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 (
          venue_id INT PRIMARY KEY AUTO_INCREMENT,
          venue_name VARCHAR(255) NOT NULL,
           address TEXT NOT NULL
986
       );
987
989 • ⊖ CREATE TABLE Customer (
         customer_id INT PRIMARY KEY AUTO_INCREMENT,
990
          customer_name VARCHAR(255) NOT NULL,
          email VARCHAR(255) NOT NULL UNIQUE,
992
993
          phone_number VARCHAR(20) NOT NULL
       );
994
996
997 • ⊖ CREATE TABLE Booking (
          booking id INT PRIMARY KEY AUTO INCREMENT,
          customer_id INT NOT NULL,
999
          num_tickets INT NOT NULL,
1000
           total_cost DECIMAL(10,2) NOT NULL,
          booking_date DATE NOT NULL,
1002
           FOREIGN KEY (customer_id) REFERENCES Customer (customer_id)
1003
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);
1012
1013
1014 • ⊖ CREATE TABLE Event (
1015
        event_id INT PRIMARY KEY AUTO_INCREMENT,
        event_name VARCHAR(255) NOT NULL,
1016
1017
         event_date DATE NOT NULL,
          event_time TIME NOT NULL,
         venue id INT NOT NULL,
1019
1020
        total_seats INT NOT NULL,
        available_seats INT NOT NULL,
1022
        ticket_price DECIMAL(10,2) NOT NULL,
         event_type ENUM('Movie', 'Sports', 'Concert') NOT NULL,
1023
        FOREIGN KEY (venue_id) REFERENCES Venu (venue_id),
1025
1026
        FOREIGN KEY (booking_id) REFERENCES Booking (booking_id)
1027 );
```

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

o Venu: venue id

o Event: event id

o Customer: customer_id

o 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:

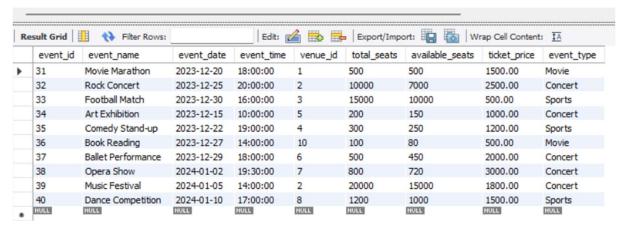
1. Write a SQL query to insert at least 10 sample records into each table.

