**EventEase - EVENT MANAGEMENT SYSTEM**

Welcome to EventEase (EMS), where we simplify the process of organizing events, whether it's a corporate conference, a wedding, or a fundraiser. With EMS, you have access to tools for venue selection, budget management, guest list coordination, and marketing strategies, empowering you to execute flawless events effortlessly. Our user-friendly interface saves you time and effort while ensuring attention to detail. Whether you're a seasoned planner or new to event organization, EMS provides the support and resources you need to execute your event flawlessly. Experience the future of event planning with EMS and turn your vision into reality, one event at a time.

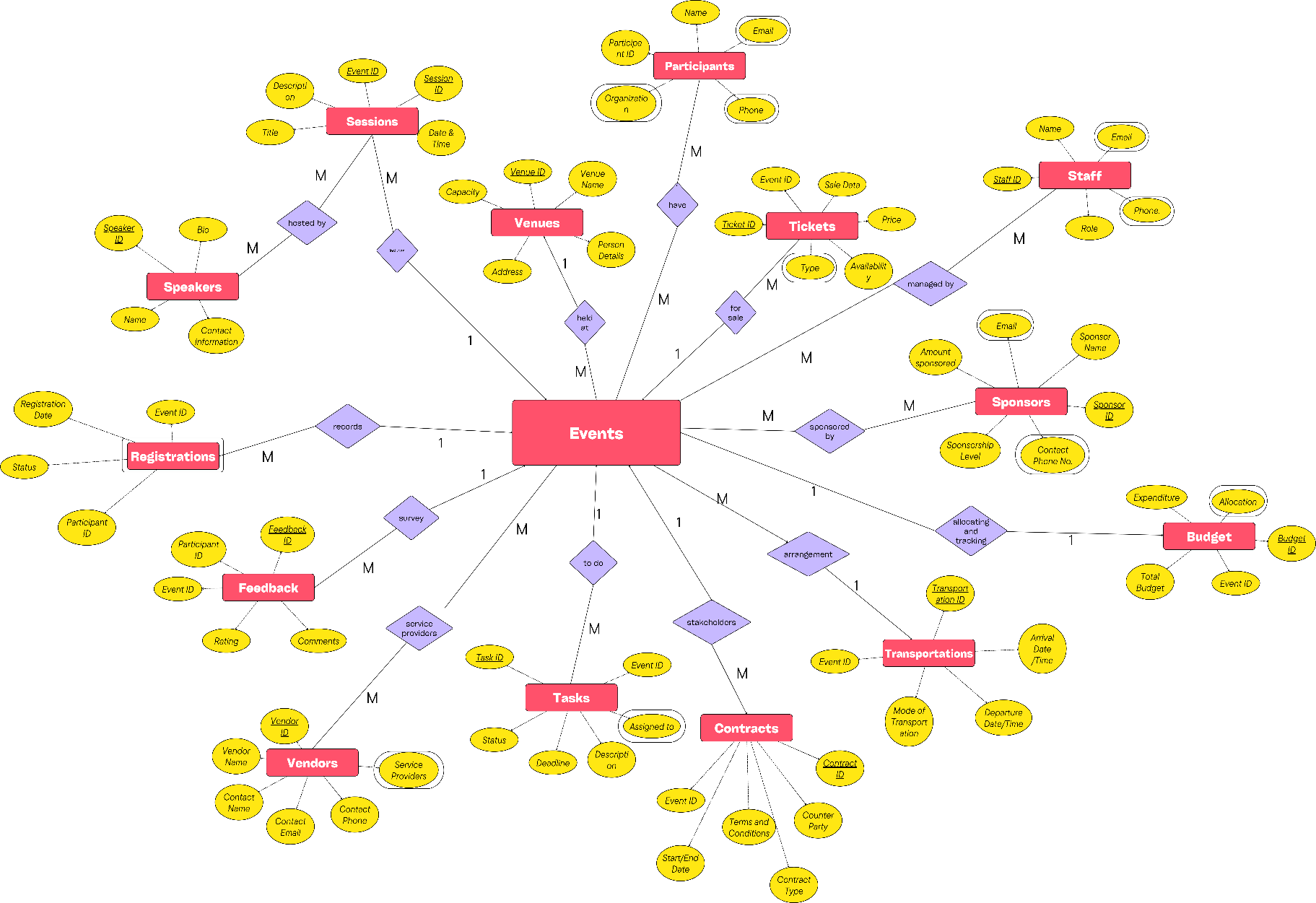
**Problem Statement:**

In today's dynamic world, organizing events efficiently is crucial for businesses, organizations, and individuals. However, many encounter challenges such as manual processes, communication gaps, and resource mismanagement. To address these issues, a comprehensive Event Management System (EMS) is needed. This system should streamline event planning, participant engagement, and resource allocation while providing insights into event performance.

