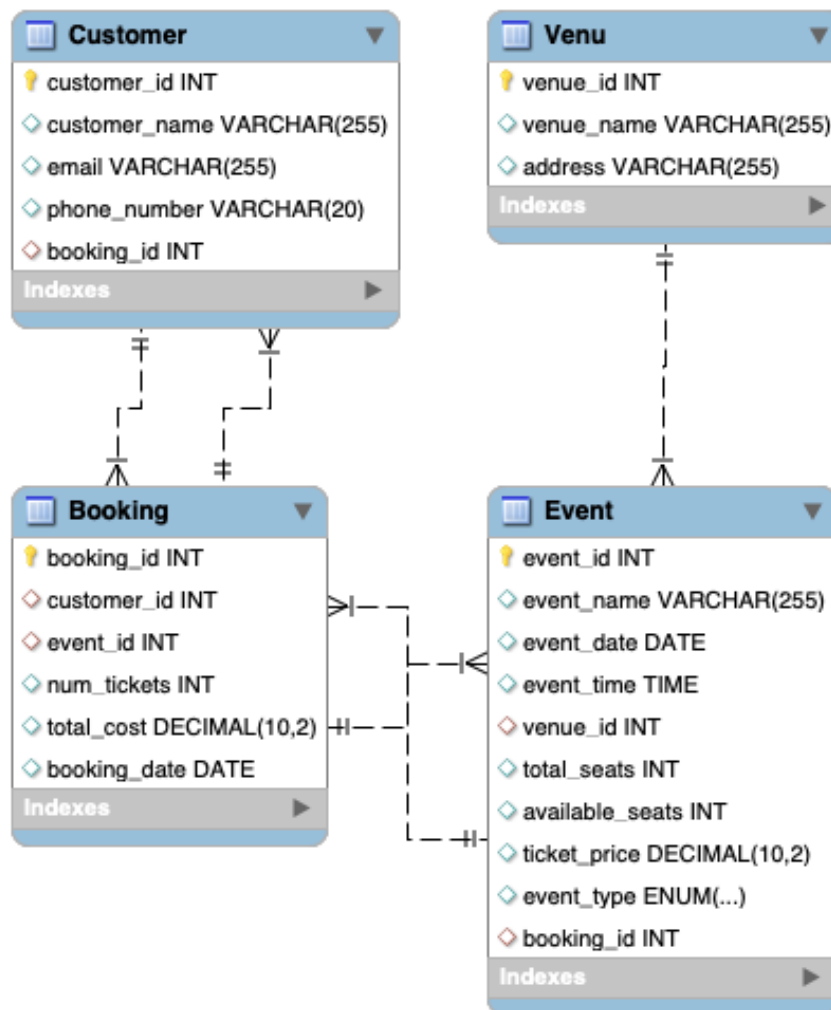


Assignment: 5

Ticket Booking System

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem"
2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
 - Venu
 - Event
 - Customers
 - Booking
3. Create an ERD (Entity Relationship Diagram) for the database.



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

```

1 • Create database if not exists TicketBookingSystem;
2 • use TicketBookingSystem;
3
4 • CREATE TABLE if not exists Venu (
5     venue_id INT PRIMARY KEY,
6     venue_name VARCHAR(255),
7     address VARCHAR(255)
8 );
9
10 • CREATE TABLE if not exists Event (
11     event_id INT PRIMARY KEY,
12     event_name VARCHAR(255),
13     event_date DATE,
14     event_time TIME,
15     venue_id INT,
16     total_seats INT,
17     available_seats INT,
18     ticket_price DECIMAL(10, 2),
19     event_type ENUM('Movie', 'Sports', 'Concert'),
20     booking_id INT,
21     FOREIGN KEY (venue_id) REFERENCES Venu(venue_id)
22 );
23 • ALTER TABLE Event
24 ADD CONSTRAINT fk_book_id
25 FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
26
27 • CREATE TABLE if not exists Customer (
28     customer_id INT PRIMARY KEY,
29     customer_name VARCHAR(255),
30     email VARCHAR(255),
31     phone_number VARCHAR(20),
32     booking_id INT
33 );
34 • ALTER TABLE Customer
35 ADD CONSTRAINT fk_booking_id
36 FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
37
38
39 • CREATE TABLE if not exists Booking (
40     booking_id INT PRIMARY KEY,
41     customer_id INT,
42     event_id INT,
43     num_tickets INT,
44     total_cost DECIMAL(10, 2),
45     booking_date DATE,
46     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
47     FOREIGN KEY (event_id) REFERENCES Event(event_id)
48 );