```
1018
          INSERT INTO Venu (venue_name, address)
1019
1020
          ('Grand Cinema', '123 Main Street'),
1021
          ('Stadium Arena', '456 Sports Complex'),
          ('Music Hall', '789 Concert Avenue'),
1022
          ('Comedy Club', '1011 Laugh Street'),
1023
          ('Art Gallery', '1213 Exhibit Lane'),
1024
1025
          ('Theater Center', '1415 Stage Road'),
1026
          ('Opera House', '1617 Aria Drive'),
          ('Dance Studio', '1819 Motion Place'),
1027
          ('Conference Hall', '2021 Meeting Blvd'),
1028
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
             Grand Cinema
                           123 Main Street
    2
             Stadium Arena 456 Sports Complex
             Music Hall
                           789 Concert Avenue
    3
    4
             Comedy Club
                           1011 Laugh Street
                           1213 Exhibit Lane
    5
             Art Gallery
    6
             Theater Center 1415 Stage Road
    7
             Opera House
                           1617 Aria Drive
    8
                           1819 Motion Place
            Dance Studio
    9
             Conference Hall
                           2021 Meeting Blvd
    10
             Bookstore Cafe
                           2223 Read Avenue
   NULL
.
1032 •
        INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type)
1033
1034
        ('Movie Marathon', '2023-12-20', '18:00:00', 1, 500, 500, 1500, 'Movie'),
        ('Rock Concert', '2023-12-25', '20:00:00', 2, 10000, 7000, 2500, 'Concert'),
       ('Football Match', '2023-12-30', '16:00:00', 3, 15000, 10000, 500, 'Sports'),
1036
        ('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'),
       ('Book Reading', '2023-12-27', '14:00:00', 10, 100, 80, 500, 'Movie'),
1039
        ('Ballet Performance', '2023-12-29', '18:00:00', 6, 500, 450, 2000, 'Concert'),
1040
       ('Opera Show', '2024-01-02', '19:30:00', 7, 800, 720, 3000, 'Concert'),
1041
        ('Music Festival', '2024-01-05', '14:00:00', 2, 20000, 15000, 1800, 'Concert'),
        ('Dance Competition', '2024-01-10', '17:00:00', 8, 1200, 1000, 1500, 'Sports');
1043
1047 •
            INSERT INTO Customer (customer_name, email, phone_number)
1048
            VALUES
            ('John Doe', 'john.doe@example.com', '+1234567890'),
1049
            ('Jane Doe', 'jane.doe@example.com', '+9876543210'),
1050
            ('Alice Smith', 'alice.smith@email.com', '+5551234567'),
1051
            ('Bob Johnson', 'bob.johnson@mail.com', '+3337654321'),
1052
            ('Mary Taylor', 'mary.taylor@web.net', '+4448765432'),
1053
            ('David Miller', 'david.miller@domain.com', '+2229876543'),
1054
            ('Sarah Williams', 'sarah.williams@yahoo.com', '+6663214567'),
1055
            ('Michael Brown', 'michael.brown@gmail.com', '+7775432109'),
1056
            ('Olivia Jones', 'olivia.jones@outlook.com', '+8881234567'),
1057
            ('Emily Garcia', 'emily.garcia@hotmail.com', '+9997654321');
1058
1050
```

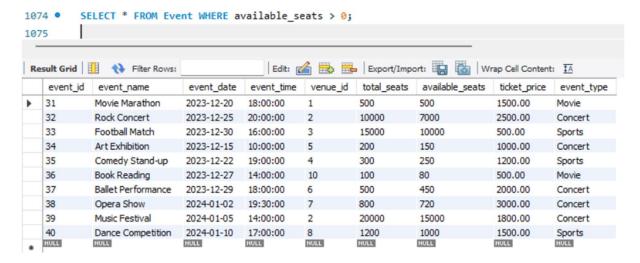
```
1060 •
         INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date)
1061
         VALUES
1062
         (1, 31, 2, 3000, '2023-12-19'),
         (2, 32, 1, 2500, '2023-12-22'),
1063
         (3, 33, 1, 1800, '2023-12-25'),
1064
         (4, 34, 3, 4500, '2023-12-20'),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

2. Write a SQL query to list all Events.

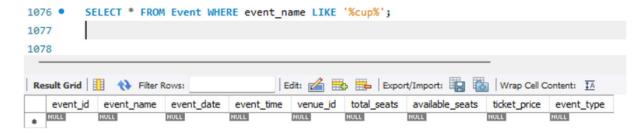
```
1071 • SELECT * FROM Event;
```



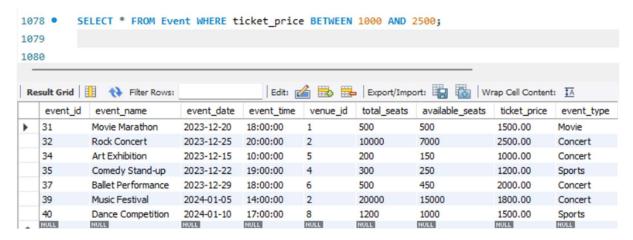
3. Write a SQL query to select events with available tickets.



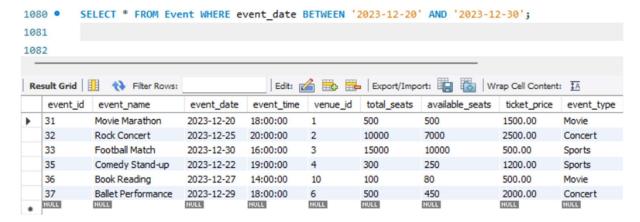
4. Write a SQL query to select events name partial match with 'cup'.



