ASSIGNMENT - 5

This assignment task is to create a ticket booking system for an Event. The system should support booking tickets for different types of events, such as movies, concerts, and plays. Each event has its own pricing strategy, and the system should also track available seats and customer bookings

TASK -1 DATABASE DESIGN

1) Create a database named "TicketBookingSystem".

```
mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.03 sec)
```

2) Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

Venu

```
mysql> CREATE TABLE VENU(
    ->  venue_id INT PRIMARY KEY ,
    -> venue_name VARCHAR(255),
    ->  address TEXT
    -> );
```

Event

```
mysql> CREATE TABLE Event(
    -> event_id INT PRIMARY KEY,
    -> event_name VARCHAR(50),
    -> event_time TIME,
    ->  venue_id INT,
    ->  total_seats INT,
    ->  available_seats INT ,
    ->  ticket_price DECIMAL(4,2),
    ->  event_type ENUM('Movie','Sports','Concert'),
    ->  booking_id INT,
    ->  FOREIGN KEY(venue_id) REFERENCES VENU(venue_id)
    -> );
```
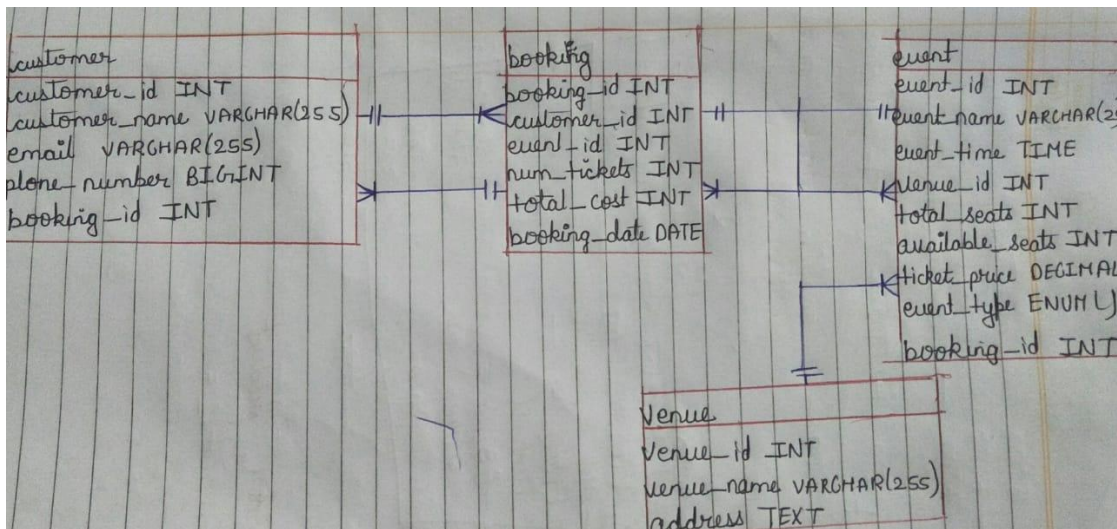
Customers

```
mysql> CREATE TABLE Customer(
    -> customer_id INT PRIMARY KEY,
    -> customer_name VARCHAR(255),
    -> email VARCHAR(255),
    -> phone_number BIGINT,
    -> booking_id INT
    -> );
```

Booking

```
mysql> CREATE TABLE BOOKING(
    -> booking_id INT PRIMARY KEY,
    -> customer_id INT,
    -> event_id INT,
    -> num_tickets INT,
    -> total_cost INT,
    -> booking_date DATE,
    ->  FOREIGN KEY(customer_id)  REFERENCES Customer(customer_id),
    -> FOREIGN KEY(event_id) REFERENCES Event(event_id)
    -> );
```

3) Create an ERD (Entity Relationship Diagram) for the database.

4) Create appropriate Primary Key and Foreign Key constraints for referential integrity.

Primary Key

TABLE_NAME	CONSTRAINT_NAME
customer	PRIMARY
venu	PRIMARY
event	PRIMARY

Foreign Key

```
------------+-------------------+
TABLE_NAME  | CONSTRAINT_NAME   |
------------+-------------------+
customer    | fk_customer_address |
customer    | fk_customer_store   |
event       | event_ibfk_1        |
event       | event_ibfk_2        |
customer    | customer_ibfk_1     |
------------+-------------------+
```

TASKS 2: SELECT, WHERE, BETWEEN, AND, LIKE

1) Write a SQL query to insert at least 10 sample records into each table.

