

Tasks 1: Database Design:

1. Create the database named "TicketBookingSystem".

```
980 • CREATE DATABASE TicketBookingSystem;
981 • USE TicketBookingSystem;
```

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

- Venu
- Event
- Customers
- Booking

```
983 • CREATE TABLE Venu (
984     venue_id INT PRIMARY KEY AUTO_INCREMENT,
985     venue_name VARCHAR(255) NOT NULL,
986     address TEXT NOT NULL
987 );
988
989 • CREATE TABLE Customer (
990     customer_id INT PRIMARY KEY AUTO_INCREMENT,
991     customer_name VARCHAR(255) NOT NULL,
992     email VARCHAR(255) NOT NULL UNIQUE,
993     phone_number VARCHAR(20) NOT NULL
994 );
995
996
997 • CREATE TABLE Booking (
998     booking_id INT PRIMARY KEY AUTO_INCREMENT,
999     customer_id INT NOT NULL,
1000     num_tickets INT NOT NULL,
1001     total_cost DECIMAL(10,2) NOT NULL,
1002     booking_date DATE NOT NULL,
1003     FOREIGN KEY (customer_id) REFERENCES Customer (customer_id)
1004 );
1005
```

```

1006 • ALTER TABLE Customer ADD booking_id INT;
1007 • ALTER TABLE Customer ADD CONSTRAINT fk_customer_booking_id FOREIGN KEY (booking_id) REFERENCES Booking (booking_id);
1008 • DESC CUSTOMER;
1009 • ALTER TABLE Booking ADD event_id INT;
1010 • ALTER TABLE Booking ADD CONSTRAINT fk_booking_event_id FOREIGN KEY (event_id) REFERENCES Event (event_id);
1011
1012
1013
1014 • CREATE TABLE Event (
1015     event_id INT PRIMARY KEY AUTO_INCREMENT,
1016     event_name VARCHAR(255) NOT NULL,
1017     event_date DATE NOT NULL,
1018     event_time TIME NOT NULL,
1019     venue_id INT NOT NULL,
1020     total_seats INT NOT NULL,
1021     available_seats INT NOT NULL,
1022     ticket_price DECIMAL(10,2) NOT NULL,
1023     event_type ENUM('Movie', 'Sports', 'Concert') NOT NULL,
1024     booking_id INT,
1025     FOREIGN KEY (venue_id) REFERENCES Venu (venue_id),
1026     FOREIGN KEY (booking_id) REFERENCES Booking (booking_id)
1027 );

```

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

- Primary Keys:
 - Venu: venue_id
 - Event: event_id
 - Customer: customer_id
 - Booking: booking_id
- Foreign Keys:
 - Event: venue_id references Venu (venue_id)
 - Event: booking_id references Booking (booking_id)
 - Customer: booking_id references Booking (booking_id)
 - Booking: customer_id references Customer (customer_id)
 - Booking: event_id references Event (event_id)

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

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

```

1018 • INSERT INTO Venu (venue_name, address)
1019 VALUES
1020 ('Grand Cinema', '123 Main Street'),
1021 ('Stadium Arena', '456 Sports Complex'),
1022 ('Music Hall', '789 Concert Avenue'),
1023 ('Comedy Club', '1011 Laugh Street'),
1024 ('Art Gallery', '1213 Exhibit Lane'),
1025 ('Theater Center', '1415 Stage Road'),
1026 ('Opera House', '1617 Aria Drive'),
1027 ('Dance Studio', '1819 Motion Place'),
1028 ('Conference Hall', '2021 Meeting Blvd'),
1029 ('Bookstore Cafe', '2223 Read Avenue');
1030 • SELECT * FROM Venu;
1031

