

# QUERY SHEET

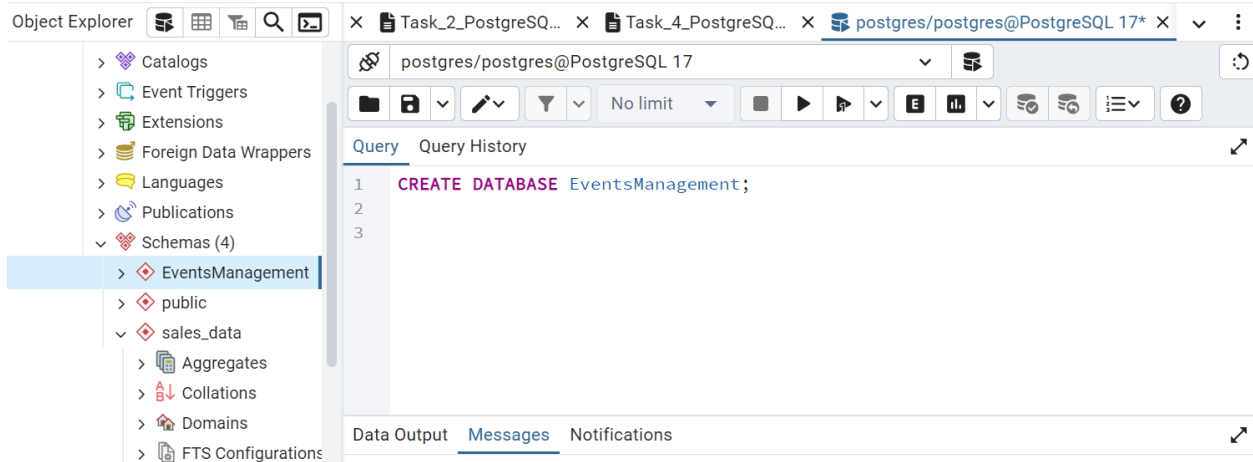
## Task 3

### Project: Event Management System using PostgreSQL.

Objective: To develop the application that allows users to create and manage events, track attendees, and handle event registrations efficiently. The project will include the following tasks:

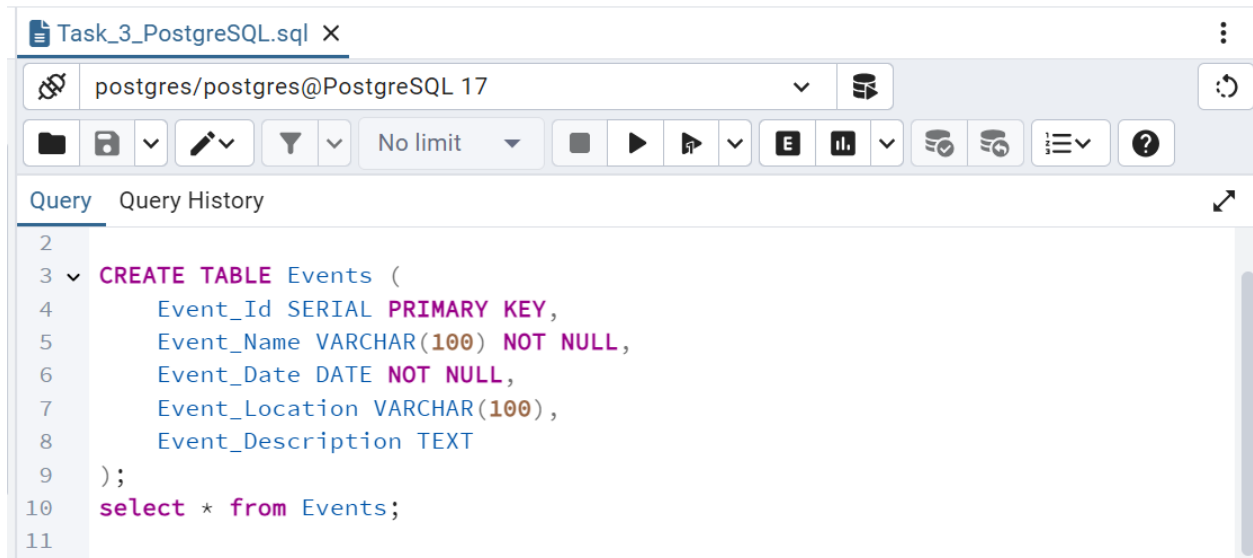
#### 1. Database Creation

a. Create a database named "EventsManagement."