Venu

venue_id	venue_name	address
101	Rajasthan Convention Center	1 Jaipur Road, Delhi
102	Taj Mahal Auditorium	22 Agra Street, Mumbai
103	Bollywood Studios	33 Film City, Mumbai
104	Ganges Stadium	44 Varanasi Avenue, Kolkata
105	Lotus Palace Theater	55 Bangalore Circle, Bangalore
106	Chennai Music Hall	66 Carnatic Street, Chennai
107	Punjab Sports Arena	77 Amritsar Road, Chandigarh
108	Kerala Cultural Center	88 Cochin Lane, Kochi
109	Hyderabad Exhibition Center	99 Hitech City, Hyderabad
110	Gujarat Pavilion	100 Ahmedabad Street, Gandhinagar

Event

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
201	Movie Night	18:00:00	101	200	200	10.00	Movie	NULL	2023-12-20
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
203	Sports Championship	15:00:00	103	1000	800	20.00	Sports	NULL	2024-02-05
204	Drama Play	19:30:00	104	150	100	15.00	Movie	NULL	2024-03-10
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
207	Dance Performance	17:30:00	107	400	350	18.00	Movie	NULL	2024-06-30
208	Art Exhibition	12:00:00	108	120	100	12.00	Sports	NULL	2024-07-12
209	Magic Show	16:00:00	109	180	150	8.00	Sports	NULL	2024-08-20
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

Customer

customer_id	customer_name	email	phone_number	booking_id
301	Aarav Patel	aarav@example.com	876543210	1
302	Aadhya Singh	aadhya@example.com	876532109	2
303	Advik Gupta	advik@example.com	7654321098	3
304	Ananya Reddy	ananya@example.com	6543210987	4
305	Ishaan Kumar	ishaan@example.com	9432109876	5
306	Aaradhya Joshi	aaradhya@example.com	4321098765	6
307	Vihaan Khanna	vihaan@example.com	9210987654	7
308	Kyra Sharma	kyra@example.com	92109876543	8
309	Reyansh Verma	reyansh@example.com	8098765432	9
310	Saisha Mishra	saisha@example.com	70987654321	10

Booking

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1	301	201	2	420	2023-12-18
2	302	202	3	150	2024-01-10
3	303	203	5	100	2024-02-01
4	304	204	1	215	2024-03-05
5	305	205	4	100	2024-04-12
6	306	206	2	660	2024-05-20
7	307	207	3	454	2024-06-25
8	308	208	2	124	2024-07-05
9	309	209	4	932	2024-08-15
10	310	210	6	132	2024-09-22

2) Write a SQL query to list all Events.

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
201	Movie Night	18:00:00	101	200	200	10.00	Movie	NULL	2023-12-20
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
203	Sports Championship	15:00:00	103	1000	800	20.00	Sports	NULL	2024-02-05
204	Drama Play	19:30:00	104	150	100	15.00	Movie	NULL	2024-03-10
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
207	Dance Performance	17:30:00	107	400	350	18.00	Movie	NULL	2024-06-30
208	Art Exhibition	12:00:00	108	120	100	12.00	Sports	NULL	2024-07-12
209	Magic Show	16:00:00	109	180	150	8.00	Sports	NULL	2024-08-20
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

3) Write a SQL query to select events with available tickets

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
201	Movie Night	18:00:00	101	200	200	10.00	Movie	NULL	2023-12-20
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
203	Sports Championship	15:00:00	103	1000	800	20.00	Sports	NULL	2024-02-05
204	Drama Play	19:30:00	104	150	100	15.00	Movie	NULL	2024-03-10
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
207	Dance Performance	17:30:00	107	400	350	18.00	Movie	NULL	2024-06-30
208	Art Exhibition	12:00:00	108	120	100	12.00	Sports	NULL	2024-07-12
209	Magic Show	16:00:00	109	180	150	8.00	Sports	NULL	2024-08-20
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_name LIKE '%cup%';
Empty set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE ticket_price BETWEEN 1000 AND 2500;
Empty set (0.00 sec)
```

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

event_id	event_name	event_time	venue_id	ticket_price	event_type
201	Movie Night	18:00:00	101	10.00	Movie

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

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15

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

customer_id	customer_name	email	phone_number	booking_id
306	Aaradhya Joshi	aaradhya@example.com	4321098765	6
307	Vihaan Khanna	vihaan@example.com	9210987654	7
308	Kyra Sharma	kyra@example.com	92109876543	8
309	Reyansh Verma	reyansh@example.com	8098765432	9
310	Saisha Mishra	saisha@example.com	70987654321	10

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