```

Result Grid			
Filter Rows:			
Edit: Export/Import: Wrap Cell Content:			
venue_id	venue_name	address	
1	Grand Cinema	123 Main Street	
2	Stadium Arena	456 Sports Complex	
3	Music Hall	789 Concert Avenue	
4	Comedy Club	1011 Laugh Street	
5	Art Gallery	1213 Exhibit Lane	
6	Theater Center	1415 Stage Road	
7	Opera House	1617 Aria Drive	
8	Dance Studio	1819 Motion Place	
9	Conference Hall	2021 Meeting Blvd	
10	Bookstore Cafe	2223 Read Avenue	
NULL	NULL	NULL	

```

1032 • INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type)
1033 VALUES
1034 ('Movie Marathon', '2023-12-20', '18:00:00', 1, 500, 500, 1500, 'Movie'),
1035 ('Rock Concert', '2023-12-25', '20:00:00', 2, 10000, 7000, 2500, 'Concert'),
1036 ('Football Match', '2023-12-30', '16:00:00', 3, 15000, 10000, 500, 'Sports'),
1037 ('Art Exhibition', '2023-12-15', '10:00:00', 5, 200, 150, 1000, 'Concert'),
1038 ('Comedy Stand-up', '2023-12-22', '19:00:00', 4, 300, 250, 1200, 'Sports'),
1039 ('Book Reading', '2023-12-27', '14:00:00', 10, 100, 80, 500, 'Movie'),
1040 ('Ballet Performance', '2023-12-29', '18:00:00', 6, 500, 450, 2000, 'Concert'),
1041 ('Opera Show', '2024-01-02', '19:30:00', 7, 800, 720, 3000, 'Concert'),
1042 ('Music Festival', '2024-01-05', '14:00:00', 2, 20000, 15000, 1800, 'Concert'),
1043 ('Dance Competition', '2024-01-10', '17:00:00', 8, 1200, 1000, 1500, 'Sports');
1044

```

```

1047 • INSERT INTO Customer (customer_name, email, phone_number)
1048 VALUES
1049 ('John Doe', 'john.doe@example.com', '+1234567890'),
1050 ('Jane Doe', 'jane.doe@example.com', '+9876543210'),
1051 ('Alice Smith', 'alice.smith@email.com', '+5551234567'),
1052 ('Bob Johnson', 'bob.johnson@mail.com', '+3337654321'),
1053 ('Mary Taylor', 'mary.taylor@web.net', '+4448765432'),
1054 ('David Miller', 'david.miller@domain.com', '+2229876543'),
1055 ('Sarah Williams', 'sarah.williams@yahoo.com', '+6663214567'),
1056 ('Michael Brown', 'michael.brown@gmail.com', '+7775432109'),
1057 ('Olivia Jones', 'olivia.jones@outlook.com', '+8881234567'),
1058 ('Emily Garcia', 'emily.garcia@hotmail.com', '+9997654321');
1059

```

```

1060 • INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date)
1061 VALUES
1062 (1, 31, 2, 3000, '2023-12-19'),
1063 (2, 32, 1, 2500, '2023-12-22'),
1064 (3, 33, 1, 1800, '2023-12-25'),
1065 (4, 34, 3, 4500, '2023-12-20'),
1066 (5, 31, 2, 4000, '2024-01-02'),
1067 (6, 33, 4, 6000, '2024-01-05'),
1068 (7, 37, 3, 6750, '2024-01-10'),
1069 (8, 36, 1, 1500, '2023-12-27'),
1070 (9, 35, 2, 6000, '2023-12-29'),
1071 (10, 35, 2, 2000, '2023-12-15');

```

2. Write a SQL query to list all Events.

```

1071 • SELECT * FROM Event;

```

Result Grid									
Filter Rows:									
Edit:									
Export/Import:									
Wrap Cell Content:									
	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type
▶	31	Movie Marathon	2023-12-20	18:00:00	1	500	500	1500.00	Movie
	32	Rock Concert	2023-12-25	20:00:00	2	10000	7000	2500.00	Concert
	33	Football Match	2023-12-30	16:00:00	3	15000	10000	500.00	Sports
	34	Art Exhibition	2023-12-15	10:00:00	5	200	150	1000.00	Concert
	35	Comedy Stand-up	2023-12-22	19:00:00	4	300	250	1200.00	Sports
	36	Book Reading	2023-12-27	14:00:00	10	100	80	500.00	Movie
	37	Ballet Performance	2023-12-29	18:00:00	6	500	450	2000.00	Concert
	38	Opera Show	2024-01-02	19:30:00	7	800	720	3000.00	Concert
	39	Music Festival	2024-01-05	14:00:00	2	20000	15000	1800.00	Concert
	40	Dance Competition	2024-01-10	17:00:00	8	1200	1000	1500.00	Sports
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```