**Target Audience:**

The target audience for this system includes event planners, organizers, businesses, educational institutions, non-profit organizations, and individuals involved in hosting various events such as conferences, seminars, workshops, festivals, concerts, and fundraisers.

**ENTITY -RELATIONSHIP DIAGRAM**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Events ID  (primary key) | Event Name | Venue ID  (foreign key) | Date | Time | Description |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Schema: Event (Event ID, Event Name, Date, Time, Venue ID, Description)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Venue ID  (primary key) | Venue Name | City | Capacity | Contact Name |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Schema: Venue (Venue ID, Venue Name, Contact Name,)

|  |  |  |  |
| --- | --- | --- | --- |
| Venue ID  (primary key) | Address | Email | Contact No. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Schema: Venue(Venue ID, Address, Email, Contact no.)

|  |  |
| --- | --- |
| Participant ID  (primary key) | Name |
|  |  |
|  |  |
|  |  |
|  |  |

Schema: Participants (Participant ID, Name, Email, Phone No.)

|  |  |  |
| --- | --- | --- |
| Participant ID  (primary key) | Email | Phone no. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Schema: Participants(Participant ID, Email, Phone no.)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ticket ID  (primary key) | Event ID  (foreign key) | Price | Type | Availability | Sale Date |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Schema: Tickets (Ticket ID, Event ID, Price, Type, Availability, Sale Date)

|  |  |  |
| --- | --- | --- |
| Staff ID  (primary key) | Name | Role |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Schema: Staff (Staff ID, Name, Role)

|  |  |  |
| --- | --- | --- |
| Staff ID  (primary key) | Email | Phone no. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Schema: Staff( Staff ID, Email, Phone no.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sponsor ID  (primary key) | Sponsor Name | Contact Person | Sponsorship Level | Amount Sponsored |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Schema: Sponsors (Sponsor ID, Sponsor Name, Sponsorship Level, Amount Sponsored)

|  |  |  |
| --- | --- | --- |
| Sponsor ID  (primary key) | Email | Phone no. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Schema: Sponsor(Sponsor ID, Email, Phone no.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Budget ID  (primary key) | Event ID  (foreign key) | Total Budget | Allocation Details | Expenditure |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Schema: Budget (Budget ID, Event ID, Total Budget, Allocation Details, Expenditure)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Session ID  (primary key) | Event ID  (foreign key) | Title | Description | Date | Time |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Schema: Session (Session ID, Event ID, Title, Description, Date, Time)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Speaker ID  (primary key) | First\_Name | Last\_Name | Bio | Contact |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Schema: Speaker (Speaker ID, First\_Name, Last\_Name, Bio, Contact)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Registration ID  (primary key) | Event ID  (foreign key) | Participant ID  (foreign key) | Registration Date | Status |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Schema: Registrations (Registration ID, Event ID, Participant ID, Registration Date, Status)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feedback ID  (primary key) | Event ID  (foreign key) | Participant ID | Rating | Comment |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Schema: Feedback (Feedback ID, Event ID, Participant ID, Rating, Comments)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vendor ID  (primary key) | Vendor First\_Name | Vendor Last\_Name | Contact  First\_Name | Contact Last\_Name | Service Providers |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Schema: Vendor (Vendor ID, Vendor First\_Name, Vendor Last\_Name, Contact First\_name, contact last\_name, service provider)

|  |  |  |
| --- | --- | --- |
| Vendor ID  (primary key) | Email | Phone no. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Schema: Vendor (Vendor ID, Email, Phone no.)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Transportation id  (primary key) | Event id  (foreign key) | Mode of transportation | Departure date | Departure time | Arrival date | Arrival time | Pick up location | Drop off location |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Schema: Transportation ( transportation id, event id, mode of transportation, departure date, departure time, arrival date, arrival time, pick up location, drop off location)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task id  (primary key) | Event id  (foreign key) | description | Assigned to | status | deadline |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Schema: tasks(task id, event id, description, assigned to, status, deadline)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Contract id  (primary key) | Event id  (foreign key) | Counter party | Contract type | Terms and conditions | Start date | End date |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Schema: contract(contract id, event id, counter party, contract type, terms and conditions start date, end date)

|  |  |  |
| --- | --- | --- |
| Participant ID | Event ID | have |
|  |  |  |
|  |  |  |