```
------------+-------------+----------+-------------+------------+-------------
 booking_id | customer_id | event_id | num_tickets | total_cost | booking_date
------------+-------------+----------+-------------+------------+-------------
          3 |         303 |      203 |           5 |        100 | 2024-02-01
         10 |         310 |      210 |           6 |        132 | 2024-09-22
------------+-------------+----------+-------------+------------+-------------
```

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

```
ysql> use ticketbookingsystem;
atabase changed
ysql> SELECT *
    -> FROM Customer
    -> WHERE phone_number LIKE '%000';
pty set (0.00 sec)
```

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

```
uery to retrieve the events in order
mysql> SELECT *
    -> FROM Event
    -> WHERE total_seats > 15000
    -> ORDER BY total_seats ASC;
Empty set (0.00 sec)

mysql>
```

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

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
201	Movie Night	18:00:00	101	200	200	10.00	Movie	NULL	2023-12-20
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
203	Sports Championship	15:00:00	103	1000	800	20.00	Sports	NULL	2024-02-05
204	Drama Play	19:30:00	104	150	100	15.00	Movie	NULL	2024-03-10
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
207	Dance Performance	17:30:00	107	400	350	18.00	Movie	NULL	2024-06-30
208	Art Exhibition	12:00:00	108	120	100	12.00	Sports	NULL	2024-07-12
209	Magic Show	16:00:00	109	180	150	8.00	Sports	NULL	2024-08-20
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

TASK 3: AGGREGATE FUNCTION, HAVING, ORDER BY, GROUPBY and JOINS

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

```
| event_id | event_name          | average_ticket_price |
+----------+---------------------+----------------------+
|      201 | Movie Night         |            10.000000 |
|      202 | Concert Extravaganza|            50.000000 |
|      203 | Sports Championship |            20.000000 |
|      204 | Drama Play          |            15.000000 |
|      205 | Live Comedy Show    |            25.000000 |
|      206 | Musical Night       |            30.000000 |
|      207 | Dance Performance   |            18.000000 |
|      208 | Art Exhibition      |            12.000000 |
|      209 | Magic Show          |             8.000000 |
|      210 | Cultural Festival   |            22.000000 |
```

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

```
+---------------+
| total_revenue |
+---------------+
|        687.00 |
+---------------+
```

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

```
+----------+-------------------+-------------------+
| event_id | event_name        | total_tickets_sold |
+----------+-------------------+-------------------+
|      210 | Cultural Festival |                 6 |
+----------+-------------------+-------------------+
```

4) Write a SQL query to Find Events with No Ticket Sales.

```
mysql> SELECT E.event_id, E.event_name
    -> FROM Event E
    -> LEFT JOIN Booking B ON E.event_id = B.event_id
    -> WHERE B.booking_id IS NULL;
Empty set (0.00 sec)
```

5) Write a SQL query to Find the User Who Has Booked the Most Tickets.

```
+-------------+----------------------+
| customer_id | total_tickets_booked |
+-------------+----------------------+
|         310 |                    6 |
+-------------+----------------------+
```

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

```
+----------+-------------------+
| event_id | total_tickets_sold |
+----------+-------------------+
|      201 |                 2 |
|      202 |                 3 |
|      203 |                 5 |
|      204 |                 1 |
|      205 |                 4 |
|      206 |                 2 |
|      207 |                 3 |
|      208 |                 2 |
|      209 |                 4 |
|      210 |                 6 |
+----------+-------------------+
```

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