1074 • SELECT * FROM Event WHERE available_seats > 0;

```

Result Grid									
Filter Rows:									
Edit:									
Export/Import:									
Wrap Cell Content:									
	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type
▶	31	Movie Marathon	2023-12-20	18:00:00	1	500	500	1500.00	Movie
	32	Rock Concert	2023-12-25	20:00:00	2	10000	7000	2500.00	Concert
	33	Football Match	2023-12-30	16:00:00	3	15000	10000	500.00	Sports
	34	Art Exhibition	2023-12-15	10:00:00	5	200	150	1000.00	Concert
	35	Comedy Stand-up	2023-12-22	19:00:00	4	300	250	1200.00	Sports
	36	Book Reading	2023-12-27	14:00:00	10	100	80	500.00	Movie
	37	Ballet Performance	2023-12-29	18:00:00	6	500	450	2000.00	Concert
	38	Opera Show	2024-01-02	19:30:00	7	800	720	3000.00	Concert
	39	Music Festival	2024-01-05	14:00:00	2	20000	15000	1800.00	Concert
	40	Dance Competition	2024-01-10	17:00:00	8	1200	1000	1500.00	Sports
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
1076 • SELECT * FROM Event WHERE event_name LIKE '%cup%';
```

```
1077
```

```
1078
```

Result Grid									
	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
1078 • SELECT * FROM Event WHERE ticket_price BETWEEN 1000 AND 2500;
```

```
1079
```

```
1080
```

Result Grid									
	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type
▶	31	Movie Marathon	2023-12-20	18:00:00	1	500	500	1500.00	Movie
	32	Rock Concert	2023-12-25	20:00:00	2	10000	7000	2500.00	Concert
	34	Art Exhibition	2023-12-15	10:00:00	5	200	150	1000.00	Concert
	35	Comedy Stand-up	2023-12-22	19:00:00	4	300	250	1200.00	Sports
	37	Ballet Performance	2023-12-29	18:00:00	6	500	450	2000.00	Concert
	39	Music Festival	2024-01-05	14:00:00	2	20000	15000	1800.00	Concert
	40	Dance Competition	2024-01-10	17:00:00	8	1200	1000	1500.00	Sports
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
1080 • SELECT * FROM Event WHERE event_date BETWEEN '2023-12-20' AND '2023-12-30';
```

```
1081
```

```
1082
```

Result Grid									
	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type
▶	31	Movie Marathon	2023-12-20	18:00:00	1	500	500	1500.00	Movie
	32	Rock Concert	2023-12-25	20:00:00	2	10000	7000	2500.00	Concert
	33	Football Match	2023-12-30	16:00:00	3	15000	10000	500.00	Sports
	35	Comedy Stand-up	2023-12-22	19:00:00	4	300	250	1200.00	Sports
	36	Book Reading	2023-12-27	14:00:00	10	100	80	500.00	Movie
	37	Ballet Performance	2023-12-29	18:00:00	6	500	450	2000.00	Concert
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
1082 • SELECT * FROM Event WHERE available_seats > 0 AND event_name LIKE '%Concert%';
```

```
1083
```

```
1084
```






Result Grid									
	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type
▶	32	Rock Concert	2023-12-25	20:00:00	2	10000	7000	2500.00	Concert
*	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.


```
1084 • SELECT * FROM Customer LIMIT 5 OFFSET 5;
```

```
1085
```

```
1086
```






Result Grid				
Filter Rows: <input type="text"/>				
Edit:   				
Export/Import:  				
	customer_id	customer_name	email	phone_number
▶	6	David Miller	david.miller@domain.com	+2229876543
	7	Sarah Williams	sarah.williams@yahoo.com	+6663214567
	8	Michael Brown	michael.brown@gmail.com	+7775432109
	9	Olivia Jones	olivia.jones@outlook.com	+8881234567
	10	Emily Garcia	emily.garcia@hotmail.com	+9997654321
*	NULL	NULL	NULL	NULL

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

```
1086 • SELECT * FROM Booking WHERE num_tickets > 4;
```

```
1087
```

```
1088
```






Result Grid					
Filter Rows: <input type="text"/>					
Edit:   					
Export/Import:  					
	booking_id	customer_id	event_id	num_tickets	total_cost
*	NULL	NULL	NULL	NULL	NULL

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

```
1088 • SELECT * FROM Customer WHERE phone_number LIKE '%000';
```

```
1089
```

```
1090
```







Result Grid				
Filter Rows: <input type="text"/>				
Edit:   				
Export/Import:  				
	customer_id	customer_name	email	phone_number
*	NULL	NULL	NULL	NULL

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

```
1090 • SELECT * FROM Event WHERE total_seats > 15000 ORDER BY total_seats DESC;
```

