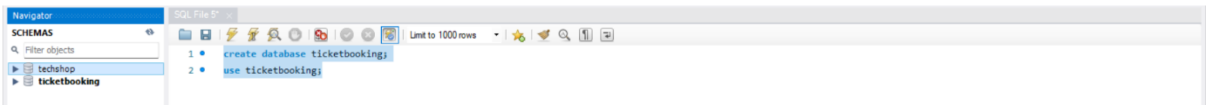
Name: R. Surya prakash

Assignment 5

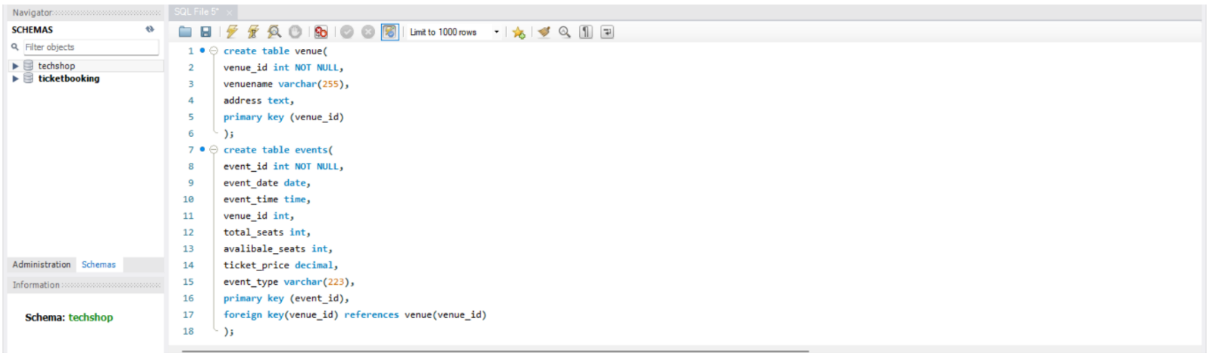
Tasks 1: Database Design:

1. Create the database named "Ticket Booking System".

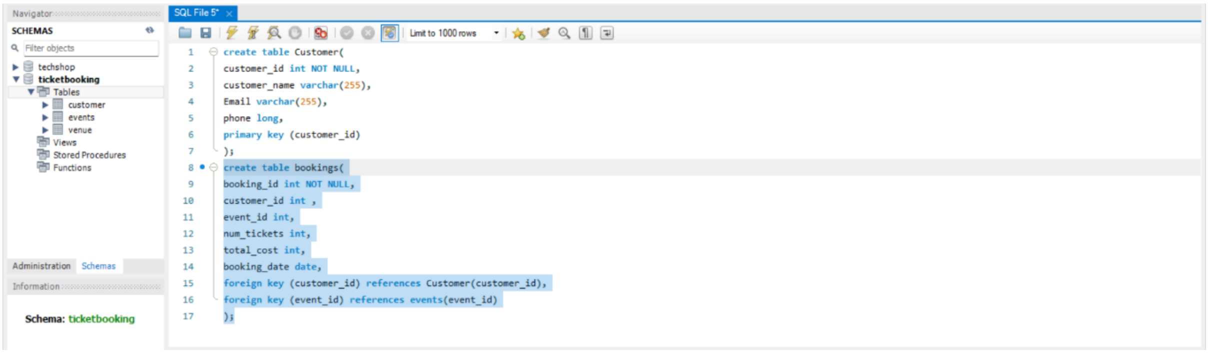


```sql
create database ticketbooking;
use ticketbooking;
```

2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
   1.Venue 2. Event 3. Customers 4. Booking



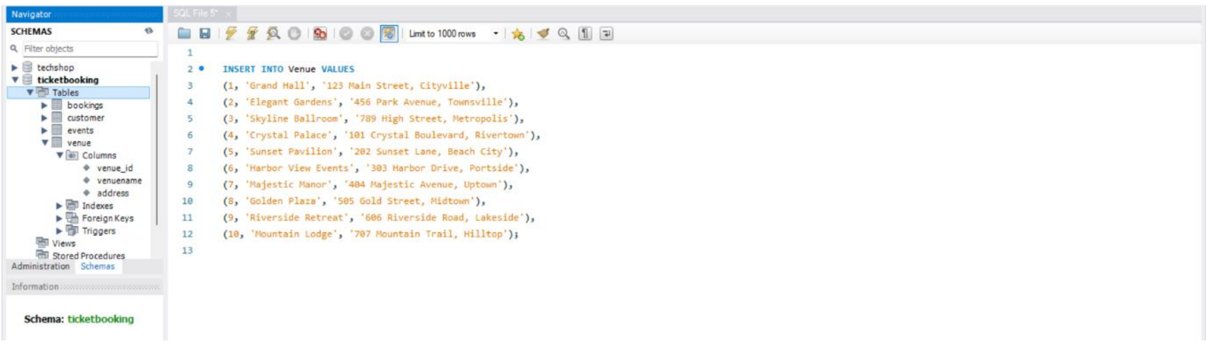
```sql
create table venue(
venue_id int NOT NULL,
venuename varchar(255),
address text,
primary key (venue_id)
);
create table events(
event_id int NOT NULL,
event_date date,
event_time time,
venue_id int,
total_seats int,
avalibale_seats int,
ticket_price decimal,
event_type varchar(223),
primary key (event_id),
foreign key(venue_id) references venue(venue_id)
);
```



```sql
create table Customer(
customer_id int NOT NULL,
customer_name varchar(255),
Email varchar(255),
phone long,
primary key (customer_id)
);
create table bookings(
booking_id int NOT NULL,
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 events(event_id)
);
```

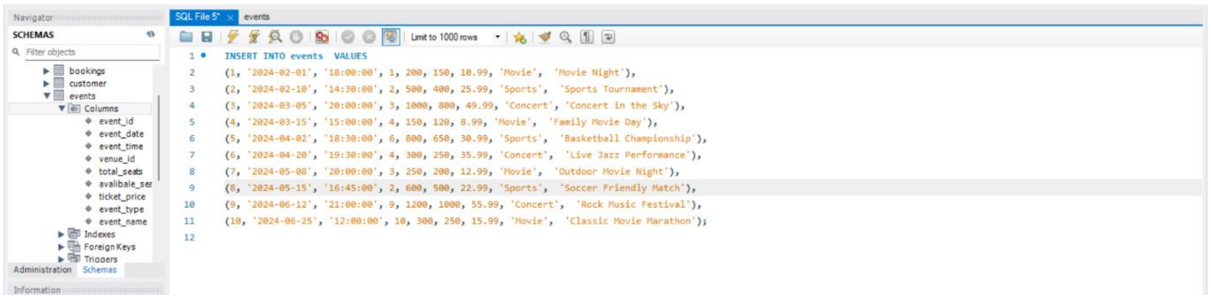
Tasks 2: Select, Where, Between, AND, LIKE:

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



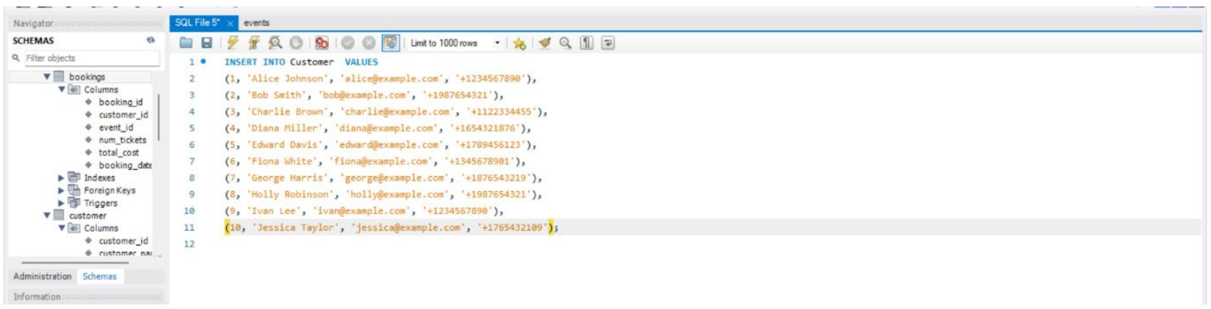
```sql
INSERT INTO Venue VALUES
(1, 'Grand Hall', '123 Main Street, Cityville'),
(2, 'Elegant Gardens', '456 Park Avenue, Townsville'),
(3, 'Skyline Ballroom', '789 High Street, Metropolis'),
(4, 'Crystal Palace', '101 Crystal Boulevard, Rivertown'),
(5, 'Sunset Pavilion', '202 Sunset Lane, Beach City'),
(6, 'Harbor View Events', '303 Harbor Drive, Portside'),
(7, 'Majestic Manor', '404 Majestic Avenue, Uptown'),
(8, 'Golden Plaza', '505 Gold Street, Midtown'),
(9, 'Riverside Retreat', '606 Riverside Road, Lakeside'),
(10, 'Mountain Lodge', '707 Mountain Trail, Hilltop');
```

2.events



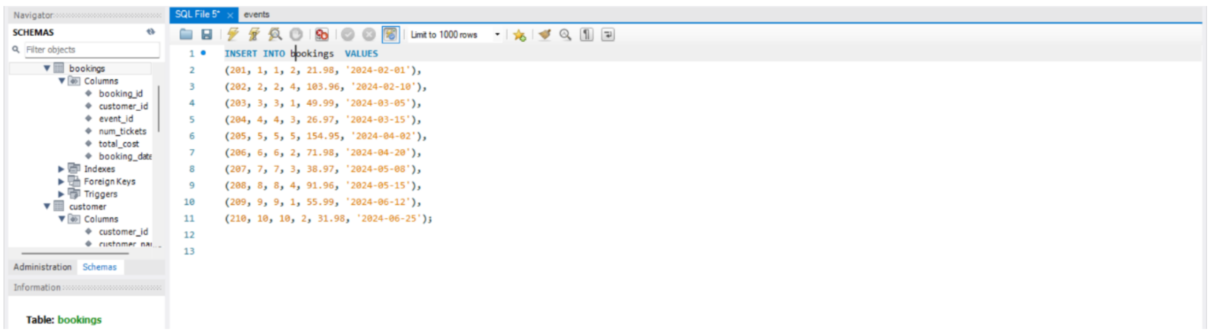
```sql
INSERT INTO events  VALUES
(1, '2024-02-01', '18:00:00', 1, 200, 150, 10.99, 'Movie',  'Movie Night'),
(2, '2024-02-10', '14:30:00', 2, 500, 400, 25.99, 'Sports',  'Sports Tournament'),
(3, '2024-03-05', '20:00:00', 3, 1000, 800, 49.99, 'Concert', 'Concert in the Sky'),
(4, '2024-03-15', '15:00:00', 4, 150, 120, 8.99, 'Movie',  'Family Movie Day'),
(5, '2024-04-02', '18:30:00', 6, 800, 650, 30.99, 'Sports',  'Basketball Championship'),
(6, '2024-04-20', '19:30:00', 4, 300, 250, 35.99, 'Concert',  'Live Jazz Performance'),
(7, '2024-05-08', '20:00:00', 3, 250, 200, 12.99, 'Movie',  'Outdoor Movie Night'),
(8, '2024-05-15', '16:45:00', 2, 600, 500, 22.99, 'Sports',  'Soccer Friendly Match'),
(9, '2024-06-12', '21:00:00', 9, 1200, 1000, 55.99, 'Concert',  'Rock Music Festival'),
(10, '2024-06-25', '12:00:00', 10, 300, 250, 15.99, 'Movie',  'Classic Movie Marathon');
```

3.Customer



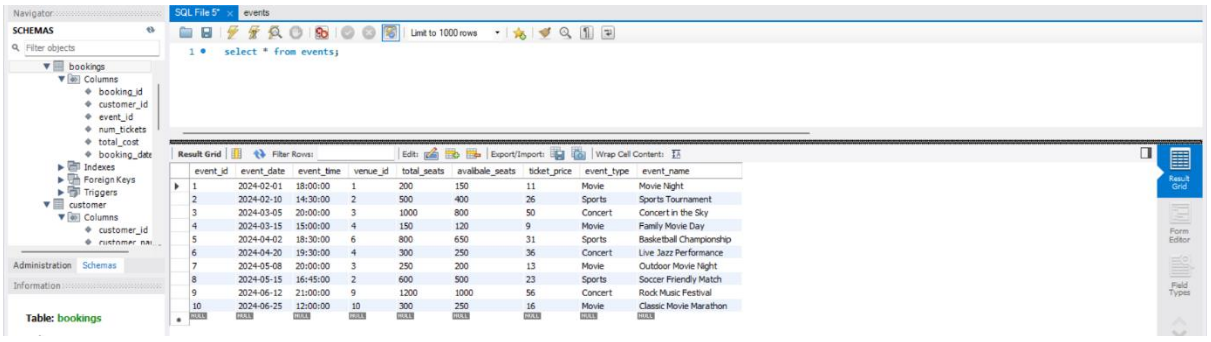
```sql
INSERT INTO Customer  VALUES
(1, 'Alice Johnson', 'alice@example.com', '+1234567890'),
(2, 'Bob Smith', 'bob@example.com', '+1987654321'),
(3, 'Charlie Brown', 'charlie@example.com', '+1122334455'),
(4, 'Diana Miller', 'diana@example.com', '+1654321876'),
(5, 'Edward Davis', 'edward@example.com', '+1789456123'),
(6, 'Fiona White', 'fiona@example.com', '+1345678901'),
(7, 'George Harris', 'george@example.com', '+1876543219'),
(8, 'Holly Robinson', 'holly@example.com', '+1987654321'),
(9, 'Ivan Lee', 'ivan@example.com', '+1234567890'),
(10, 'Jessica Taylor', 'jessica@example.com', '+1765432109');
```

4.Bookings



```sql
INSERT INTO bookings  VALUES
(201, 1, 1, 2, 21.98, '2024-02-01'),
(202, 2, 2, 4, 103.96, '2024-02-10'),
(203, 3, 3, 1, 49.99, '2024-03-05'),
(204, 4, 4, 3, 26.97, '2024-03-15'),
(205, 5, 5, 5, 154.95, '2024-04-02'),
(206, 6, 6, 2, 71.98, '2024-04-20'),
(207, 7, 7, 3, 38.97, '2024-05-08'),
(208, 8, 8, 4, 91.96, '2024-05-15'),
(209, 9, 9, 1, 55.99, '2024-06-12'),
(210, 10, 10, 2, 31.98, '2024-06-25');
```