```
+----------+---------------------+-------+-------------------+
| event_id | event_name          | month | total_tickets_sold |
+----------+---------------------+-------+-------------------+
|      202 | Concert Extravaganza |     1 |                 3 |
|      203 | Sports Championship  |     2 |                 5 |
|      204 | Drama Play           |     3 |                 1 |
|      205 | Live Comedy Show     |     4 |                 4 |
|      206 | Musical Night        |     5 |                 2 |
|      207 | Dance Performance    |     6 |                 3 |
|      208 | Art Exhibition       |     7 |                 2 |
|      209 | Magic Show           |     8 |                 4 |
|      210 | Cultural Festival    |     9 |                 6 |
|      201 | Movie Night          |    12 |                 2 |
+----------+---------------------+-------+-------------------+
```

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

```
+----------+-----------------------------+-----------------------+
| venue_id | venue_name                  | average_ticket_price |
+----------+-----------------------------+-----------------------+
|      101 | Rajasthan Convention Center |            10.000000 |
|      102 | Taj Mahal Auditorium        |            50.000000 |
|      103 | Bollywood Studios           |            20.000000 |
|      104 | Ganges Stadium              |            15.000000 |
|      105 | Lotus Palace Theater        |            25.000000 |
|      106 | Chennai Music Hall          |            30.000000 |
|      107 | Punjab Sports Arena         |            18.000000 |
|      108 | Kerala Cultural Center      |            12.000000 |
|      109 | Hyderabad Exhibition Center |             8.000000 |
|      110 | Gujarat Pavilion            |            22.000000 |
+----------+-----------------------------+-----------------------+
```

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

```
+------+---------------+
| year | total_revenue |
+------+---------------+
| 2023 |         20.00 |
| 2024 |        667.00 |
+------+---------------+
```

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

```
mysql> SELECT customer_id, COUNT(DISTINCT event_id) AS num_events_booked
    -> FROM Booking
    -> GROUP BY customer_id
    -> HAVING COUNT(DISTINCT event_id) > 1;
Empty set (0.00 sec)
```

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

```
+-------------+---------------+
| customer_id | total_revenue |
+-------------+---------------+
|         301 |         20.00 |
|         302 |        150.00 |
|         303 |        100.00 |
|         304 |         15.00 |
|         305 |        100.00 |
|         306 |         60.00 |
|         307 |         54.00 |
|         308 |         24.00 |
|         309 |         32.00 |
|         310 |        132.00 |
```

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

```
+------------+----------+----------------------+
| event_type | venue_id | average_ticket_price |
+------------+----------+----------------------+
| Movie      |      101 |            10.000000 |
| Concert    |      102 |            50.000000 |
| Sports     |      103 |            20.000000 |
| Movie      |      104 |            15.000000 |
| Concert    |      105 |            25.000000 |
| Concert    |      106 |            30.000000 |
| Movie      |      107 |            18.000000 |
| Sports     |      108 |            12.000000 |
| Sports     |      109 |             8.000000 |
| Concert    |      110 |            22.000000 |
+------------+----------+----------------------+
```

13) Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.

```
+-------------+-------------------------+
| customer_id | total_tickets_purchased |
+-------------+-------------------------+
|         301 |                       2 |
|         302 |                       3 |
|         303 |                       5 |
|         304 |                       1 |
|         305 |                       4 |
|         306 |                       2 |
|         307 |                       3 |
|         308 |                       2 |
|         309 |                       4 |
|         310 |                       6 |
+-------------+-------------------------+
```

TASKS 4: SUBQUERY and IT'S TYPES

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

```
+----------+----------------------------+--------------------------------+
| venue_id | venue_name                 | average_ticket_price_per_venue |
+----------+----------------------------+--------------------------------+
|      101 | Rajasthan Convention Center |                     10.000000 |
|      102 | Taj Mahal Auditorium       |                      50.000000 |
|      103 | Bollywood Studios          |                      20.000000 |
|      104 | Ganges Stadium             |                      15.000000 |
|      105 | Lotus Palace Theater       |                      25.000000 |
|      106 | Chennai Music Hall         |                      30.000000 |
|      107 | Punjab Sports Arena        |                      18.000000 |
|      108 | Kerala Cultural Center     |                      12.000000 |
|      109 | Hyderabad Exhibition Center |                      8.000000 |
|      110 | Gujarat Pavilion           |                      22.000000 |
+----------+----------------------------+--------------------------------+
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE (
    ->     SELECT SUM(num_tickets)
    ->     FROM Booking
    ->     WHERE Booking.event_id = Event.event_id
    -> ) > (0.5 * total_seats);
Empty set (0.00 sec)
```

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