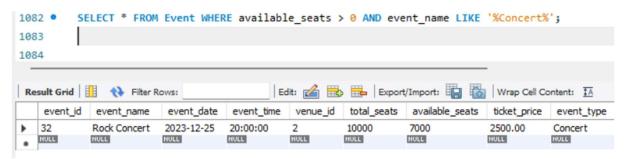
5. Write a SQL query to select events with ticket price range is between 1000 to 2500.



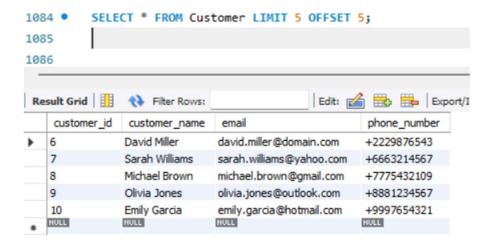
6. Write a SQL query to retrieve events with dates falling within a specific range.



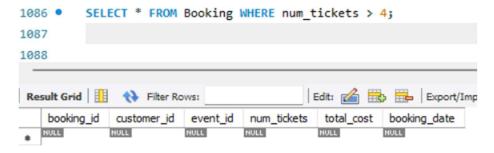
7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.



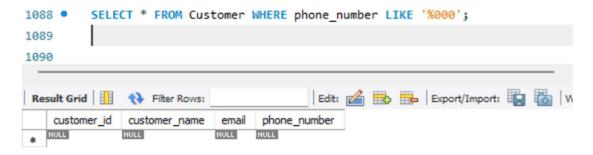
8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.



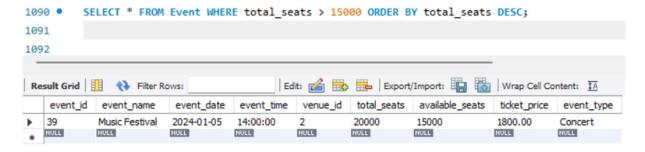
9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.



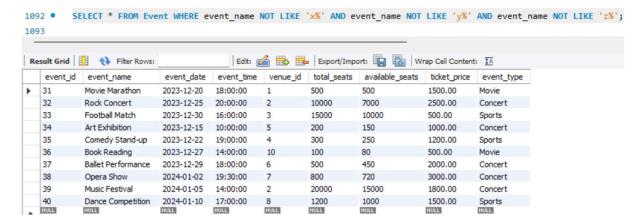
10. Write a SQL query to retrieve customer information whose phone number end with '000'