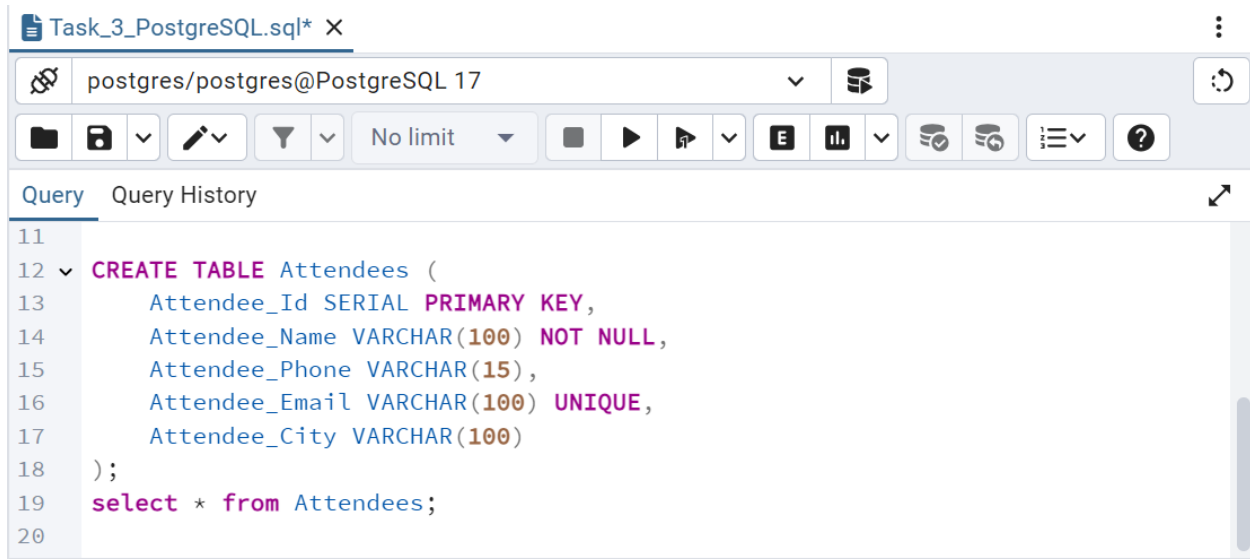


b. Create tables for Events, Attendees, and Registrations.

i. Events- Event\_Id, Event\_Name, Event\_Date, Event\_Location, Event\_Description



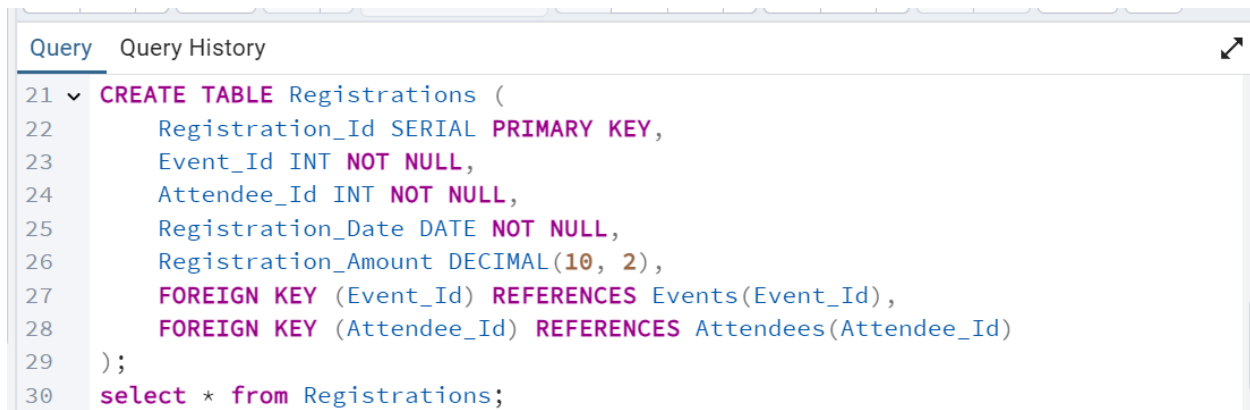
ii. Attendees- Attendee\_Id, Attendee\_Name, Attendee\_Phone, Attendee\_Email, Attendee\_City



```
Task_3_PostgreSQL.sql* x
postgres/postgres@PostgreSQL 17
No limit
Query Query History
11
12 CREATE TABLE Attendees (
13     Attendee_Id SERIAL PRIMARY KEY,
14     Attendee_Name VARCHAR(100) NOT NULL,
15     Attendee_Phone VARCHAR(15),
16     Attendee_Email VARCHAR(100) UNIQUE,
17     Attendee_City VARCHAR(100)
18 );
19 select * from Attendees;
20
```