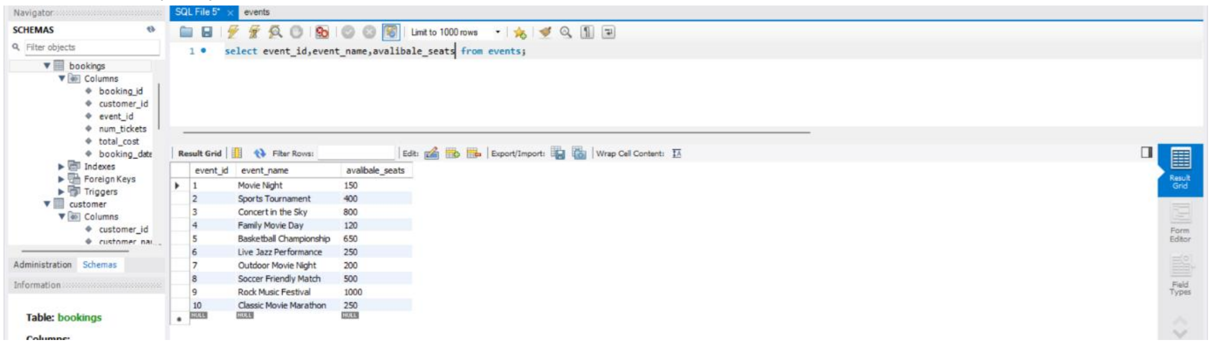
2. Write a SQL query to list all Events.



```sql
select * from events;
```

event_id	event_date	event_time	venue_id	total_seats	avalibale_seats	ticket_price	event_type	event_name
1	2024-02-01	18:00:00	1	200	150	11	Movie	Movie Night
2	2024-02-10	14:30:00	2	500	400	26	Sports	Sports Tournament
3	2024-03-05	20:00:00	3	1000	800	50	Concert	Concert in the Sky
4	2024-03-15	15:00:00	4	150	120	9	Movie	Family Movie Day
5	2024-04-02	18:30:00	6	800	650	31	Sports	Basketball Championship
6	2024-04-20	19:30:00	4	300	250	36	Concert	Live Jazz Performance
7	2024-05-08	20:00:00	3	250	200	13	Movie	Outdoor Movie Night
8	2024-05-15	16:45:00	2	600	500	23	Sports	Soccer Friendly Match
9	2024-06-12	21:00:00	9	1200	1000	56	Concert	Rock Music Festival
10	2024-06-25	12:00:00	10	300	250	16	Movie	Classic Movie Marathon
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

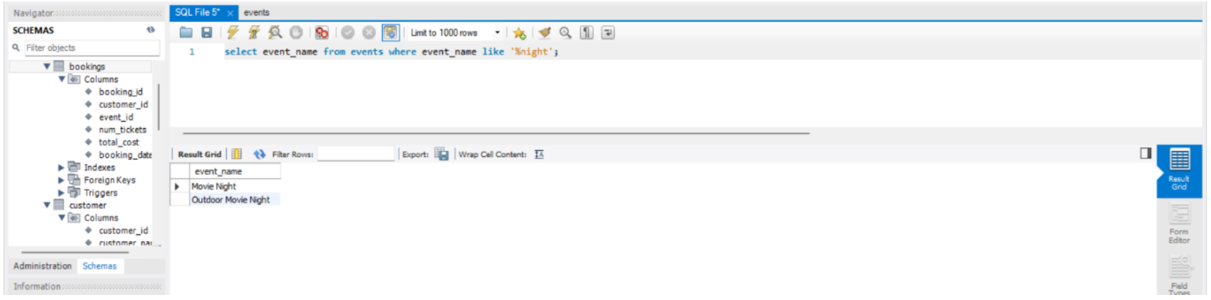
3. Write a SQL query to select events with available tickets.



```sql
select event_id,event_name,avalibale_seats from events;
```

event_id	event_name	avalibale_seats
1	Movie Night	150
2	Sports Tournament	400
3	Concert in the Sky	800
4	Family Movie Day	120
5	Basketball Championship	650
6	Live Jazz Performance	250
7	Outdoor Movie Night	200
8	Soccer Friendly Match	500
9	Rock Music Festival	1000
10	Classic Movie Marathon	250
NULL	NULL	NULL

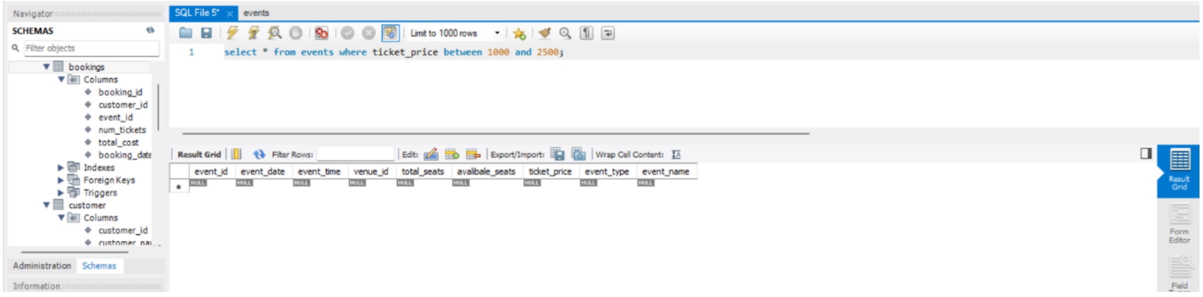
4. Write a SQL query to select events name partial match with 'cup'.



```sql
select event_name from events where event_name like '%night';
```

event_name
Movie Night
Outdoor Movie Night

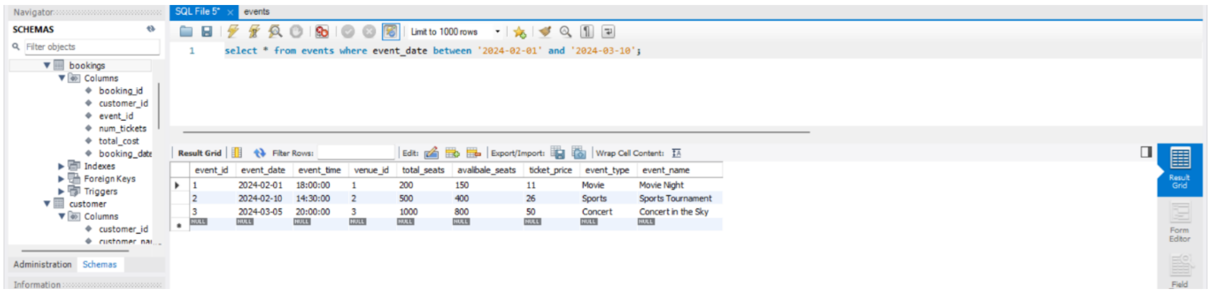
5.Write a SQL query to select events with ticket price range is between 1000 to 2500.



```sql
select * from events where ticket_price between 1000 and 2500;
```

event_id	event_date	event_time	venue_id	total_seats	avalibale_seats	ticket_price	event_type	event_name
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

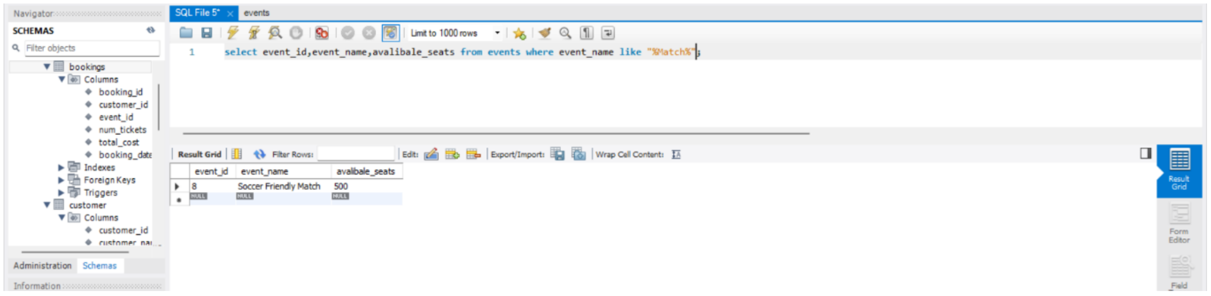
6. Write a SQL query to retrieve events with dates falling within a specific range.



```sql
select * from events where event_date between '2024-02-01' and '2024-03-10';
```

event_id	event_date	event_time	venue_id	total_seats	avalibale_seats	ticket_price	event_type	event_name
1	2024-02-01	18:00:00	1	200	150	11	Movie	Movie Night
2	2024-02-10	14:30:00	2	500	400	26	Sports	Sports Tournament
3	2024-03-05	20:00:00	3	1000	800	50	Concert	Concert in the Sky
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