```

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

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

```

50 • INSERT INTO Venu (venue_id, venue_name, address)
51 VALUES
52     (1, 'Venue 1', '123 Main Street, City 1, Country 1'),
53     (2, 'Venue 2', '456 Oak Avenue, City 2, Country 2'),
54     (3, 'Venue 3', '789 Maple Lane, City 3, Country 3'),
55     (4, 'Venue 4', '101 Pine Road, City 4, Country 4'),
56     (5, 'Venue 5', '202 Cedar Street, City 5, Country 5'),
57     (6, 'Venue 6', '303 Birch Boulevard, City 6, Country 6'),
58     (7, 'Venue 7', '404 Elm Drive, City 7, Country 7'),
59     (8, 'Venue 8', '505 Spruce Court, City 8, Country 8'),
60     (9, 'Venue 9', '606 Redwood Avenue, City 9, Country 9'),
61     (10, 'Venue 10', '707 Sycamore Lane, City 10, Country 10');
62 • select * from Venu;

```

100% 20:62

Result Grid Filter Rows: Search Edit Export/Import:

	venue_id	venue_name	address
▶	1	Venue 1	123 Main Street, City 1, Country 1
	2	Venue 2	456 Oak Avenue, City 2, Country 2
	3	Venue 3	789 Maple Lane, City 3, Country 3
	4	Venue 4	101 Pine Road, City 4, Country 4
	5	Venue 5	202 Cedar Street, City 5, Country 5
	6	Venue 6	303 Birch Boulevard, City 6, Country 6
	7	Venue 7	404 Elm Drive, City 7, Country 7
	8	Venue 8	505 Spruce Court, City 8, Country 8
	9	Venue 9	606 Redwood Avenue, City 9, Country 9
	10	Venue 10	707 Sycamore Lane, City 10, Country 10



```

66 • INSERT INTO Event (event_id, event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type, booking_id)
67 VALUES
68 (1, 'Event 1', '2023-12-13', '18:00:00', 1, 100, 100, 20.00, 'Movie', 101),
69 (2, 'Event 2', '2023-12-14', '19:30:00', 2, 150, 150, 30.00, 'Sports', 102),
70 (3, 'Event 3', '2023-12-15', '20:00:00', 3, 200, 200, 40.00, 'Concert', 103),
71 (4, 'Event 4', '2023-12-16', '17:45:00', 4, 120, 120, 25.00, 'Movie', 104),
72 (5, 'Event 5', '2023-12-17', '21:15:00', 5, 180, 180, 35.00, 'Sports', 105),
73 (6, 'Event 6', '2023-12-18', '16:30:00', 6, 250, 250, 50.00, 'Concert', 106),
74 (7, 'Event 7', '2023-12-19', '19:00:00', 7, 130, 130, 28.00, 'Movie', 107),
75 (8, 'Event 8', '2023-12-20', '18:45:00', 8, 160, 160, 32.00, 'Sports', 108),
76 (9, 'Event 9', '2023-12-21', '20:30:00', 9, 220, 220, 45.00, 'Concert', 109),
77 (10, 'Event 10', '2023-12-22', '17:00:00', 10, 140, 140, 30.00, 'Movie', 110);
78 • select * from Event;

```

100% 21:78

Result Grid Filter Rows: Search Edit: Export/Import:

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
▶	1	Event 1	2023-12-13	18:00:00	1	100	100	20.00	Movie	101
▶	2	Event 2	2023-12-14	19:30:00	2	150	150	30.00	Sports	102
▶	3	Event 3	2023-12-15	20:00:00	3	200	200	40.00	Concert	103
▶	4	Event 4	2023-12-16	17:45:00	4	120	120	25.00	Movie	104
▶	5	Event 5	2023-12-17	21:15:00	5	180	180	35.00	Sports	105
▶	6	Event 6	2023-12-18	16:30:00	6	250	250	50.00	Concert	106
▶	7	Event 7	2023-12-19	19:00:00	7	130	130	28.00	Movie	107
▶	8	Event 8	2023-12-20	18:45:00	8	160	160	32.00	Sports	108
▶	9	Event 9	2023-12-21	20:30:00	9	220	220	45.00	Concert	109
▶	10	Event 10	2023-12-22	17:00:00	10	140	140	30.00	Movie	110

```

80 • INSERT INTO Customer (customer_id, customer_name, email, phone_number, booking_id)
81 VALUES
82 (1, 'John Doe', 'john.doe@example.com', '123-456-7890', 101),
83 (2, 'Jane Smith', 'jane.smith@example.com', '987-654-3210', 102),
84 (3, 'Robert Johnson', 'robert.j@example.com', '555-123-4567', 103),
85 (4, 'Emily Davis', 'emily.d@example.com', '888-999-0000', 104),
86 (5, 'Michael Brown', 'michael.b@example.com', '111-222-3333', 105),
87 (6, 'Amanda White', 'amanda.w@example.com', '444-555-6666', 106),
88 (7, 'Daniel Taylor', 'daniel.t@example.com', '777-888-9999', 107),
89 (8, 'Sophia Miller', 'sophia.m@example.com', '333-666-9999', 108),
90 (9, 'Christopher Lee', 'chris.lee@example.com', '123-789-4560', 109),
91 (10, 'Olivia Wilson', 'olivia.w@example.com', '789-456-1230', 110);
92 • select * from Customer;

```

100% 24:92

Result Grid Filter Rows: Search Edit: Export/Import:

	customer_id	customer_name	email	phone_number	booking_id
▶	1	John Doe	john.doe@example.com	123-456-7890	101
▶	2	Jane Smith	jane.smith@example.com	987-654-3210	102
▶	3	Robert Johnson	robert.j@example.com	555-123-4567	103
▶	4	Emily Davis	emily.d@example.com	888-999-0000	104
▶	5	Michael Brown	michael.b@example.com	111-222-3333	105
▶	6	Amanda White	amanda.w@example.com	444-555-6666	106
▶	7	Daniel Taylor	daniel.t@example.com	777-888-9999	107
▶	8	Sophia Miller	sophia.m@example.com	333-666-9999	108
▶	9	Christopher Lee	chris.lee@example.com	123-789-4560	109
▶	10	Olivia Wilson	olivia.w@example.com	789-456-1230	110

```

93 • INSERT INTO Booking (booking_id, customer_id, event_id, num_tickets, total_cost, booking_date)
94 VALUES
95 (101, 1, 1, 2, 40.00, '2023-12-13'),
96 (102, 2, 2, 3, 90.00, '2023-12-14'),
97 (103, 3, 3, 1, 40.00, '2023-12-15'),
98 (104, 4, 4, 4, 100.00, '2023-12-16'),
99 (105, 5, 5, 2, 70.00, '2023-12-17'),
100 (106, 6, 6, 5, 250.00, '2023-12-18'),
101 (107, 7, 7, 3, 84.00, '2023-12-19'),
102 (108, 8, 8, 2, 64.00, '2023-12-20'),
103 (109, 9, 9, 3, 135.00, '2023-12-21'),
104 (110, 10, 10, 2, 60.00, '2023-12-22');
105 • select * from Booking;

```

100% 23:105

Result Grid Filter Rows: Search Edit: Export/Import:

	booking_id	customer_id	event_id	num_ticke...	total_cost	booking_date
▶	101	1	1	2	40.00	2023-12-13
▶	102	2	2	3	90.00	2023-12-14
▶	103	3	3	1	40.00	2023-12-15
▶	104	4	4	4	100.00	2023-12-16
▶	105	5	5	2	70.00	2023-12-17
▶	106	6	6	5	250.00	2023-12-18
▶	107	7	7	3	84.00	2023-12-19
▶	108	8	8	2	64.00	2023-12-20
▶	109	9	9	3	135.00	2023-12-21
▶	110	10	10	2	60.00	2023-12-22

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

```
121 • SELECT *
122 FROM Event
123 WHERE event_date BETWEEN '2023-01-01' AND '2023-12-31';
```