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

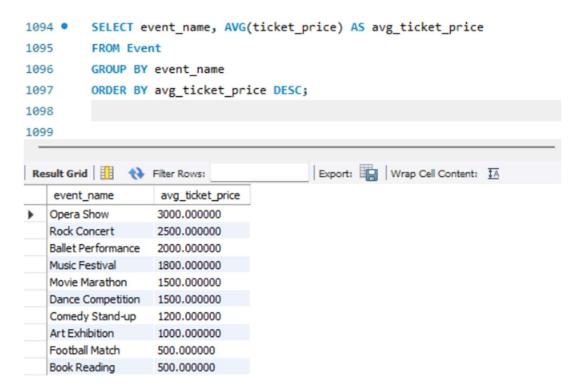


12. Write a SQL query to select events name not start with 'x', 'y', 'z'



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

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

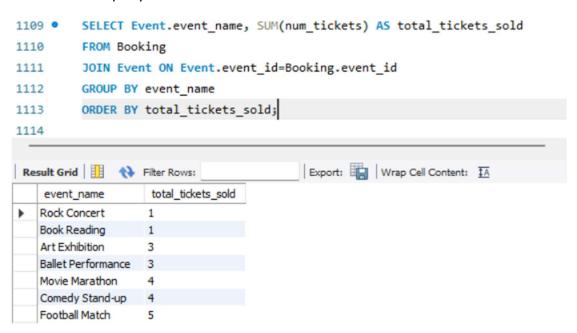


2. Write a SQL query to Calculate the Total Revenue Generated by Events.

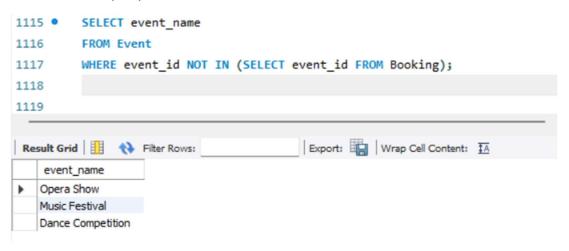
3. Write a SQL query to find the event with the highest ticket sales.

```
1102 •
         SELECT Event.event_name, SUM(num_tickets) AS total_tickets_sold
1103
         FROM Booking
         JOIN Event ON Event.event_id=Booking.event_id
1104
1105
        GROUP BY event name
        ORDER BY total_tickets_sold DESC
1106
1107
        LIMIT 1;
1108
                                       Export: Wrap Cell Content: A Fetch rows
event_name
               total_tickets_sold
Football Match
```

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



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



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
        INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
1121
1122
        GROUP BY customer_name
1123
        ORDER BY total_tickets_booked DESC
1124
        LIMIT 1;
1125
1126
                                       Export: Wrap Cell Content: TA Fetch row
customer_name total_tickets_booked
David Miller
```

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

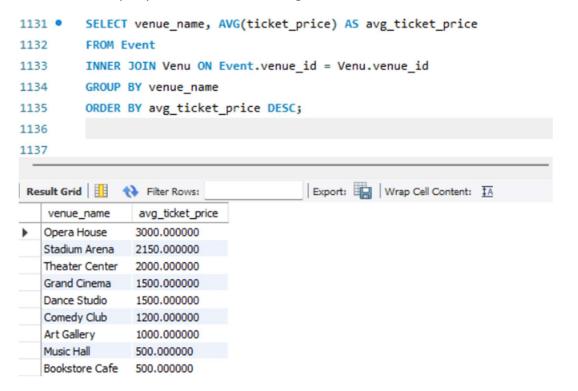
```
1126 • SELECT MONTHNAME(booking_date) AS month, SUM(num_tickets) AS total_tickets_sold

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.



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

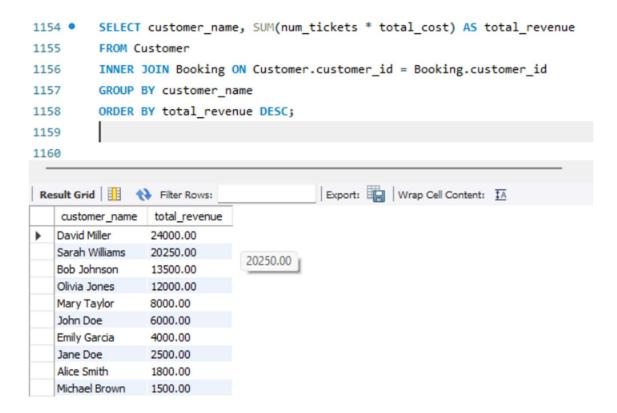
```
1137 •
         SELECT event_type, SUM(num_tickets) AS total_tickets_sold
1138
         FROM Booking
         INNER JOIN Event ON Booking.event_id = Event.event_id
1139
1140
         GROUP BY event_type
         ORDER BY total_tickets_sold DESC;
1141
1142
1143
                                       Export: Wrap Cell Content: TA
event_type total_tickets_sold
   Sports
   Concert
             7
   Movie
             5
```

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

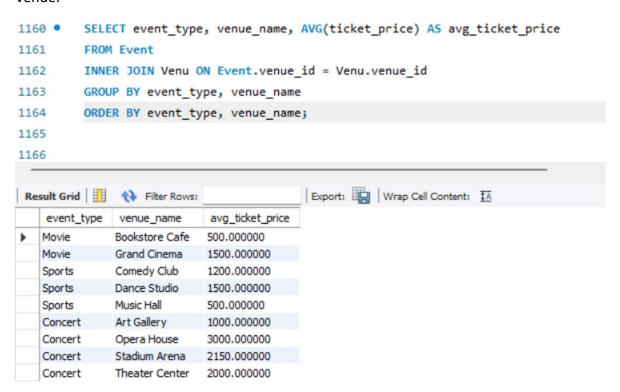
```
1143 •
        SELECT YEAR(booking_date) AS year, SUM(num_tickets * total_cost) AS total_revenue
1144
        FROM Booking
1145
        GROUP BY YEAR(booking_date)
1146
        ORDER BY year;
1147
Export: Wrap Cell Content: IA
   year
        total_revenue
        41300.00
  2023
  2024 52250.00
```