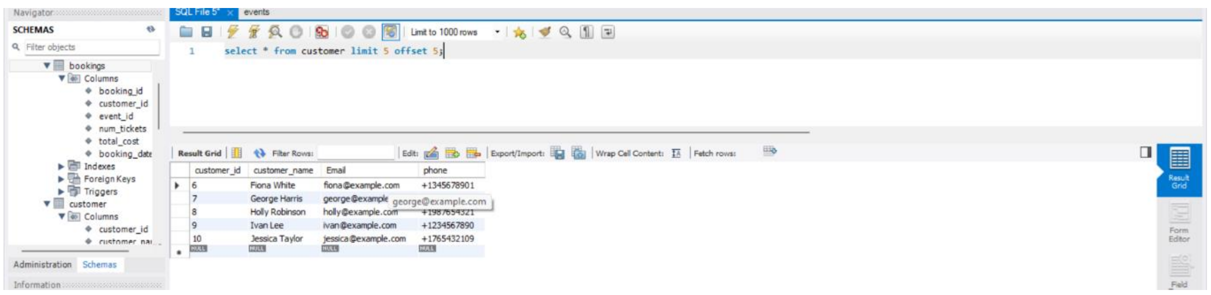
7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.



```sql
select event_id,event_name,avalibale_seats from events where event_name like "%Match%";
```

event_id	event_name	avalibale_seats
8	Soccer Friendly Match	500
NULL	NULL	NULL

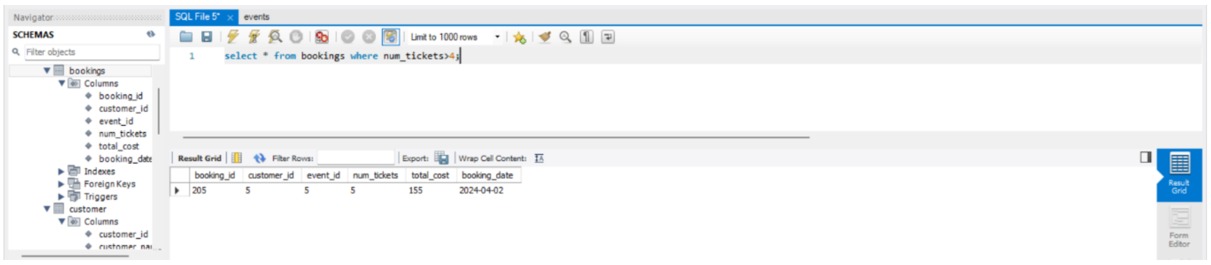
8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.



```sql
select * from customer limit 5 offset 5;
```

customer_id	customer_name	Email	phone
6	Fiona White	fiona@example.com	+1345678901
7	George Harris	george@example.com	[illegible]
8	Holly Robinson	holly@example.com	+1987654321
9	Ivan Lee	ivan@example.com	+1234567890
10	Jessica Taylor	jessica@example.com	+1765432109
NULL	NULL	NULL	NULL

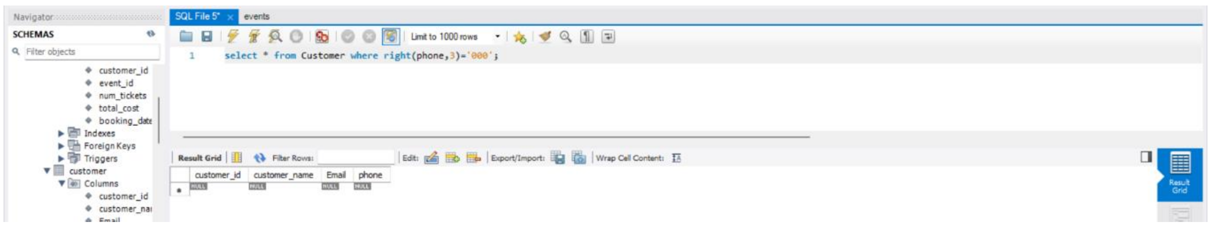
9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.



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

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
205	5	5	5	155	2024-04-02

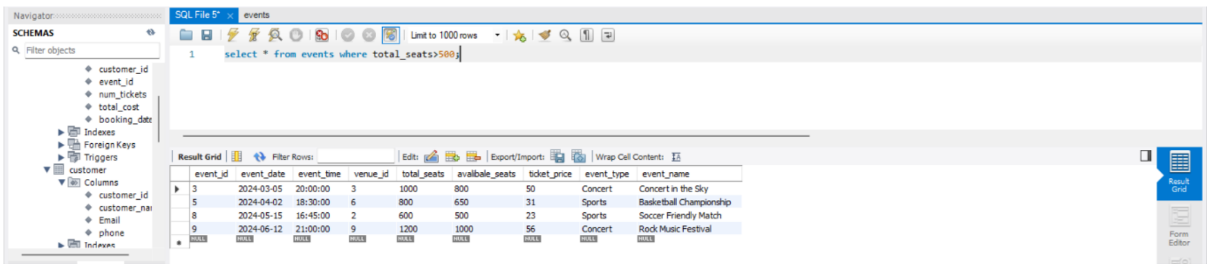
10. Write a SQL query to retrieve customer information whose phone number end with ‘000’



```
select * from Customer where right(phone,3)='000';
```

customer_id	customer_name	Email	phone
NULL	NULL	NULL	NULL

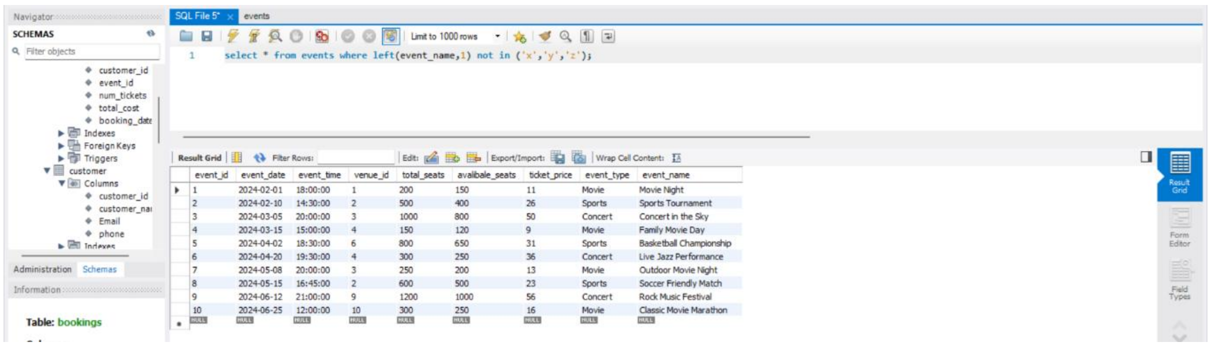
11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.



```
select * from events where total_seats>500;
```

event_id	event_date	event_time	venue_id	total_seats	avalibale_seats	ticket_price	event_type	event_name
3	2024-03-05	20:00:00	3	1000	800	50	Concert	Concert in the Sky
5	2024-04-02	18:30:00	6	800	650	31	Sports	Basketball Championship
8	2024-05-15	16:45:00	2	600	500	23	Sports	Soccer Friendly Match
9	2024-06-12	21:00:00	9	1200	1000	56	Concert	Rock Music Festival
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