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
▶	1	Event 1	2023-12-13	18:00:00	1	100	100	20.00	Movie	101
	2	Event 2	2023-12-14	19:30:00	2	150	150	30.00	Sports	102
	3	Event 3	2023-12-15	20:00:00	3	200	200	40.00	Concert	103
	4	Event 4	2023-12-16	17:45:00	4	120	120	25.00	Movie	104
	5	Event 5	2023-12-17	21:15:00	5	180	180	35.00	Sports	105
	6	Event 6	2023-12-18	16:30:00	6	250	250	50.00	Concert	106
	7	Event 7	2023-12-19	19:00:00	7	130	130	28.00	Movie	107
	8	Event 8	2023-12-20	18:45:00	8	160	160	32.00	Sports	108
	9	Event 9	2023-12-21	20:30:00	9	220	220	45.00	Concert	109
	10	Event 10	2023-12-22	17:00:00	10	140	140	30.00	Movie	110

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

```
125 • SELECT *
126 FROM Event
127 WHERE available_seats > 0
128 AND event_name LIKE '%Concert%';
129
```

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
▶	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
130 • SELECT *
131 FROM Customer
132 ORDER BY customer_id
133 LIMIT 5 OFFSET 5;
```

	customer_id	customer_name	email	phone_number	booking_id
▶	6	Amanda White	amanda.w@example.com	444-555-6666	106
	7	Daniel Taylor	daniel.t@example.com	777-888-9999	107
	8	Sophia Miller	sophia.m@example.com	333-666-9999	108
	9	Christopher Lee	chris.lee@example.com	123-789-4560	109
	10	Olivia Wilson	olivia.w@example.com	789-456-1230	110

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

```
135 • SELECT *
136 FROM Booking
137 WHERE num_tickets > 4;
```

	booking_id	customer_id	event_id	num_ticke...	total_cost	booking_date
▶	106	6	6	5	250.00	2023-12-18

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

```
139 • SELECT *
140 FROM Customer
141 WHERE phone_number LIKE '%000';
```

147 100% 32:141

Result Grid

	customer_id	customer_name	email	phone_number	booking_id
▶ 4		Emily Davis	emily.d@example.com	888-999-0000	104

Result Grid

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

```
143 • SELECT *
144 FROM Event
145 WHERE total_seats > 15000
146 ORDER BY total_seats ASC;
```

147 100% 26:146

Result Grid

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
▶	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid

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

148 • SELECT *

149 FROM Event

150 WHERE event_name NOT LIKE 'x%'

151 AND event_name NOT LIKE 'y%'

152 AND event_name NOT LIKE 'z%';

152

100%

32:152

Result Grid

Filter Rows:

Search

Edit:

Export/Import:

	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_pri...	event_type	booking_id
▶ 1	Event 1	2023-12-13	18:00:00	1	100	100	20.00	Movie	101	
2	Event 2	2023-12-14	19:30:00	2	150	150	30.00	Sports	102	
3	Event 3	2023-12-15	20:00:00	3	200	200	40.00	Concert	103	
4	Event 4	2023-12-16	17:45:00	4	120	120	25.00	Movie	104	
5	Event 5	2023-12-17	21:15:00	5	180	180	35.00	Sports	105	
6	Event 6	2023-12-18	16:30:00	6	250	250	50.00	Concert	106	
7	Event 7	2023-12-19	19:00:00	7	130	130	28.00	Movie	107	
8	Event 8	2023-12-20	18:45:00	8	160	160	32.00	Sports	108	
9	Event 9	2023-12-21	20:30:00	9	220	220	45.00	Concert	109	
10	Event 10	2023-12-22	17:00:00	10	140	140	30.00	Movie	110	

Result Grid

Form Editor

Field Types

Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

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

```

155 • SELECT
156     event_id,
157     event_name,
158     AVG(ticket_price) AS average_ticket_price
159 FROM
160     Event
161 GROUP BY
162     event_id, event_name;

```

event_id	event_name	average_ticket_pri...
1	Event 1	20.000000
2	Event 2	30.000000
3	Event 3	40.000000
4	Event 4	25.000000
5	Event 5	35.000000
6	Event 6	50.000000
7	Event 7	28.000000
8	Event 8	32.000000
9	Event 9	45.000000
10	Event 10	30.000000

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

```

164 • SELECT
165     SUM(total_cost) AS total_revenue
166 FROM
167     Booking;
168

```

total_revenue
933.00

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