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

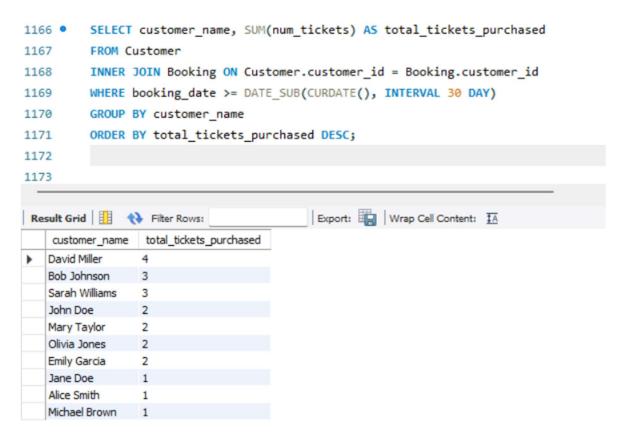
12. Write a SQL query to calculate the Total Revenue Generated by Events for Each User.



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

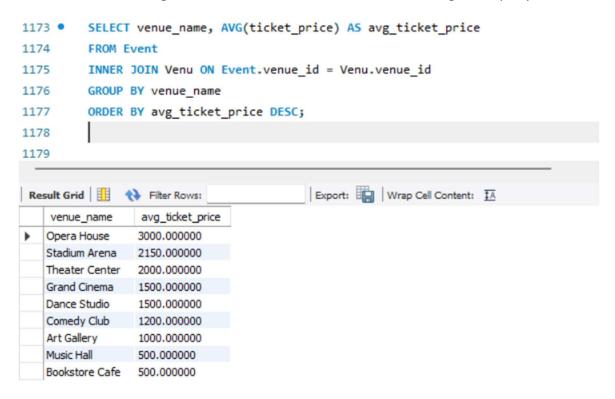


14. Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.



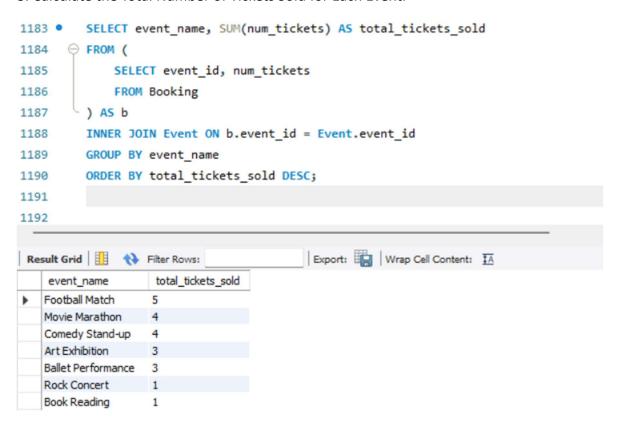
Tasks 4: Subquery and its types

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



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

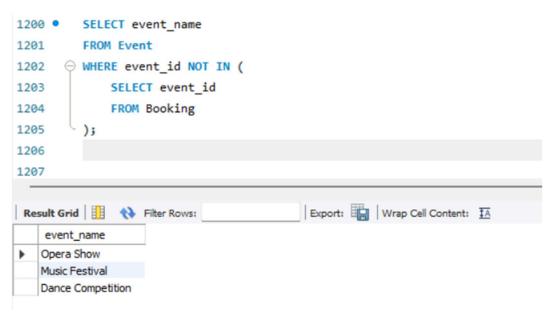
3. Calculate the Total Number of Tickets Sold for Each Event.