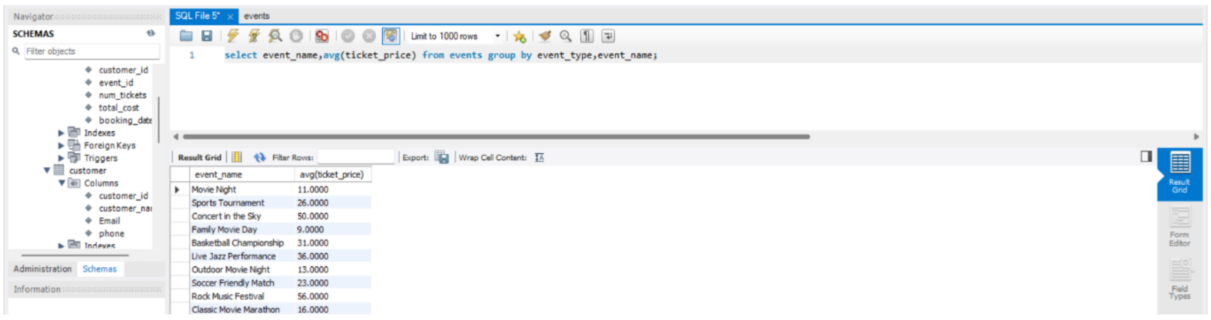


```
select * from events where left(event_name,1) not in ('x','y','z');
```

event_id	event_date	event_time	venue_id	total_seats	avalibale_seats	ticket_price	event_type	event_name
1	2024-02-01	18:00:00	1	200	150	11	Movie	Movie Night
2	2024-02-10	14:30:00	2	500	400	26	Sports	Sports Tournament
3	2024-03-05	20:00:00	3	1000	800	50	Concert	Concert in the Sky
4	2024-03-15	15:00:00	4	150	120	9	Movie	Family Movie Day
5	2024-04-02	18:30:00	6	800	650	31	Sports	Basketball Championship
6	2024-04-20	19:30:00	4	300	250	36	Concert	Live Jazz Performance
7	2024-05-08	20:00:00	3	250	200	13	Movie	Outdoor Movie Night
8	2024-05-15	16:45:00	2	600	500	23	Sports	Soccer Friendly Match
9	2024-06-12	21:00:00	9	1200	1000	56	Concert	Rock Music Festival
10	2024-06-25	12:00:00	10	300	250	16	Movie	Classic Movie Marathon
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

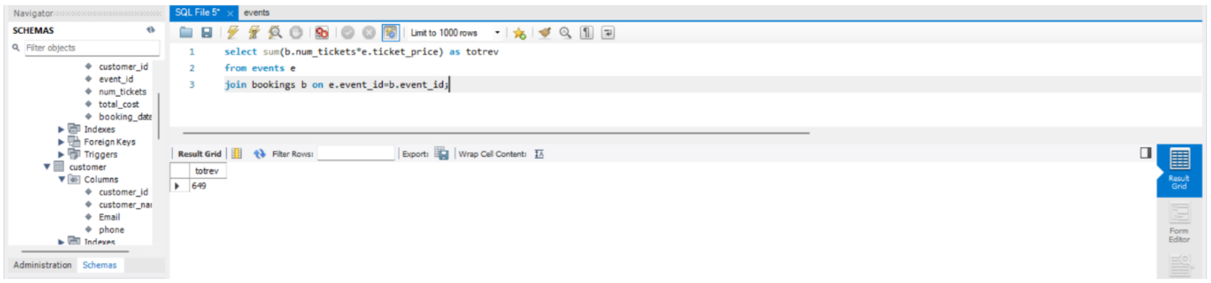
1. Write a SQL query to List Events and Their Average Ticket Prices.



```
select event_name,avg(ticket_price) from events group by event_type,event_name;
```

```
event_name                avg(ticket_price)
Movie Night               11.0000
Sports Tournament         26.0000
Concert in the Sky        50.0000
Family Movie Day          9.0000
Basketball Championship   31.0000
Live Jazz Performance     36.0000
Outdoor Movie Night       13.0000
Soccer Friendly Match     23.0000
Rock Music Festival       56.0000
Classic Movie Marathon    16.0000
```

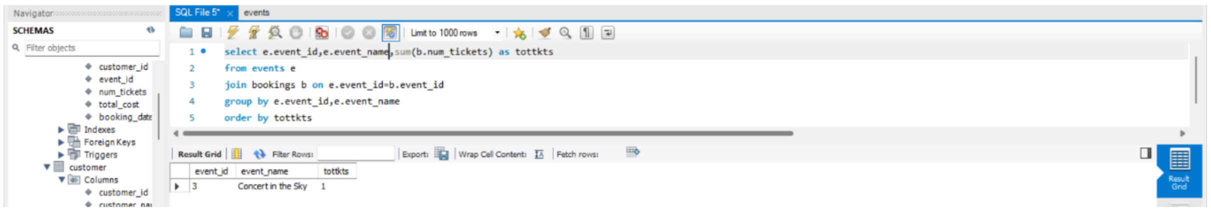
2. Write a SQL query to Calculate the Total Revenue Generated by Events.



```
select sum(b.num_tickets*e.ticket_price) as totrev
from events e
join bookings b on e.event_id=b.event_id;
```

```
totrev
649
```

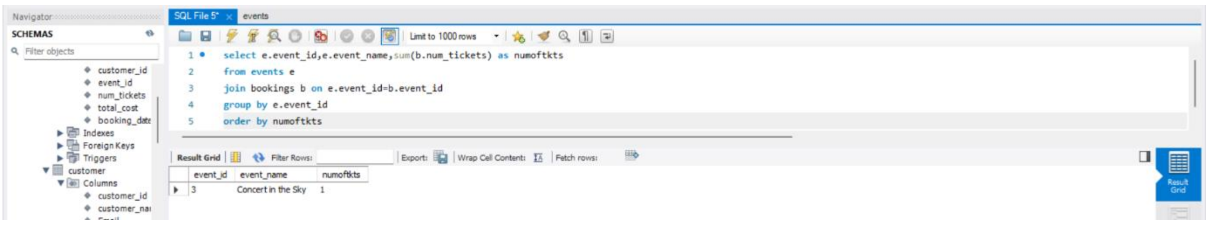
3.Write a SQL query to find the event with the highest ticket sales.



```
select e.event_id,e.event_name,sum(b.num_tickets) as tottkts
from events e
join bookings b on e.event_id=b.event_id
group by e.event_id,e.event_name
order by tottkts
```

```
event_id   event_name           tottkts
3          Concert in the Sky   1
```