```

174 • SELECT
175     e.event_id,
176     e.event_name,
177     SUM(b.num_tickets) AS total_ticket_sales
178 FROM Event e
179 JOIN Booking b ON e.event_id = b.event_id
180 GROUP BY e.event_id
181 ORDER BY total_ticket_sales DESC
182 LIMIT 1;

```

event_id	event_name	total_ticket_sal...
6	Event 6	5

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

```

172 • SELECT
173     e.event_id,
174     e.event_name,
175     SUM(b.num_tickets) AS total_ticket_sales
176 FROM Event e
177 JOIN Booking b ON e.event_id = b.event_id
178 GROUP BY e.event_id

```

event_id	event_name	total_ticket_sal...
1	Event 1	2
2	Event 2	3
3	Event 3	1
4	Event 4	4
5	Event 5	2
6	Event 6	5
7	Event 7	3
8	Event 8	2
9	Event 9	3
10	Event 10	2

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

```
185 • SELECT
186     e.event_id,
187     e.event_name
188 FROM Event e
189 LEFT JOIN Booking b ON e.event_id = b.event_id
190 WHERE b.booking_id IS NULL;
```

100% 7:190

Result Grid Filter Rows: Search Export:

event_id	event_name
----------	------------

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

```
192 • SELECT
193     c.customer_id,
194     c.customer_name,
195     SUM(b.num_tickets) AS total_tickets_booked
196 FROM Customer c
197 JOIN Booking b ON c.customer_id = b.customer_id
198 GROUP BY c.customer_id, c.customer_name
199 ORDER BY total_tickets_booked DESC
200 LIMIT 1;
```

100% 10:199

Result Grid Filter Rows: Search Export: Fetch rows:

customer_id	customer_name	total_tickets_booked
-------------	---------------	----------------------

6	Amanda White	5
---	--------------	---

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