```
1091
```

```
1092
```

Result Grid									
Filter Rows: <input type="text"/>									
Edit:   									
Export/Import:  									
Wrap Cell Content: 									
	event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type
▶	39	Music Festival	2024-01-05	14:00:00	2	20000	15000	1800.00	Concert
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

1092 • `SELECT * FROM Event WHERE event_name NOT LIKE 'x%' AND event_name NOT LIKE 'y%' AND event_name NOT LIKE 'z%';`

1093

event_id	event_name	event_date	event_time	venue_id	total_seats	available_seats	ticket_price	event_type
31	Movie Marathon	2023-12-20	18:00:00	1	500	500	1500.00	Movie
32	Rock Concert	2023-12-25	20:00:00	2	10000	7000	2500.00	Concert
33	Football Match	2023-12-30	16:00:00	3	15000	10000	500.00	Sports
34	Art Exhibition	2023-12-15	10:00:00	5	200	150	1000.00	Concert
35	Comedy Stand-up	2023-12-22	19:00:00	4	300	250	1200.00	Sports
36	Book Reading	2023-12-27	14:00:00	10	100	80	500.00	Movie
37	Ballet Performance	2023-12-29	18:00:00	6	500	450	2000.00	Concert
38	Opera Show	2024-01-02	19:30:00	7	800	720	3000.00	Concert
39	Music Festival	2024-01-05	14:00:00	2	20000	15000	1800.00	Concert
40	Dance Competition	2024-01-10	17:00:00	8	1200	1000	1500.00	Sports
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

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

1094 • `SELECT event_name, AVG(ticket_price) AS avg_ticket_price`
 1095 `FROM Event`
 1096 `GROUP BY event_name`
 1097 `ORDER BY avg_ticket_price DESC;`

1098

1099

event_name	avg_ticket_price
Opera Show	3000.000000
Rock Concert	2500.000000
Ballet Performance	2000.000000
Music Festival	1800.000000
Movie Marathon	1500.000000
Dance Competition	1500.000000
Comedy Stand-up	1200.000000
Art Exhibition	1000.000000
Football Match	500.000000
Book Reading	500.000000

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

1099 • `SELECT SUM(num_tickets * total_cost) AS total_revenue`
 1100 `FROM Booking;`

1101

total_revenue
93550.00

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

```

1102 • SELECT Event.event_name, SUM(num_tickets) AS total_tickets_sold
1103 FROM Booking
1104 JOIN Event ON Event.event_id=Booking.event_id
1105 GROUP BY event_name
1106 ORDER BY total_tickets_sold DESC
1107 LIMIT 1;
1108

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:	Fetch rows
event_name	total_tickets_sold			
Football Match	5			

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

```

1109 • SELECT Event.event_name, SUM(num_tickets) AS total_tickets_sold
1110 FROM Booking
1111 JOIN Event ON Event.event_id=Booking.event_id
1112 GROUP BY event_name
1113 ORDER BY total_tickets_sold;
1114

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
event_name	total_tickets_sold		
Rock Concert	1		
Book Reading	1		
Art Exhibition	3		
Ballet Performance	3		
Movie Marathon	4		
Comedy Stand-up	4		
Football Match	5		

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

```

1115 • SELECT event_name
1116 FROM Event
1117 WHERE event_id NOT IN (SELECT event_id FROM Booking);
1118
1119

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
event_name			
Opera Show			
Music Festival			
Dance Competition			

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


```

1119 • SELECT customer_name, SUM(num_tickets) AS total_tickets_booked
1120 FROM Customer
1121 INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
1122 GROUP BY customer_name
1123 ORDER BY total_tickets_booked DESC
1124 LIMIT 1;
1125
1126

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows

	customer_name	total_tickets_booked
▶	David Miller	4

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

```

1126 • SELECT MONTHNAME(booking_date) AS month, SUM(num_tickets) AS total_tickets_sold
1127 FROM Booking
1128 GROUP BY MONTH(booking_date)
1129 ORDER BY month;





```

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

```

1131 • SELECT venue_name, AVG(ticket_price) AS avg_ticket_price
1132 FROM Event
1133 INNER JOIN Venu ON Event.venue_id = Venu.venue_id
1134 GROUP BY venue_name
1135 ORDER BY avg_ticket_price DESC;
1136
1137

```

Result Grid |   Filter Rows: | Export:  | Wrap Cell Content: 

	venue_name	avg_ticket_price
▶	Opera House	3000.000000
	Stadium Arena	2150.000000
	Theater Center	2000.000000
	Grand Cinema	1500.000000
	Dance Studio	1500.000000
	Comedy Club	1200.000000
	Art Gallery	1000.000000
	Music Hall	500.000000
	Bookstore Cafe	500.000000

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