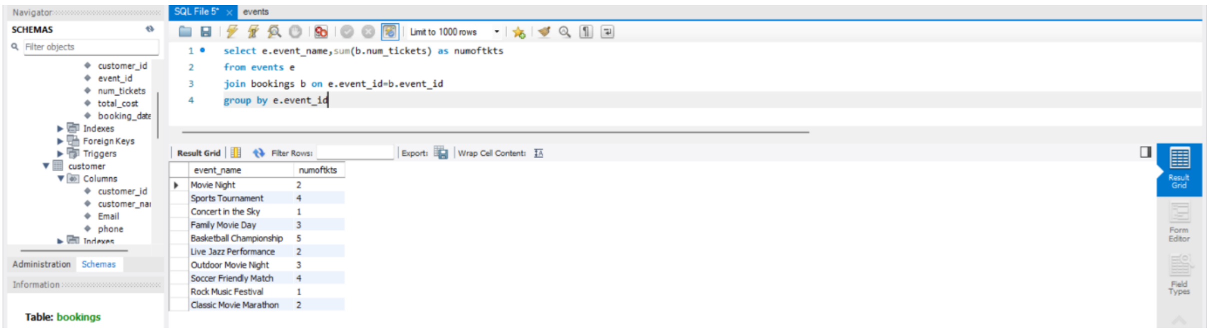
4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.



```sql
select e.event_id,e.event_name,sum(b.num_tickets) as numoftkts
from events e
join bookings b on e.event_id=b.event_id
group by e.event_id
order by numoftkts
```

event_id	event_name	numoftkts
3	Concert in the Sky	1

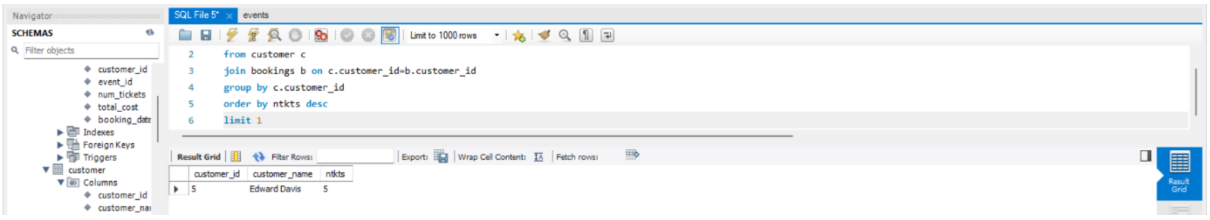
5. Write a SQL query to Find Events with No Ticket Sales.



```sql
select e.event_name,sum(b.num_tickets) as numoftkts
from events e
join bookings b on e.event_id=b.event_id
group by e.event_id
```

event_name	numoftkts
Movie Night	2
Sports Tournament	4
Concert in the Sky	1
Family Movie Day	3
Basketball Championship	5
Live Jazz Performance	2
Outdoor Movie Night	3
Soccer Friendly Match	4
Rock Music Festival	1
Classic Movie Marathon	2

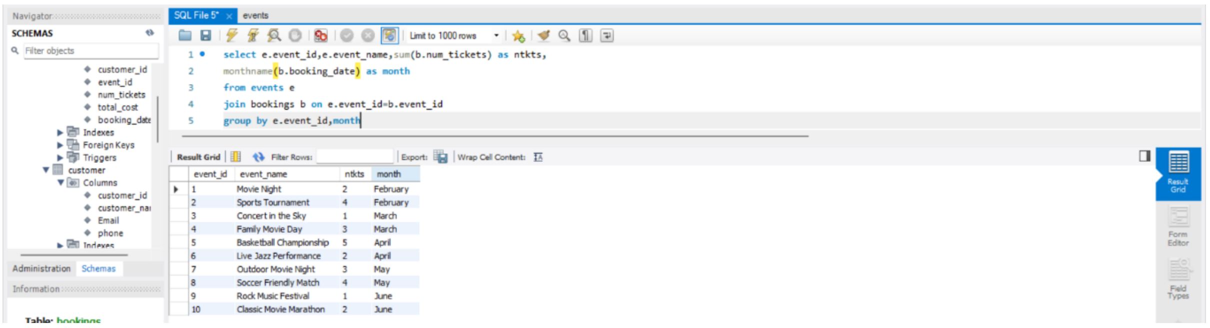
6. Write a SQL query to Find the User Who Has Booked the Most Tickets.



```sql
from customer c
join bookings b on c.customer_id=b.customer_id
group by c.customer_id
order by ntkts desc
limit 1
```

customer_id	customer_name	ntkts
5	Edward Davis	5

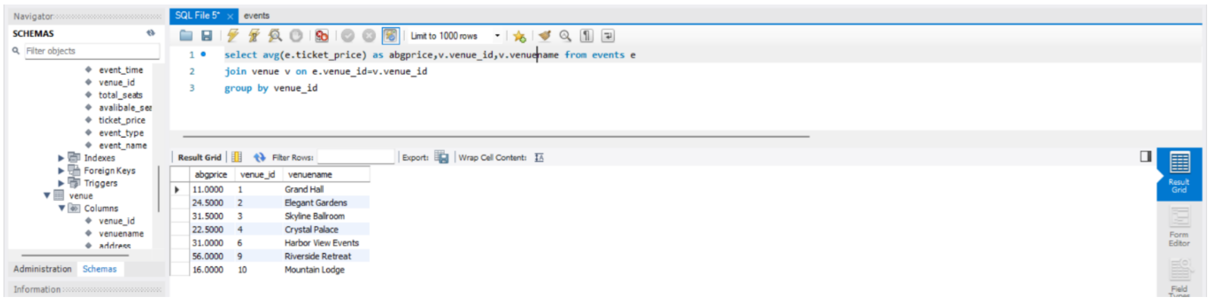
7. Write a SQL query to List Events and the total number of tickets sold for each month.



```sql
select e.event_id,e.event_name,sum(b.num_tickets) as ntkts,
monthname(b.booking_date) as month
from events e
join bookings b on e.event_id=b.event_id
group by e.event_id,month
```

event_id	event_name	ntkts	month
1	Movie Night	2	February
2	Sports Tournament	4	February
3	Concert in the Sky	1	March
4	Family Movie Day	3	March
5	Basketball Championship	5	April
6	Live Jazz Performance	2	April
7	Outdoor Movie Night	3	May
8	Soccer Friendly Match	4	May
9	Rock Music Festival	1	June
10	Classic Movie Marathon	2	June

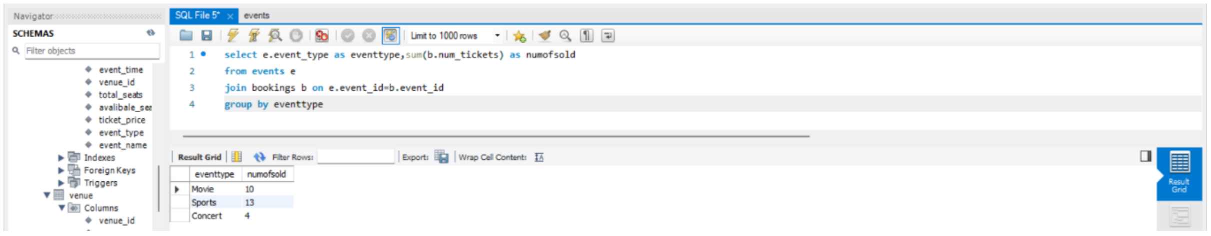
8.Write a SQL query to calculate the average Ticket Price for Events in Each Venue.



```sql
select avg(e.ticket_price) as abgprice,v.venue_id,v.venuename from events e
join venue v on e.venue_id=v.venue_id
group by venue_id
```

abgprice	venue_id	venuename
11.0000	1	Grand Hall
24.5000	2	Elegant Gardens
31.5000	3	Skyline Ballroom
22.5000	4	Crystal Palace
31.0000	6	Harbor View Events
56.0000	9	Riverside Retreat
16.0000	10	Mountain Lodge