```
203 • SELECT
204     e.event_id,
205     e.event_name,
206     EXTRACT(MONTH FROM b.booking_date) AS booking_month,
207     SUM(b.num_tickets) AS total_tickets_sold
208 FROM Event e
209 JOIN Booking b ON e.event_id = b.event_id
210 GROUP BY e.event_id, e.event_name, EXTRACT(MONTH FROM b.booking_date);
```

100% 71:210

Result Grid Filter Rows: Search Export:

event_id	event_name	booking_mon...	total_tickets_s...
----------	------------	----------------	--------------------

1	Event 1	12	2
2	Event 2	12	3
3	Event 3	12	1
4	Event 4	12	4
5	Event 5	12	2
6	Event 6	12	5
7	Event 7	12	3
8	Event 8	12	2
9	Event 9	12	3
10	Event 10	12	2

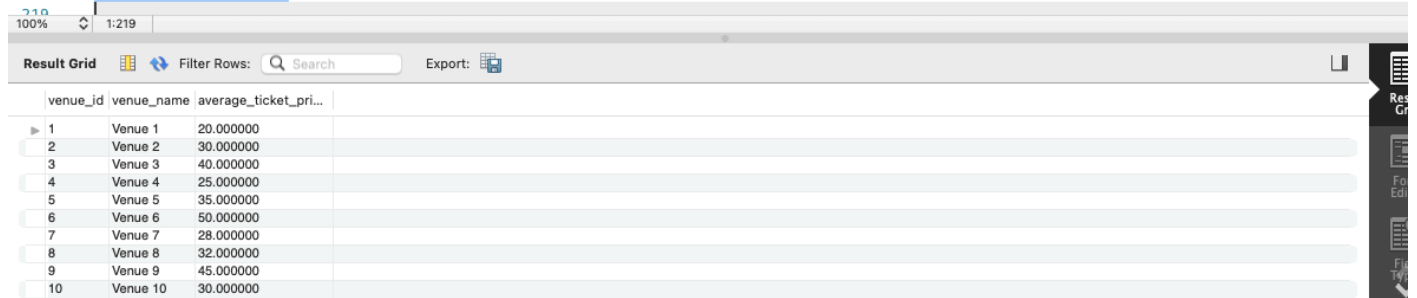
Result Grid

Form Editor

Field Types

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

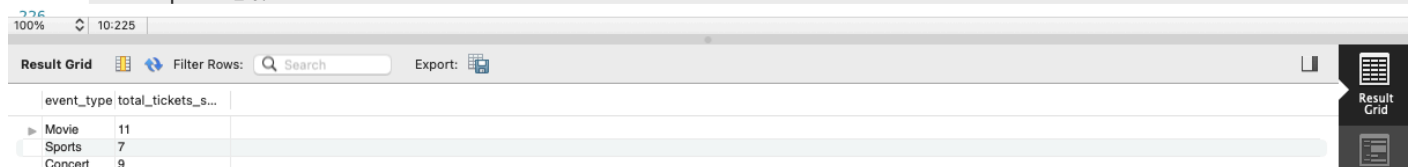
```
212 • SELECT
213     v.venue_id,
214     v.venue_name,
215     AVG(e.ticket_price) AS average_ticket_price
216 FROM Venue v
217 JOIN Event e ON v.venue_id = e.venue_id
218 GROUP BY v.venue_id;
```



venue_id	venue_name	average_ticket_price
1	Venue 1	20.000000
2	Venue 2	30.000000
3	Venue 3	40.000000
4	Venue 4	25.000000
5	Venue 5	35.000000
6	Venue 6	50.000000
7	Venue 7	28.000000
8	Venue 8	32.000000
9	Venue 9	45.000000
10	Venue 10	30.000000

9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.

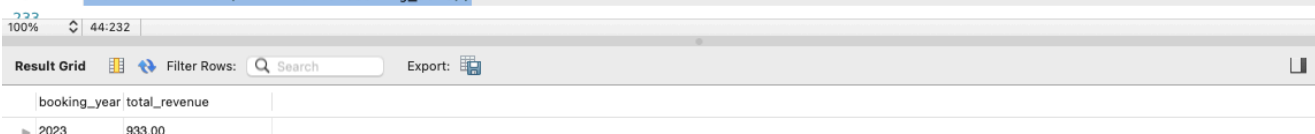
```
220 • SELECT
221     e.event_type,
222     SUM(b.num_tickets) AS total_tickets_sold
223 FROM Event e
224 JOIN Booking b ON e.event_id = b.event_id
225 GROUP BY e.event_type;
```



event_type	total_tickets_sold
Movie	11
Sports	7
Concert	9

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

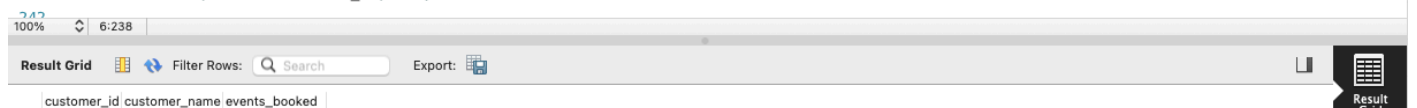
```
227 • SELECT
228     EXTRACT(YEAR FROM b.booking_date) AS booking_year,
229     SUM(e.ticket_price * b.num_tickets) AS total_revenue
230 FROM Event e
231 JOIN Booking b ON e.event_id = b.event_id
232 GROUP BY EXTRACT(YEAR FROM b.booking_date);
```



booking_year	total_revenue
2023	933.00

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

```
234 • SELECT
235     c.customer_id,
236     c.customer_name,
237     COUNT(DISTINCT b.event_id) AS events_booked
238 FROM Customer c
239 JOIN Booking b ON c.customer_id = b.customer_id
240 GROUP BY c.customer_id, c.customer_name
241 HAVING COUNT(DISTINCT b.event_id) > 1;
```



customer_id	customer_name	events_booked

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

```

244 • SELECT
245     c.customer_id,
246     c.customer_name,
247     SUM(e.ticket_price * b.num_tickets) AS total_revenue
248 FROM Customer c
249 JOIN Booking b ON c.customer_id = b.customer_id
250 JOIN Event e ON b.event_id = e.event_id
251 GROUP BY c.customer_id, c.customer_name;

```

100% 41:251

Result Grid Filter Rows: Search Export:

	customer_id	customer_name	total_revenue
▶ 1	John Doe	40.00	
▶ 2	Jane Smith	90.00	
▶ 3	Robert Johnson	40.00	
▶ 4	Emily Davis	100.00	
▶ 5	Michael Brown	70.00	
▶ 6	Amanda White	250.00	
▶ 7	Daniel Taylor	84.00	
▶ 8	Sophia Miller	64.00	
▶ 9	Christopher Lee	135.00	
▶ 10	Olivia Wilson	60.00	

Result Grid Form Editor Field Types

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