iii. Registrations-Registration\_id, Event\_Id, Attendee\_Id,Registration\_Date,Registration\_Amount.

The FOREIGN KEY constraint in the Registrations table references the Event\_Id column in the Events table and the Attendee\_Id column in the Attendees table.

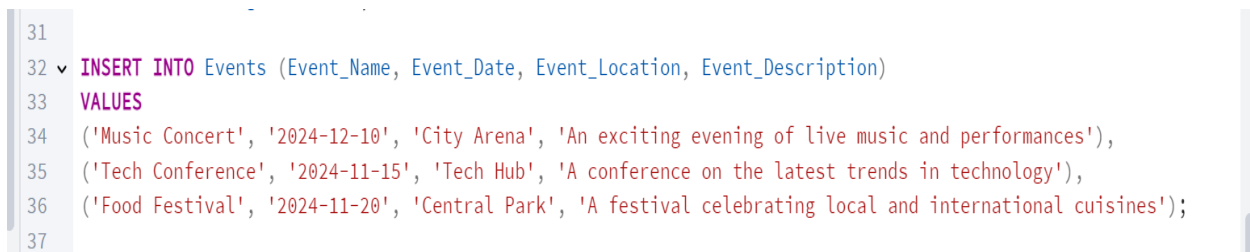


```
Query Query History
21 CREATE TABLE Registrations (
22     Registration_Id SERIAL PRIMARY KEY,
23     Event_Id INT NOT NULL,
24     Attendee_Id INT NOT NULL,
25     Registration_Date DATE NOT NULL,
26     Registration_Amount DECIMAL(10, 2),
27     FOREIGN KEY (Event_Id) REFERENCES Events(Event_Id),
28     FOREIGN KEY (Attendee_Id) REFERENCES Attendees(Attendee_Id)
29 );
30 select * from Registrations;
```

## 2. Data Creation

Insert some sample data for Events, Attendees, and Registrations tables with respective fields.

a. Insert data in Events Table



```
31
32 INSERT INTO Events (Event_Name, Event_Date, Event_Location, Event_Description)
33 VALUES
34 ('Music Concert', '2024-12-10', 'City Arena', 'An exciting evening of live music and performances'),
35 ('Tech Conference', '2024-11-15', 'Tech Hub', 'A conference on the latest trends in technology'),
36 ('Food Festival', '2024-11-20', 'Central Park', 'A festival celebrating local and international cuisines');
37
```

### b. Insert data in Attendees Table

```
postgres/postgres@PostgreSQL 17
Query Query History
37
38 INSERT INTO Attendees (Attendee_Name, Attendee_Phone, Attendee_Email, Attendee_City)
39 VALUES
40 ('Sucharita', '1234567890', 'Sucharita@example.com', 'Bengaluru'),
41 ('Madhuri', '9876543210', 'Madhuri@example.com', 'Pune'),
42 ('Samrath', '5556667777', 'Samrath@example.com', 'Hyderabad');
43
```