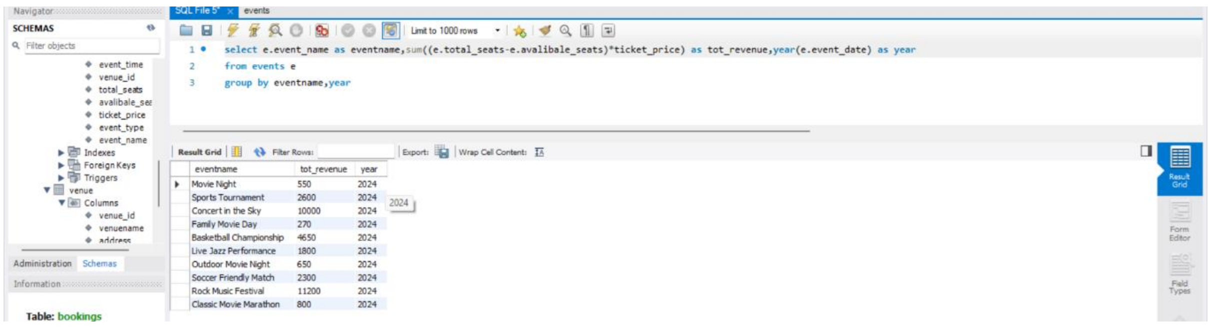
9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.



```sql
select e.event_type as eventtype,sum(b.num_tickets) as numofsold
from events e
join bookings b on e.event_id=b.event_id
group by eventtype
```

eventtype	numofsold
Movie	10
Sports	13
Concert	4

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

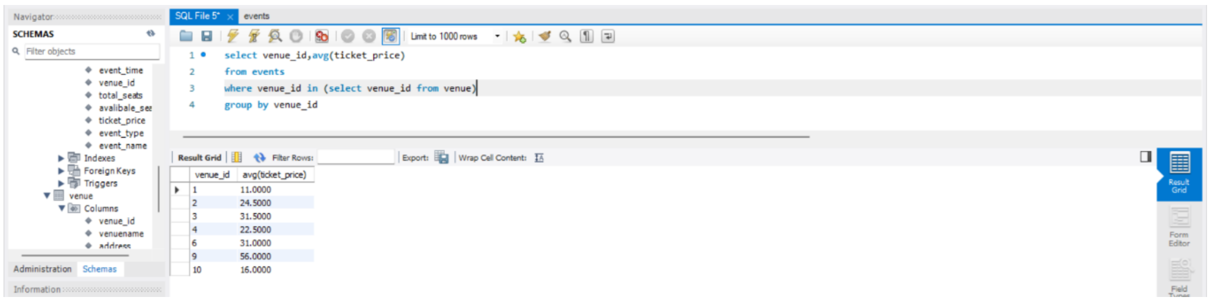


```sql
select e.event_name as eventname,sum((e.total_seats-e.avalibale_seats)*ticket_price) as tot_revenue,year(e.event_date) as year
from events e
group by eventname,year
```

eventname	tot_revenue	year
Movie Night	550	2024
Sports Tournament	2600	2024
Concert in the Sky	10000	2024
Family Movie Day	270	2024
Basketball Championship	4650	2024
Live Jazz Performance	1800	2024
Outdoor Movie Night	650	2024
Soccer Friendly Match	2300	2024
Rock Music Festival	11200	2024
Classic Movie Marathon	800	2024

Tasks 4: Subquery and its types

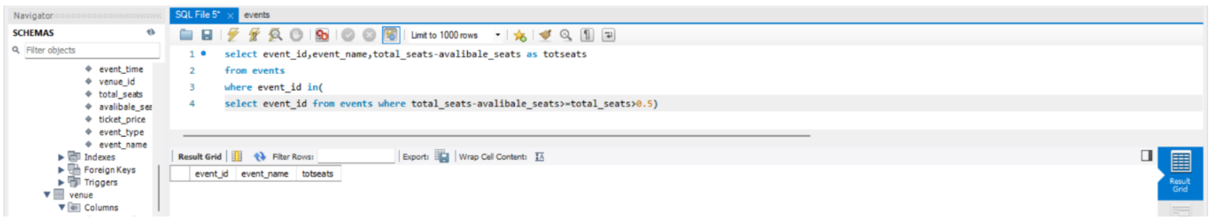
1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.



```sql
select venue_id,avg(ticket_price)
from events
where venue_id in (select venue_id from venue)
group by venue_id
```

```
   venue_id   avg(ticket_price)
▶  1          11.0000
   2          24.5000
   3          31.5000
   4          22.5000
   6          31.0000
   9          56.0000
   10         16.0000
```

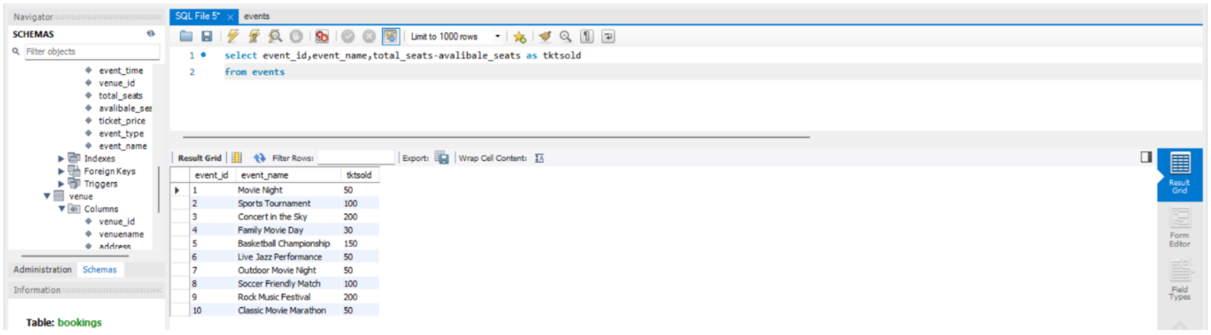
2. Find Events with More Than 50% of Tickets Sold using subquery.



```sql
select event_id,event_name,total_seats-avalibale_seats as totseats
from events
where event_id in(
select event_id from events where total_seats-avalibale_seats>=total_seats>0.5)
```

```
   event_id   event_name   totseats
```

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

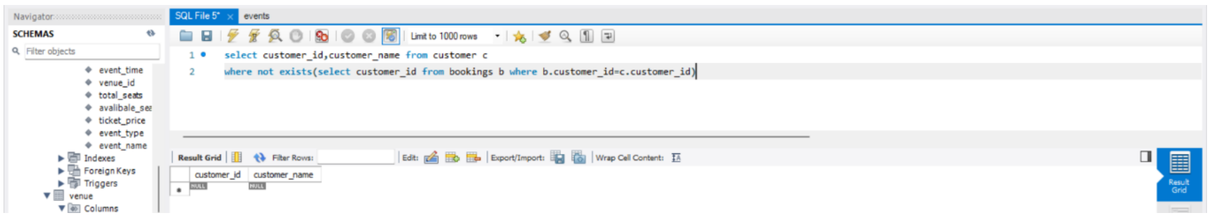


```sql
select event_id,event_name,total_seats-avalibale_seats as tktsold
from events
```

```
   event_id   event_name                tktsold
▶  1          Movie Night               50
   2          Sports Tournament         100
   3          Concert in the Sky        200
   4          Family Movie Day          30
   5          Basketball Championship   150
   6          Live Jazz Performance     50
   7          Outdoor Movie Night       50
   8          Soccer Friendly Match     100
   9          Rock Music Festival       200
   10         Classic Movie Marathon    50
```