```

1137 • SELECT event_type, SUM(num_tickets) AS total_tickets_sold
1138 FROM Booking
1139 INNER JOIN Event ON Booking.event_id = Event.event_id
1140 GROUP BY event_type
1141 ORDER BY total_tickets_sold DESC;
1142
1143

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
event_type	total_tickets_sold		
Sports	9		
Concert	7		
Movie	5		

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

```

1143 • SELECT YEAR(booking_date) AS year, SUM(num_tickets * total_cost) AS total_revenue
1144 FROM Booking
1145 GROUP BY YEAR(booking_date)
1146 ORDER BY year;
1147

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
year	total_revenue		
2023	41300.00		
2024	52250.00		

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

```

1148 • SELECT DISTINCT customer_name
1149 FROM Customer
1150 INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
1151 GROUP BY customer_name
1152 HAVING COUNT(booking_id) > 1;
1153
1154

```



Result Grid	Filter Rows:	Export:	Wrap Cell Content:
customer_name			

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

```

1154 • SELECT customer_name, SUM(num_tickets * total_cost) AS total_revenue
1155 FROM Customer
1156 INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
1157 GROUP BY customer_name
1158 ORDER BY total_revenue DESC;
1159
1160

```



Result Grid		
Filter Rows: <input type="text"/>		
Export:  Wrap Cell Content: 		
	customer_name	total_revenue
▶	David Miller	24000.00
	Sarah Williams	20250.00
	Bob Johnson	13500.00
	Olivia Jones	12000.00
	Mary Taylor	8000.00
	John Doe	6000.00
	Emily Garcia	4000.00
	Jane Doe	2500.00
	Alice Smith	1800.00
	Michael Brown	1500.00

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

```

1160 • SELECT event_type, venue_name, AVG(ticket_price) AS avg_ticket_price
1161 FROM Event
1162 INNER JOIN Venu ON Event.venue_id = Venu.venue_id
1163 GROUP BY event_type, venue_name
1164 ORDER BY event_type, venue_name;
1165
1166

```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	event_type	venue_name	avg_ticket_price
▶	Movie	Bookstore Cafe	500.000000
	Movie	Grand Cinema	1500.000000
	Sports	Comedy Club	1200.000000
	Sports	Dance Studio	1500.000000
	Sports	Music Hall	500.000000
	Concert	Art Gallery	1000.000000
	Concert	Opera House	3000.000000
	Concert	Stadium Arena	2150.000000
	Concert	Theater Center	2000.000000

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

```

1166 • SELECT customer_name, SUM(num_tickets) AS total_tickets_purchased
1167 FROM Customer
1168 INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
1169 WHERE booking_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
1170 GROUP BY customer_name
1171 ORDER BY total_tickets_purchased DESC;
1172
1173

```

Result Grid		
	customer_name	total_tickets_purchased
▶	David Miller	4
	Bob Johnson	3
	Sarah Williams	3
	John Doe	2
	Mary Taylor	2
	Olivia Jones	2
	Emily Garcia	2
	Jane Doe	1
	Alice Smith	1
	Michael Brown	1

Tasks 4: Subquery and its types

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

```

1173 • SELECT venue_name, AVG(ticket_price) AS avg_ticket_price
1174 FROM Event
1175 INNER JOIN Venu ON Event.venue_id = Venu.venue_id
1176 GROUP BY venue_name
1177 ORDER BY avg_ticket_price DESC;
1178
1179

```

Result Grid		
	venue_name	avg_ticket_price
▶	Opera House	3000.000000
	Stadium Arena	2150.000000
	Theater Center	2000.000000
	Grand Cinema	1500.000000
	Dance Studio	1500.000000
	Comedy Club	1200.000000
	Art Gallery	1000.000000
	Music Hall	500.000000
	Bookstore Cafe	500.000000

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

```

1179 • SELECT event_name, available_seats / total_seats AS ticket_sold_percentage
1180 FROM Event
1181 WHERE available_seats / total_seats * 100 < 50;
1182
1183

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
event_name	ticket_sold_percentage		

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

```

1183 • SELECT event_name, SUM(num_tickets) AS total_tickets_sold
1184 FROM (
1185     SELECT event_id, num_tickets
1186     FROM Booking
1187 ) AS b
1188 INNER JOIN Event ON b.event_id = Event.event_id
1189 GROUP BY event_name
1190 ORDER BY total_tickets_sold DESC;
1191
1192

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
event_name	total_tickets_sold		
Football Match	5		
Movie Marathon	4		
Comedy Stand-up	4		
Art Exhibition	3		
Ballet Performance	3		
Rock Concert	1		
Book Reading	1		

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