### c. Insert data in Registrations Table

```
Query Query History
44
45 INSERT INTO Registrations (Event_Id, Attendee_Id, Registration_Date, Registration_Amount)
46 VALUES
47 (1, 1, '2024-12-01', 150.00),
48 (2, 2, '2024-11-10', 200.00),
49 (3, 3, '2024-11-18', 300.00);
```

## 3. Manage Event Details

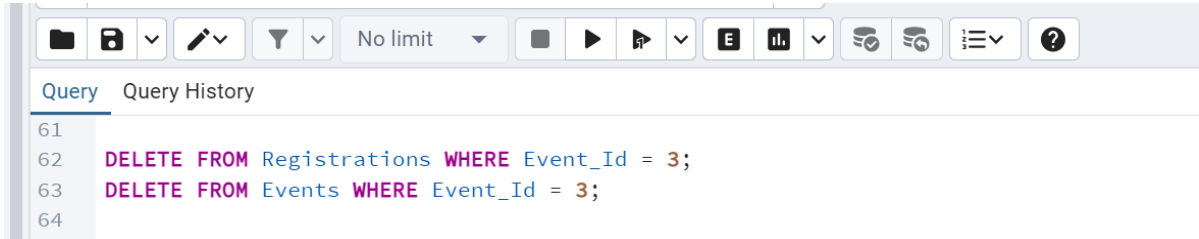
### a) Inserting a new event.

```
Query Query History
53
54 INSERT INTO Events (Event_Name, Event_Date, Event_Location, Event_Description)
55 VALUES
56 ('Fashion Show', '2024-12-05', 'Ramp Gallery', 'A display of modern Fashion Jewellery and Dresses');
57
```

### b) Updating an event's information.

```
postgres/postgres@PostgreSQL 17
Query Query History
57
58 UPDATE Events
59 SET Event_Date = '2024-11-18'
60 WHERE Event_Name = 'Tech Conference';
61
```

c) Deleting an event.



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, editing, filtering, and execution. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query on lines 61 to 64. The query consists of two DELETE statements: one for the 'Registrations' table and one for the 'Events' table, both targeting records where 'Event\_Id' is 3.

```
61
62 DELETE FROM Registrations WHERE Event_Id = 3;
63 DELETE FROM Events WHERE Event_Id = 3;
64
```

#### 4) Manage Track Attendees & Handle Events

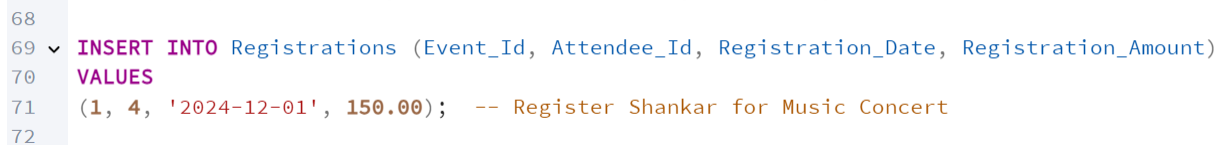
a) Inserting a new attendee.



The screenshot shows a SQL query editor with a toolbar at the top. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query on lines 65 to 68. The query is an INSERT statement into the 'Attendees' table, with columns 'Attendee\_Name', 'Attendee\_Phone', 'Attendee\_Email', and 'Attendee\_City'. The values provided are 'Shankar', '7978699431', 'Shankar@example.com', and 'Mumbai'.

```
65 INSERT INTO Attendees (Attendee_Name, Attendee_Phone, Attendee_Email, Attendee_City)
66 VALUES
67 ('Shankar', '7978699431', 'Shankar@example.com', 'Mumbai');
68
```

b) Registering an attendee for an event.



