_name ,
 (select sum(b.total_cost)
from booking b join event e on e.id = b.event_id where c.id = b.customer_id) as total_revenue
from customer c;
```

-- 9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.

```
select customer_name , id
from customer where id in  (select customer_id from booking where event_id in (select id from
event where venue_id =1 ));
```

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

```
select event_type , sum(Total_tickets)
```

```
from (
select event_type ,(total_seats - available_seats ) as Total_tickets  from event ) as sub
group by event_type ;
```

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

```
select customer_name , booking_month from(
select c.customer_name, month(b.booking_date) as booking_month
from booking b join customer c on c.id = b.customer_id) as subquery
group by booking_month;
```

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

```
select id ,avg(ticket_price) as Average_price
from event where venue_id in ( select id from venue)
group by venue_id;
```