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

```
1192 •
       SELECT customer_name
1193
       FROM Customer
SELECT *
1195
1196
          FROM Booking
          WHERE customer_id = Customer.customer_id
1197
1198
      );
1199
1200
                                Export: Wrap Cell
customer_name
```

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

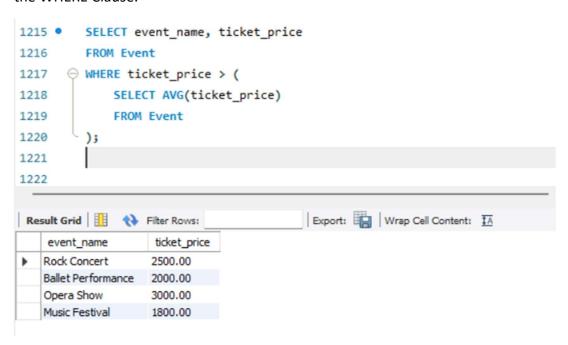


6. Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause.

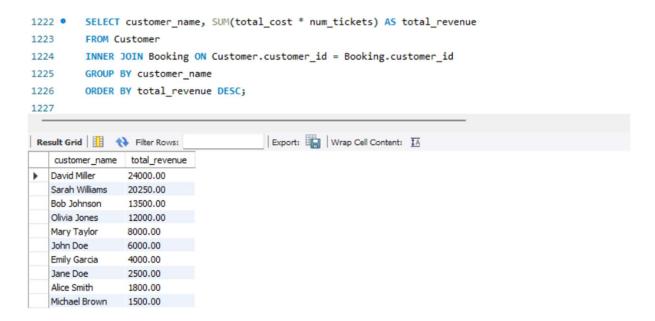
```
SELECT event_type, SUM(num_tickets) AS total_tickets_sold
1206 •
1207

⊖ FROM (
             SELECT event_id, num_tickets
1208
1209
            FROM Booking
       ) AS b
1210
         INNER JOIN Event ON b.event_id = Event.event_id
1211
1212
         GROUP BY event_type
1213
         ORDER BY total_tickets_sold DESC;
1214
1215
                                       Export: Wrap Cell Content: IA
event_type total_tickets_sold
  Sports
             7
   Concert
   Movie
             5
```

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



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



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

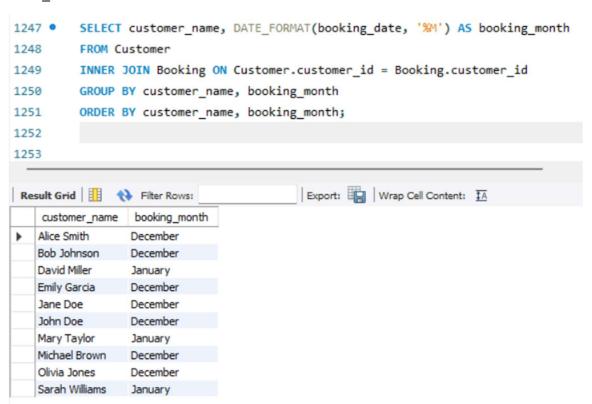
```
1228 •
        SELECT customer_name
1229
        FROM Customer
1230
        INNER JOIN Booking ON Customer.customer_id = Booking.customer_id
        INNER JOIN Event ON Booking.event_id = Event.event_id
1231
SELECT venue id
1233
1234
           FROM Venu
1235
           WHERE venue name = 'Grand Cinema'
1236
       );
1237
1238
                                    Export: Wrap Cell Content: TA
customer_name
   John Doe
   Mary Taylor
```

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

```
SELECT event_type, SUM(num_tickets) AS total_tickets_sold

⊖ FROM (
1239
1240
             SELECT Booking.event_id, num_tickets, event_type
             FROM Booking
1241
1242
             INNER JOIN Event ON Booking.event_id = Event.event_id
1243
       ) AS b
         GROUP BY event_type
1244
1245
         ORDER BY total_tickets_sold DESC;
1246
1247
1248
                                        Export: Wrap Cell Content: TA
event_type total_tickets_sold
   Sports
             7
   Concert
   Movie
             5
```

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



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

Krushnakumar Patle

```
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
         ORDER BY avg_ticket_price DESC;
1257
1258
1259
                                        Export: Wrap Cell Content: IA
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
                            1000.000000
   Music Hall 500.000000
   Bookstore Cafe 500.000000
```