```

253 • SELECT
254     e.event_type,
255     v.venue_id,
256     v.venue_name,
257     AVG(e.ticket_price) AS average_ticket_price
258 FROM Event e
259 JOIN Venue v ON e.venue_id = v.venue_id
260 GROUP BY e.event_type, v.venue_id, v.venue_name;

```

100% 49:260

Result Grid Filter Rows: Search Export:

	event_type	venue_id	venue_name	average_ticket_pri...
▶ Movie	1	Venue 1	20.000000	
▶ Sports	2	Venue 2	30.000000	
▶ Concert	3	Venue 3	40.000000	
▶ Movie	4	Venue 4	25.000000	
▶ Sports	5	Venue 5	35.000000	
▶ Concert	6	Venue 6	50.000000	
▶ Movie	7	Venue 7	28.000000	
▶ Sports	8	Venue 8	32.000000	
▶ Concert	9	Venue 9	45.000000	
▶ Movie	10	Venue 10	30.000000	

Result Grid Form Editor Field Types

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

```

263 • SELECT
264     c.customer_id,
265     c.customer_name,
266     SUM(b.num_tickets) AS total_tickets_purchased
267 FROM Customer c
268 JOIN Booking b ON c.customer_id = b.customer_id
269 WHERE b.booking_date >= CURRENT_DATE - INTERVAL '30' DAY
270 GROUP BY c.customer_id, c.customer_name;
271

```

100% 10:270

Result Grid Filter Rows: Search Export:

	customer_id	customer_name	total_tickets_purcha...
▶ 1	John Doe	2	
▶ 2	Jane Smith	3	
▶ 3	Robert Johnson	1	
▶ 4	Emily Davis	4	
▶ 5	Michael Brown	2	
▶ 6	Amanda White	5	
▶ 7	Daniel Taylor	3	
▶ 8	Sophia Miller	2	
▶ 9	Christopher Lee	3	
▶ 10	Olivia Wilson	2	

Result Grid Form Editor Field Types

Tasks 4: Subquery and its types

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

```
273 • SELECT
274     v.venue_id,
275     v.venue_name,
276     (
277         SELECT AVG(e.ticket_price)
278         FROM Event e
279         WHERE e.venue_id = v.venue_id
280     ) AS average_ticket_price
281 FROM
282     Venue v;
```

100% 12:282

Result Grid Filter Rows: Search Export:

	venue_id	venue_name	average_ticket_pri...
▶	1	Venue 1	20.000000
▶	2	Venue 2	30.000000
▶	3	Venue 3	40.000000
▶	4	Venue 4	25.000000
▶	5	Venue 5	35.000000
▶	6	Venue 6	50.000000
▶	7	Venue 7	28.000000
▶	8	Venue 8	32.000000
▶	9	Venue 9	45.000000
▶	10	Venue 10	30.000000

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

```
285 • SELECT
286     e.event_id,
287     e.event_name
288 FROM Event e
289 WHERE
290     (
291         SELECT COUNT(*)
292         FROM Booking b
293         WHERE b.event_id = e.event_id
294     ) > 0.5 * e.total_seats;
```

100% 1:295

Result Grid Filter Rows: Search Edit: Export/Import:

	event_id	event_name
▶	NULL	NULL

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

```
296 • SELECT
297     e.event_id,
298     e.event_name,
299     (
300         SELECT SUM(b.num_tickets)
301         FROM Booking b
302         WHERE b.event_id = e.event_id
303     ) AS total_tickets_sold
304 FROM Event e;
```

100% 14:304

Result Grid Filter Rows: Search Export:

	event_id	event_name	total_tickets_s...
▶	1	Event 1	2
▶	2	Event 2	3
▶	3	Event 3	1
▶	4	Event 4	4
▶	5	Event 5	2
▶	6	Event 6	5
▶	7	Event 7	3
▶	8	Event 8	2
▶	9	Event 9	3
▶	10	Event 10	2

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