Schema: have(Participant ID, Event ID)

|  |  |  |
| --- | --- | --- |
| Sponsor ID | Event ID | Sponsored by |
|  |  |  |
|  |  |  |

Schema: Sponsored by(Sponsor ID, Event ID)

|  |  |  |
| --- | --- | --- |
| Staff ID | Event ID | Managed by |
|  |  |  |
|  |  |  |

Schema: Managed by( Staff ID, Event ID)

|  |  |  |
| --- | --- | --- |
| Vendor ID | Event ID | Service provided |
|  |  |  |
|  |  |  |

Schema: Service Provided(Vendor ID, Event ID)

|  |  |  |
| --- | --- | --- |
| Session ID | Speaker ID | Hosted by |
|  |  |  |
|  |  |  |

Schema: Hosted by(Session ID, Speaker ID)

**CREATION OF TABLES**

-- Create Venues table

CREATE TABLE Venues (

Venue\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Venue\_Name VARCHAR(100) NOT NULL,

Address VARCHAR(200) NOT NULL,

Capacity INT NOT NULL,

Contact\_Person VARCHAR(100),

Contact\_Email VARCHAR(100),

Contact\_Phone VARCHAR(20)

);

INSERT INTO Venues (Venue\_Name, Address, Capacity, Contact\_Person, Contact\_Email, Contact\_Phone)

VALUES

('Convention Center', '1 Main St, Cityville', 2000, 'John Smith', 'john@conventioncenter.com', '123-456-7890'),

('Auditorium', '5 Park Ave, Townsville', 500, 'Jane Doe', 'jane@auditorium.com', '987-654-3210'),

('Art Gallery', '10 Museum Rd, Villagetown', 150, 'Alice Johnson', 'alice@artgallery.com', '111-222-3333'),

('Coworking Space', '15 Tech Lane, Hamletville', 100, 'Bob Brown', 'bob@coworking.com', '444-555-6666'),

('Hotel Grand', '20 Central Ave, Lodgetown', 800, 'Carol White', 'carol@hotelgrand.com', '777-888-9999');

-- Create Participants table

CREATE TABLE Participants (

Participant\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) NOT NULL UNIQUE, -- Enforce unique email addresses

Phone VARCHAR(20),

Organization VARCHAR(100)

);

INSERT INTO Participants (Name, Email, Phone, Organization)

VALUES

('Michael Lee', 'michael.lee@gmail.com', '555-123-4567', 'Tech Company Inc.'),

('Sarah Jones', 'sarah.jones@yahoo.com', '222-333-4444', 'Marketing Agency'),

('David Williams', 'david.williams@outlook.com', '111-555-7777', 'University'),

('Emily Brown', 'emily.brown@hotmail.com', '999-888-6666', 'Freelancer'),

('Daniel Garcia', 'daniel.garcia@aol.com', '333-222-1111', 'Non-Profit Organization');

-- Create Speakers table

CREATE TABLE Speakers (

Speaker\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Bio VARCHAR(500),

Contact\_Information VARCHAR(200)

);

-- Inserting data into the Speakers table

INSERT INTO Speakers (Name, Bio, Contact\_Information) VALUES

('John Doe', 'Speaker bio for John Doe', 'john@example.com'),

('Jane Smith', 'Speaker bio for Jane Smith', 'jane@example.com'),

('Alice Johnson', 'Speaker bio for Alice Johnson', 'alice@example.com'),

('Michael Brown', 'Speaker bio for Michael Brown', 'michael@example.com'),

('Emily Davis', 'Speaker bio for Emily Davis', 'emily@example.com');

-- Create Staff table

CREATE TABLE Staff (

Staff\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) NOT NULL UNIQUE, -- Enforce unique email addresses