```

1192 • SELECT customer_name
1193 FROM Customer
1194 WHERE NOT EXISTS (
1195     SELECT *
1196     FROM Booking
1197     WHERE customer_id = Customer.customer_id
1198 );
1199
1200

```

Result Grid | Filter Rows: | Export: | Wrap Cell

customer_name

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

```

1200 • SELECT event_name
1201 FROM Event
1202 WHERE event_id NOT IN (
1203     SELECT event_id
1204     FROM Booking
1205 );
1206
1207

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

event_name
Opera Show
Music Festival
Dance Competition

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

```

1206 • SELECT event_type, SUM(num_tickets) AS total_tickets_sold
1207 FROM (
1208     SELECT event_id, num_tickets
1209     FROM Booking
1210 ) AS b
1211 INNER JOIN Event ON b.event_id = Event.event_id
1212 GROUP BY event_type
1213 ORDER BY total_tickets_sold DESC;
1214
1215

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	event_type	total_tickets_sold
▶	Sports	9
	Concert	7
	Movie	5

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

```

1215 • SELECT event_name, ticket_price
1216 FROM Event
1217 WHERE ticket_price > (
1218     SELECT AVG(ticket_price)
1219     FROM Event
1220 );
1221
1222

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	event_name	ticket_price
▶	Rock Concert	2500.00
	Ballet Performance	2000.00
	Opera Show	3000.00
	Music Festival	1800.00

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

```

1222 • SELECT customer_name, SUM(total_cost * num_tickets) AS total_revenue
1223 FROM Customer
1224 INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
1225 GROUP BY customer_name
1226 ORDER BY total_revenue DESC;
1227

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	customer_name	total_revenue
▶	David Miller	24000.00
	Sarah Williams	20250.00
	Bob Johnson	13500.00
	Olivia Jones	12000.00
	Mary Taylor	8000.00
	John Doe	6000.00
	Emily Garcia	4000.00
	Jane Doe	2500.00
	Alice Smith	1800.00
	Michael Brown	1500.00

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

```

1228 • SELECT customer_name
1229 FROM Customer
1230 INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
1231 INNER JOIN Event ON Booking.event_id = Event.event_id
1232 WHERE Event.venue_id = (
1233     SELECT venue_id
1234     FROM Venu
1235     WHERE venue_name = 'Grand Cinema'
1236 );
1237
1238

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)




	customer_name
▶	John Doe
	Mary Taylor

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

```

1238 • SELECT event_type, SUM(num_tickets) AS total_tickets_sold
1239 FROM (
1240     SELECT Booking.event_id, num_tickets, event_type
1241     FROM Booking
1242     INNER JOIN Event ON Booking.event_id = Event.event_id
1243 ) AS b
1244 GROUP BY event_type
1245 ORDER BY total_tickets_sold DESC;
1246
1247
1248

```




Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 		
	event_type	total_tickets_sold
▶	Sports	9
	Concert	7
	Movie	5

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

```



1247 • SELECT customer_name, DATE_FORMAT(booking_date, '%M') AS booking_month
1248 FROM Customer
1249 INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
1250 GROUP BY customer_name, booking_month
1251 ORDER BY customer_name, booking_month;
1252
1253

```

Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 		
	customer_name	booking_month
▶	Alice Smith	December
	Bob Johnson	December
	David Miller	January
	Emily Garcia	December
	Jane Doe	December
	John Doe	December
	Mary Taylor	January
	Michael Brown	December
	Olivia Jones	December
	Sarah Williams	January

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

```
1253 • SELECT venue_name, AVG(ticket_price) AS avg_ticket_price
1254 FROM Event
1255 INNER JOIN Venu ON Event.venue_id = Venu.venue_id
1256 GROUP BY venue_name
1257 ORDER BY avg_ticket_price DESC;
1258
1259
```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

	venue_name	avg_ticket_price
▶	Opera House	3000.000000
	Stadium Arena	2150.000000
	Theater Center	2000.000000
	Grand Cinema	1500.000000
	Dance Studio	1500.000000
	Comedy Club	1200.000000
	Art Gallery	1000.000000
	Music Hall	500.000000
	Bookstore Cafe	500.000000