Table: bookings

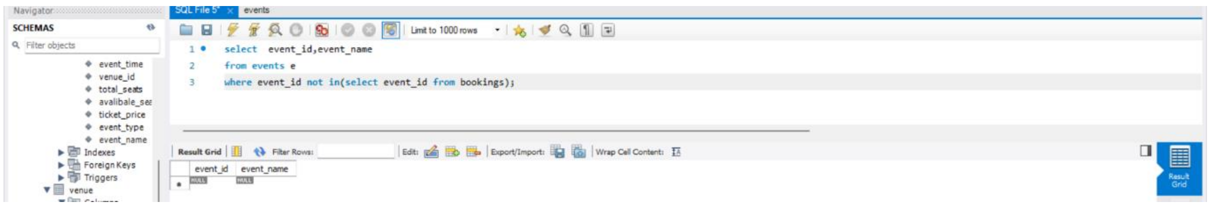
4. Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.



```sql
select customer_id,customer_name from customer c
where not exists(select customer_id from bookings b where b.customer_id=c.customer_id)
```

```
customer_id  customer_name
NULL         NULL
```

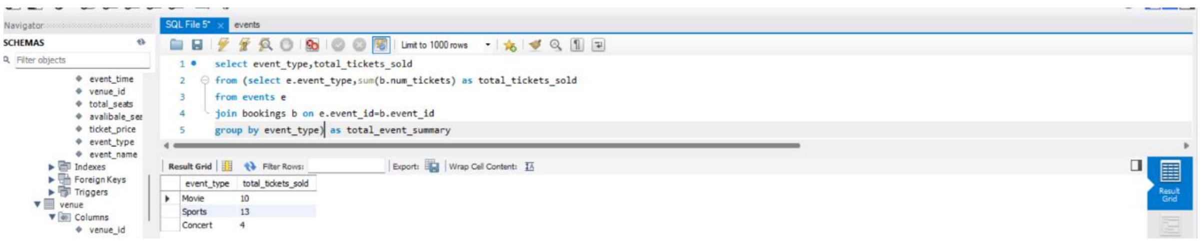
5. List Events with No Ticket Sales Using a NOT IN Subquery.



```sql
select  event_id,event_name
from events e
where event_id not in(select event_id from bookings);
```

```
event_id  event_name
NULL      NULL
```

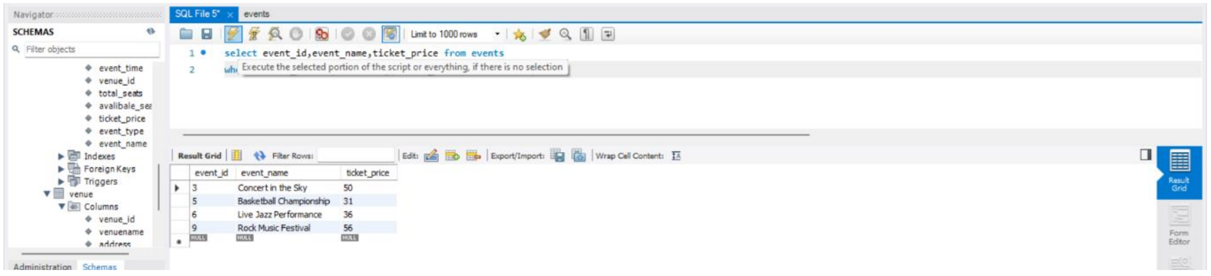
6. Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause.



```sql
select event_type,total_tickets_sold
from (select e.event_type,sum(b.num_tickets) as total_tickets_sold
from events e
join bookings b on e.event_id=b.event_id
group by event_type) as total_event_summary
```

```
event_type  total_tickets_sold
Movie       10
Sports      13
Concert     4
```

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

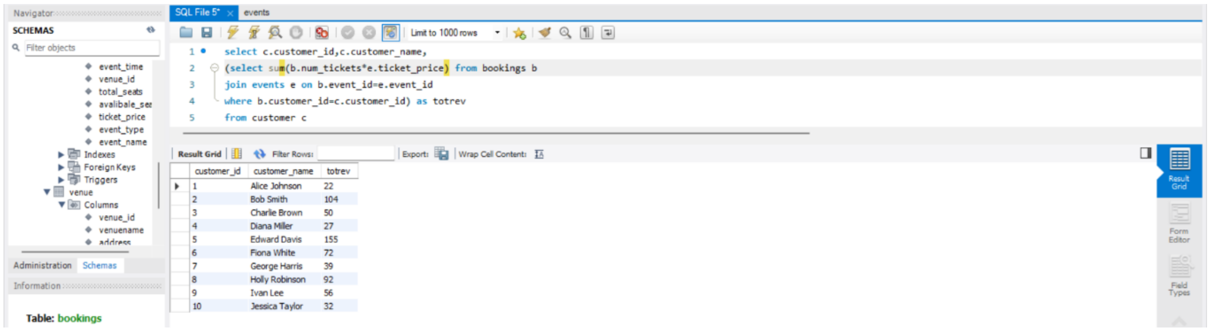


```sql
select event_id,event_name,ticket_price from events
whi
```

Execute the selected portion of the script or everything, if there is no selection

```
event_id  event_name               ticket_price
3         Concert in the Sky       50
5         Basketball Championship  31
6         Live Jazz Performance    36
9         Rock Music Festival      56
NULL      NULL                     NULL
```

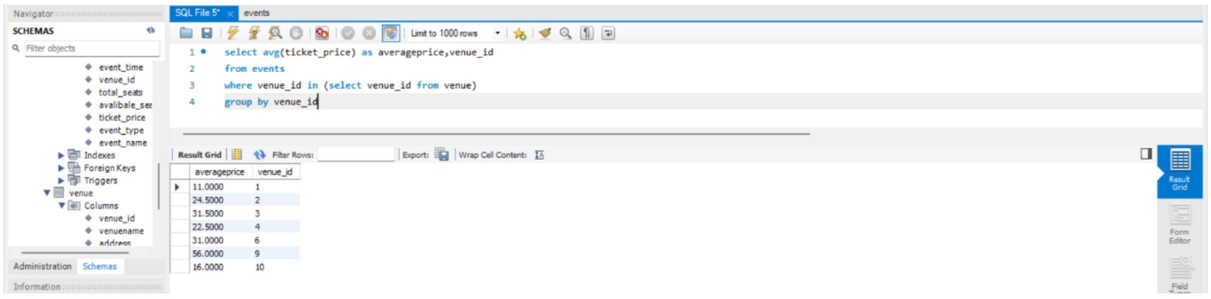
8. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.



```sql
select c.customer_id,c.customer_name,
(select sum(b.num_tickets*e.ticket_price) from bookings b
join events e on b.event_id=e.event_id
where b.customer_id=c.customer_id) as totrev
from customer c
```

```
customer_id  customer_name    totrev
1            Alice Johnson    22
2            Bob Smith        104
3            Charlie Brown    50
4            Diana Miller     27
5            Edward Davis     155
6            Fiona White      72
7            George Harris    39
8            Holly Robinson   92
9            Ivan Lee         56
10           Jessica Taylor   32
```

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



```sql
select avg(ticket_price) as averageprice,venue_id
from events
where venue_id in (select venue_id from venue)
group by venue_id
```

```
averageprice  venue_id
11.0000       1
24.5000       2
31.5000       3
22.5000       4
31.0000       6
56.0000       9
16.0000       10
```