Phone VARCHAR(20),

Role VARCHAR(50)

);

INSERT INTO Staff (Name, Email, Phone, Role)

VALUES

('John Smith', 'john.smith@company.com', '555-123-4567', 'Event Manager'),

('Jane Doe', 'jane.doe@company.com', '222-333-4444', 'Marketing Specialist'),

('David Williams', 'david.williams@company.com', '111-555-7777', 'Technical Lead'),

('Emily Brown', 'emily.brown@company.com', '999-888-6666', 'Registration Coordinator'),

('Daniel Garcia', 'daniel.garcia@company.com', '333-222-1111', 'Logistics Coordinator');

-- Create Sponsors table

CREATE TABLE Sponsors (

Sponsor\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Sponsor\_Name VARCHAR(100) NOT NULL,

Contact\_Person VARCHAR(100),

Contact\_Email VARCHAR(100),

Contact\_Phone VARCHAR(20),

Sponsorship\_Level VARCHAR(50),

Amount\_Sponsored DECIMAL(10,2) NOT NULL DEFAULT 0, -- Use DECIMAL for currency values

CONSTRAINT chk\_Amount\_Sponsored CHECK (Amount\_Sponsored >= 0) -- Check if amount sponsored is non-negative

);

INSERT INTO Sponsors (Sponsor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Sponsorship\_Level, Amount\_Sponsored)

VALUES

('Tech Giant Inc.', 'Sarah Lee', 'sarah.lee@techgiant.com', '123-456-7890', 'Platinum', 10000.00),

('Marketing Agency', 'David Miller', 'david.miller@marketingagency.com', '555-222-1111', 'Gold', 5000.00),

('Software Solutions', 'Emily Jones', 'emily.jones@softwaresolutions.com', '987-654-3210', 'Silver', 2500.00),

('Cloud Provider', 'Michael Brown', 'michael.brown@cloudprovider.com', '444-333-2222', 'Bronze', 1000.00),

('Non-Profit Organization', 'Alice Garcia', 'alice.garcia@nonprofit.org', '777-888-9999', 'In-Kind', 2000.00);

CREATE TABLE Events (

Event\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Event\_Name VARCHAR(100) NOT NULL,

Event\_Date DATE NOT NULL,

Event\_Time DATETIME NOT NULL, -- DATETIME for combined date and time

Venue\_ID INT NOT NULL,

Description VARCHAR(500),

CONSTRAINT fk\_Events\_Venues FOREIGN KEY (Venue\_ID) REFERENCES Venues(Venue\_ID)

);

INSERT INTO Events (Event\_Name, Event\_Date, Event\_Time, Venue\_ID, Description)

VALUES

('Tech Conference 2024', '2024-06-15', '09-00-00', 1, 'A gathering of tech enthusiasts and industry leaders'),

('Art Exhibition Opening', '2024-05-20', '18-00-00', 3, 'Showcase of local and international contemporary art'),

('Community Workshop', '2024-07-10', '10-00-00', 4, 'Interactive session on sustainable living practices'),

('Networking Mixer', '2024-04-25', '17-00-00', 2, 'Opportunity to connect with professionals across various fields'),

('Summer Music Festival', '2024-08-12', '14-00-00', 5, 'Live music performances by popular artists');

-- Create Tickets table

CREATE TABLE Tickets (

Ticket\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Event\_ID INT NOT NULL,

Price DECIMAL(10,2) NOT NULL, -- Use DECIMAL for currency values

Type VARCHAR(50) NOT NULL,

Availability INT NOT NULL DEFAULT 0,

Sale\_Date DATE,

CONSTRAINT fk\_Tickets\_Events FOREIGN KEY (Event\_ID) REFERENCES Events(Event\_ID),

CONSTRAINT chk\_Ticket\_Price CHECK (Price > 0), -- Check if ticket price is positive

CONSTRAINT chk\_Ticket\_Availability CHECK (Availability >= 0) -- Check if availability is non-negative

);

-- Inserting data into the Tickets table

INSERT INTO Tickets (Event\_ID, Price, Type, Availability, Sale\_Date) VALUES

(1, 50.00, 'General Admission', 100, '2024-04-01'),

(1, 75.00, 'VIP', 50, '2024-04-01'),

(2, 30.00, 'Standard', 200, '2024-03-15'),

(2, 40.00, 'Premium', 150, '2024-03-15'),

(3, 20.00, 'Early Bird', 300, '2024-04-10');