```
+----------+-------------------+
| event_id | total_tickets_sold |
+----------+-------------------+
|      201 |                 2 |
|      202 |                 3 |
|      203 |                 5 |
|      204 |                 1 |
|      205 |                 4 |
|      206 |                 2 |
|      207 |                 3 |
|      208 |                 2 |
|      209 |                 4 |
|      210 |                 6 |
+----------+-------------------+
```

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

```
mysql> SELECT *
    -> FROM Customer C
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM Booking B
    ->     WHERE B.customer_id = C.customer_id
    -> );
Empty set (0.00 sec)
```

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

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_id NOT IN (
    ->     SELECT DISTINCT event_id
    ->     FROM Booking
    -> );
Empty set (0.00 sec)
```

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

```
+------------+-------------------+
| event_type | total_tickets_sold |
+------------+-------------------+
| Movie      |                 6 |
| Concert    |                15 |
| Sports     |                11 |
+------------+-------------------+
```

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

```
+----------+---------------------+------------+----------+-------------+-----------------+--------------+------------+------------+------------+
| event_id | event_name          | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id | event_date |
+----------+---------------------+------------+----------+-------------+-----------------+--------------+------------+------------+------------+
|      202 | Concert Extravaganza | 20:30:00   |      102 |         500 |             500 |        50.00 | Concert    |       NULL | 2024-01-15 |
|      205 | Live Comedy Show     | 21:00:00   |      105 |         300 |             300 |        25.00 | Concert    |       NULL | 2024-04-18 |
|      206 | Musical Night        | 19:00:00   |      106 |         250 |             200 |        30.00 | Concert    |       NULL | 2024-05-25 |
|      210 | Cultural Festival    | 14:00:00   |      110 |         600 |             550 |        22.00 | Concert    |       NULL | 2024-09-28 |
+----------+---------------------+------------+----------+-------------+-----------------+--------------+------------+------------+------------+
```

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

```
+-------------+---------------+
| customer_id | total_revenue |
+-------------+---------------+
|         301 |         20.00 |
|         302 |        150.00 |
|         303 |        100.00 |
|         304 |         15.00 |
|         305 |        100.00 |
|         306 |         60.00 |
|         307 |         54.00 |
|         308 |         24.00 |
|         309 |         32.00 |
|         310 |        132.00 |
+-------------+---------------+
```

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

```
+-------------+----------------+-------------------+--------------+------------+
| customer_id | customer_name  | email             | phone_number | booking_id |
+-------------+----------------+-------------------+--------------+------------+
|         306 | Aaradhya Joshi | aaradhya@example.com |  4321098765 |          6 |
+-------------+----------------+-------------------+--------------+------------+
```

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

```
| event_type | total_tickets_sold_per_category |
+------------+---------------------------------+
| Movie      |                               6 |
| Concert    |                              15 |
| Sports     |                              11 |
+------------+---------------------------------+
```

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

```
| customer_id | booked_month |
+-------------+--------------+
|         301 |           12 |
|         302 |            1 |
|         303 |            2 |
|         304 |            3 |
|         305 |            4 |
|         306 |            5 |
|         307 |            6 |
|         308 |            7 |
|         309 |            8 |
|         310 |            9 |
+-------------+--------------+
```

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

```
| venue_id | venue_name                  | average_ticket_price_per_venue |
+----------+-----------------------------+--------------------------------+
|      101 | Rajasthan Convention Center |                      10.000000 |
|      102 | Taj Mahal Auditorium        |                      50.000000 |
|      103 | Bollywood Studios           |                      20.000000 |
|      104 | Ganges Stadium              |                      15.000000 |
|      105 | Lotus Palace Theater        |                      25.000000 |
|      106 | Chennai Music Hall          |                      30.000000 |
|      107 | Punjab Sports Arena         |                      18.000000 |
|      108 | Kerala Cultural Center      |                      12.000000 |
|      109 | Hyderabad Exhibition Center |                       8.000000 |
|      110 | Gujarat Pavilion            |                      22.000000 |
+----------+-----------------------------+--------------------------------+
```