The screenshot shows a SQL query editor with a toolbar at the top. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query on lines 68 to 72. The query is an INSERT statement into the 'Registrations' table, with columns 'Event\_Id', 'Attendee\_Id', 'Registration\_Date', and 'Registration\_Amount'. The values provided are 1, 4, '2024-12-01', and 150.00. A comment '-- Register Shankar for Music Concert' is included at the end of the query.

```
68
69 INSERT INTO Registrations (Event_Id, Attendee_Id, Registration_Date, Registration_Amount)
70 VALUES
71 (1, 4, '2024-12-01', 150.00); -- Register Shankar for Music Concert
72
```

5. Develop queries to retrieve event information, generate attendee lists, and calculate event attendance statistics.

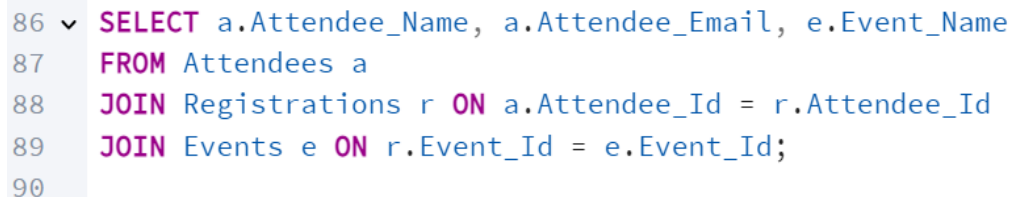
a) To retrieve event information



The screenshot shows a SQL query editor with a toolbar at the top. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query on lines 72 to 74. The query is a simple SELECT statement that retrieves all data from the 'Events' table.

```
72
73 SELECT * FROM Events;
74
```

b) To generate attendee lists



The screenshot shows a SQL query editor with a toolbar at the top. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query on lines 86 to 90. The query is a JOIN statement that retrieves attendee information from the 'Attendees' table, joined with the 'Registrations' table, and then with the 'Events' table. The columns selected are 'Attendee\_Name', 'Attendee\_Email', and 'Event\_Name'.

```
86 SELECT a.Attendee_Name, a.Attendee_Email, e.Event_Name
87 FROM Attendees a
88 JOIN Registrations r ON a.Attendee_Id = r.Attendee_Id
89 JOIN Events e ON r.Event_Id = e.Event_Id;
90
```

c) To generate attendee list for a specific Event (Tech Conference)

```
75 ✓ SELECT Attendees.Attendee_Name, Attendees.Attendee_Email, Registrations.Registration_Date
76 FROM Attendees
77 JOIN Registrations ON Attendees.Attendee_Id = Registrations.Attendee_Id
78 JOIN Events ON Registrations.Event_Id = Events.Event_Id
79 WHERE Events.Event_Name = 'Tech Conference';
80
```

d) To calculate event attendance statistics

```
89 ✓ SELECT e.Event_Name, COUNT(r.Registration_Id) AS Total_Attendees, SUM(r.Registration_Amount) AS Total_Revenue
90 FROM Events e
91 LEFT JOIN Registrations r ON e.Event_Id = r.Event_Id
92 GROUP BY e.Event_Name;
```

## Full Query

### 1. Database Creation

```
CREATE DATABASE EventsManagement;
```

### 2. Create Tables

#### a. Create the Events table

```
CREATE TABLE Events (
    Event_Id SERIAL PRIMARY KEY,
    Event_Name VARCHAR(100) NOT NULL,
    Event_Date DATE NOT NULL,
    Event_Location VARCHAR(100),
    Event_Description TEXT
);
select * from Events;
```

#### b. Create the Attendees table

```
CREATE TABLE Attendees (
    Attendee_Id SERIAL PRIMARY KEY,
    Attendee_Name VARCHAR(100) NOT NULL,
    Attendee_Phone VARCHAR(15),
```

```
Attendee_Email VARCHAR(100) UNIQUE,  
Attendee_City VARCHAR(100)  
);
```

```
select * from Attendees;
```

### **c. Create the Registrations table**

```
CREATE TABLE Registrations (  
    Registration_Id SERIAL PRIMARY KEY,  
    Event_Id INT NOT NULL,  
    Attendee_Id INT NOT NULL,  
    Registration_Date DATE NOT NULL,  
    Registration_Amount DECIMAL(10, 2),  
    FOREIGN KEY (Event_Id) REFERENCES Events(Event_Id),  
    FOREIGN KEY (Attendee_Id) REFERENCES Attendees(Attendee_Id)  
);
```

```
select * from Registrations;
```