-- Create Budget table

CREATE TABLE Budget (

Budget\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Event\_ID INT NOT NULL,

Total\_Budget DECIMAL(10,2) NOT NULL, -- Use DECIMAL for currency values

Allocation\_Details VARCHAR(500),

Expenditure DECIMAL(10,2), -- Use DECIMAL for currency values

CONSTRAINT fk\_Budget\_Events FOREIGN KEY (Event\_ID) REFERENCES Events(Event\_ID),

CONSTRAINT chk\_Total\_Budget CHECK (Total\_Budget >= 0), -- Check if total budget is non-negative

CONSTRAINT chk\_Expenditure CHECK (Expenditure >= 0) -- Check if expenditure is non-negative

);

-- Inserting data into the Budget table

INSERT INTO Budget (Event\_ID, Total\_Budget, Allocation\_Details, Expenditure) VALUES

(1, 5000.00, 'Venue rental, catering, marketing materials', 3500.00),

(2, 8000.00, 'Speaker fees, audio-visual equipment, advertising', 6000.00),

(3, 3000.00, 'Decorations, refreshments, volunteer expenses', 2500.00),

(4, 6000.00, 'Entertainment, prizes, event staff', 4500.00),

(5, 4000.00, 'Food, beverages, transportation', 3500.00);

-- Create Registrations table

CREATE TABLE Registrations (

Registration\_ID INT PRIMARY KEY,

Event\_ID INT NOT NULL,

Participant\_ID INT NOT NULL,

Registration\_Date DATE NOT NULL,

Status VARCHAR(50) NOT NULL,

FOREIGN KEY (Event\_ID) REFERENCES Events(Event\_ID),

FOREIGN KEY (Participant\_ID) REFERENCES Participants(Participant\_ID)

);

-- Inserting data into the Registrations table

INSERT INTO Registrations (Registration\_ID, Event\_ID, Participant\_ID, Registration\_Date, Status) VALUES

(1, 1, 1, '2024-03-01', 'Confirmed'),

(2, 1, 2, '2024-03-02', 'Confirmed'),

(3, 2, 3, '2024-03-03', 'Pending'),

(4, 3, 4, '2024-03-04', 'Confirmed'),

(5, 3, 5, '2024-03-05', 'Cancelled');

-- Create Feedback table

CREATE TABLE Feedback (

Feedback\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Event\_ID INT NOT NULL,

Participant\_ID INT NOT NULL,

Rating INT NOT NULL,

Comments VARCHAR(500),

CONSTRAINT fk\_Feedback\_Events FOREIGN KEY (Event\_ID) REFERENCES Events(Event\_ID),

CONSTRAINT fk\_Feedback\_Participants FOREIGN KEY (Participant\_ID) REFERENCES Participants(Participant\_ID),

CONSTRAINT chk\_Rating CHECK (Rating >= 1 AND Rating <= 5) -- Check if rating is within valid range

);

-- Inserting data into the Feedback table

INSERT INTO Feedback (Event\_ID, Participant\_ID, Rating, Comments) VALUES

(1, 1, 4, 'Great event overall. Enjoyed the sessions.'),

(1, 2, 5, 'Excellent organization and content. Would attend again.'),

(2, 3, 3, 'Good event, but could have been more interactive.'),

(3, 4, 5, 'Fantastic experience. Highly recommended.'),

(3, 5, 2, 'Disappointed with the lack of variety in sessions.');

-- Create Vendors table

CREATE TABLE Vendors (

Vendor\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Vendor\_Name VARCHAR(100) NOT NULL,

Contact\_Person VARCHAR(100),

Contact\_Email VARCHAR(100),

Contact\_Phone VARCHAR(20),

Services\_Provided VARCHAR(200)

);

-- Inserting data into the Vendors table

INSERT INTO Vendors (Vendor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Services\_Provided) VALUES

('ABC Catering', 'John Smith', 'john@example.com', '123-456-7890', 'Catering services'),

('XYZ Rentals', 'Alice Johnson', 'alice@example.com', '987-654-3210', 'Event equipment rentals'),

('PQR Decorations', 'Emily Davis', 'emily@example.com', '456-789-0123', 'Event decorations'),