```
306 • SELECT
307     c.customer_id,
308     c.customer_name
309 FROM Customer c
310 WHERE NOT EXISTS (
311     SELECT 1
312     FROM Booking b
313     WHERE b.customer_id = c.customer_id
314 );
```

100% 3:314

Result Grid Filter Rows: Search Edit: Export/Import:

customer_id	customer_name
NULL	NULL

Result Grid

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

```
316 • SELECT
317     event_id,
318     event_name
319 FROM Event
320 WHERE event_id NOT IN (
321     SELECT DISTINCT event_id
322     FROM Booking
323 );
```

100% 3:323

Result Grid Filter Rows: Search Edit: Export/Import:

event_id	event_name
NULL	NULL

Result Grid

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

```
326 • SELECT
327     event_id,
328     event_name,
329     ticket_price
330 FROM Event
331 WHERE ticket_price > (
332     SELECT AVG(ticket_price)
333     FROM Event
334 );
```

100% 3:334

Result Grid Filter Rows: Search Edit: Export/Import:

event_id	event_name	ticket_pri...
3	Event 3	40.00
5	Event 5	35.00
6	Event 6	50.00
9	Event 9	45.00

Result Grid Form

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

```
336 • SELECT
337     c.customer_id,
338     c.customer_name,
339     (
340         SELECT SUM(e.ticket_price * b.num_tickets)
341         FROM Event e
342         JOIN Booking b ON e.event_id = b.event_id
343         WHERE b.customer_id = c.customer_id
344     ) AS total_revenue
345 FROM Customer c;
```

Result Grid			Filter Rows: <input type="text" value="Search"/>	Export:
customer_id	customer_name	total_revenue		
1	John Doe	40.00		
2	Jane Smith	90.00		
3	Robert Johnson	40.00		
4	Emily Davis	100.00		
5	Michael Brown	70.00		
6	Amanda White	250.00		
7	Daniel Taylor	84.00		
8	Sophia Miller	64.00		
9	Christopher Lee	135.00		
10	Olivia Wilson	60.00		

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

```
348 • SELECT
349     c.customer_id,
350     c.customer_name
351 FROM Customer c
352 WHERE EXISTS ( SELECT 1
353                 FROM Booking b
354                 JOIN Event e ON b.event_id = e.event_id
355                 WHERE b.customer_id = c.customer_id
356                       AND e.venue_id = '1'
357               );
358
```

Result Grid		Filter Rows: <input type="text" value="Search"/>	Edit:	Export/Import:
customer_id	customer_name			
1	John Doe			

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

```
359 • SELECT
360     e.event_type,
361     (
362         SELECT SUM(b.num_tickets)
363         FROM Booking b
364         WHERE b.event_id IN (
365             SELECT event_id
366             FROM Event
367             WHERE event_type = e.event_type
368         )
369     ) AS total_tickets_sold
370 FROM Event e
371 GROUP BY e.event_type;
```

event_type	total_tickets_s...
Movie	11
Sports	7
Concert	9

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

```
373 • SELECT
374     c.customer_id,
375     c.customer_name,
376     DATE_FORMAT(b.booking_date, '%Y-%m') AS booking_month
377 FROM
378     Customer c
379 JOIN
380     Booking b ON c.customer_id = b.customer_id
381 GROUP BY
382     c.customer_id, c.customer_name, booking_month;
```

customer_id	customer_name	booking_mon...
1	John Doe	2023-12
2	Jane Smith	2023-12
3	Robert Johnson	2023-12
4	Emily Davis	2023-12
5	Michael Brown	2023-12
6	Amanda White	2023-12
7	Daniel Taylor	2023-12
8	Sophia Miller	2023-12
9	Christopher Lee	2023-12
10	Olivia Wilson	2023-12

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

```
386 • SELECT
387     v.venue_id,
388     v.venue_name,
389     (
390         SELECT AVG(e.ticket_price)
391         FROM Event e
392         WHERE e.venue_id = v.venue_id
393     ) AS average_ticket_price
394 FROM
395     Venue v;
```

venue_id	venue_name	average_ticket_pri...
1	Venue 1	20.000000
2	Venue 2	30.000000
3	Venue 3	40.000000
4	Venue 4	25.000000
5	Venue 5	35.000000
6	Venue 6	50.000000
7	Venue 7	28.000000
8	Venue 8	32.000000
9	Venue 9	45.000000
10	Venue 10	30.000000