## **3. Data Creation**

### **a. Inserting sample data for Events**

```
INSERT INTO Events (Event_Name, Event_Date, Event_Location, Event_Description)  
VALUES  
( 'Music Concert', '2024-12-10', 'City Arena', 'An exciting evening of live music and performances'),  
( 'Tech Conference', '2024-11-15', 'Tech Hub', 'A conference on the latest trends in technology'),  
( 'Food Festival', '2024-11-20', 'Central Park', 'A festival celebrating local and international cuisines');
```

### **b. Inserting sample data for Attendees**

```
INSERT INTO Attendees (Attendee_Name, Attendee_Phone, Attendee_Email, Attendee_City)  
VALUES  
( 'Sucharita', '1234567890', 'sucharita@example.com', 'Bengaluru'),  
( 'Madhuri', '9876543210', 'madhuri@example.com', 'Pune'),  
( 'Samrath', '5556667777', 'samrath@example.com', 'Hyderabad');
```

### **c. Inserting sample data for Registrations**

```
INSERT INTO Registrations (Event_Id, Attendee_Id, Registration_Date, Registration_Amount)
VALUES
(1, 1, '2024-12-01', 150.00),
(2, 2, '2024-11-10', 200.00),
(3, 3, '2024-11-18', 300.00);
```

## **4. Manage Event Details**

### **a) Inserting a New Event**

To insert a new event into the Events table:

```
INSERT INTO Events (Event_Name, Event_Date, Event_Location, Event_Description)
VALUES
('Fashion Show', '2024-12-05', 'Ramp Gallery', 'A display of modern Fashion Jewellery and Dresses');
```

### **b) Updating an Event's Information**

To update an existing event's details (e.g., changing the date of the "Tech Conference"):

```
UPDATE Events
SET Event_Date = '2024-11-18'
WHERE Event_Name = 'Tech Conference';
```

### **c) Deleting an Event**

To delete an event (e.g., delete the "Food Festival")

```
DELETE FROM Registrations WHERE Event_Id = 3;
DELETE FROM Events WHERE Event_Id = 3;
```

## **5. Manage Track Attendees & Handle Events**

### **a) Inserting a New Attendee**

To insert a new attendee Shankar

```
INSERT INTO Attendees (Attendee_Name, Attendee_Phone, Attendee_Email, Attendee_City)
VALUES
('Shankar', '7978699431', 'Shankar@example.com', 'Mumbai');
```

**b) Registering an Attendee for an Event**

```
INSERT INTO Registrations (Event_Id, Attendee_Id, Registration_Date, Registration_Amount)
```

```
VALUES
```

```
(1, 4, '2024-12-01', 150.00); -- Register Shankar for Music Concert
```

**6. Develop Queries to Retrieve Event Information and Attendance Statistics**

**a) Retrieve All Event Information**

```
SELECT * FROM Events;
```

**b) Retrieve List of Attendees for a Specific Event**

To generate attendee list for a specific Event for ex:- Tech Conference

```
SELECT Attendees.Attendee_Name, Attendees.Attendee_Email, Registrations.Registration_Date
```

```
FROM Attendees
```

```
JOIN Registrations ON Attendees.Attendee_Id = Registrations.Attendee_Id
```

```
JOIN Events ON Registrations.Event_Id = Events.Event_Id
```

```
WHERE Events.Event_Name = 'Tech Conference';
```

**c) To generate attendee lists**

```
SELECT a.Attendee_Name, a.Attendee_Email, e.Event_Name
```

```
FROM Attendees a
```

```
JOIN Registrations r ON a.Attendee_Id = r.Attendee_Id
```

```
JOIN Events e ON r.Event_Id = e.Event_Id;
```

**d) To calculate event attendance statistics**

```
SELECT e.Event_Name, COUNT(r.Registration_Id) AS Total_Attendees, SUM(r.Registration_Amount) AS  
Total_Revenue
```

```
FROM Events e
```

```
LEFT JOIN Registrations r ON e.Event_Id = r.Event_Id
```

```
GROUP BY e.Event_Name;
```