('EFG Sound Systems', 'Michael Brown', 'michael@example.com', '789-012-3456', 'Sound system rental'),

('LMN Photography', 'Sarah Wilson', 'sarah@example.com', '321-654-9870', 'Event photography');

-- Create Transportation table

CREATE TABLE Transportation (

Transportation\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Event\_ID INT NOT NULL,

Mode\_of\_Transportation VARCHAR(100) NOT NULL,

Departure\_Date DATETIME NOT NULL,

Arrival\_Date DATETIME NOT NULL,

Pickup\_Dropoff\_Locations VARCHAR(200),

CONSTRAINT fk\_Transportation\_Events FOREIGN KEY (Event\_ID) REFERENCES Events(Event\_ID),

CONSTRAINT chk\_Transportation\_Dates CHECK (Departure\_Date < Arrival\_Date) -- Check if departure date is before arrival date

);

-- Inserting data into the Transportation table

INSERT INTO Transportation (Event\_ID, Mode\_of\_Transportation, Departure\_Date, Arrival\_Date, Pickup\_Dropoff\_Locations) VALUES

(1, 'Bus', '2024-04-01 08:00:00', '2024-04-01 10:00:00', 'Airport - Event Venue'),

(1, 'Shuttle', '2024-04-01 08:30:00', '2024-04-01 09:30:00', 'Hotel - Event Venue'),

(2, 'Car Rental', '2024-03-15 10:00:00', '2024-03-15 12:00:00', 'Airport - Event Venue'),

(3, 'Train', '2024-04-10 09:00:00', '2024-04-10 11:00:00', 'Station - Event Venue'),

(4, 'Taxi', '2024-05-01 14:00:00', '2024-05-01 16:00:00', 'Airport - Event Venue');

-- Create Tasks table

CREATE TABLE Tasks (

Task\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Event\_ID INT NOT NULL,

Description VARCHAR(500) NOT NULL,

Assigned\_To VARCHAR(100) NOT NULL,

Deadline DATE NOT NULL,

Status VARCHAR(50) NOT NULL,

CONSTRAINT fk\_Tasks\_Events FOREIGN KEY (Event\_ID) REFERENCES Events(Event\_ID)

);

-- Inserting data into the Tasks table

INSERT INTO Tasks (Event\_ID, Description, Assigned\_To, Deadline, Status) VALUES

(1, 'Prepare presentation slides', 'John Doe', '2024-04-01', 'Pending'),

(1, 'Setup event venue', 'Jane Smith', '2024-04-01', 'In Progress'),

(2, 'Arrange transportation', 'Alice Johnson', '2024-03-15', 'Completed'),

(3, 'Decorate event hall', 'Michael Brown', '2024-04-10', 'Pending'),

(4, 'Confirm guest list', 'Emily Davis', '2024-05-01', 'In Progress');

-- Create Sessions table

CREATE TABLE Sessions (

Session\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Event\_ID INT NOT NULL,

Title VARCHAR(100) NOT NULL,

Description VARCHAR(500),

Session\_Date DATE NOT NULL,

Session\_Time DATETIME NOT NULL,

CONSTRAINT fk\_Sessions\_Events FOREIGN KEY (Event\_ID) REFERENCES Events(Event\_ID)

);

-- Inserting data into the Sessions table

INSERT INTO Sessions (Event\_ID, Title, Description, Session\_Date, Session\_Time) VALUES

(1, 'Keynote Address', 'Opening keynote by renowned speaker', '2024-04-01', '2024-04-01 09:00:00'),

(1, 'Panel Discussion', 'Panel discussion on industry trends', '2024-04-01', '2024-04-01 11:00:00'),

(2, 'Workshop: Data Analytics', 'Hands-on workshop on data analysis techniques', '2024-03-15', '2024-03-15 10:00:00'),

(2, 'Networking Lunch', 'Networking session over lunch', '2024-03-15', '2024-03-15 12:30:00'),

(3, 'Product Showcase', 'Showcasing latest products and innovations', '2024-04-10', '2024-04-10 11:00:00');

-- Create Contracts table

CREATE TABLE Contracts (

Contract\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Event\_ID INT NOT NULL,

Counterparty VARCHAR(100),

Contract\_Type VARCHAR(50),

Terms\_and\_Conditions VARCHAR(500),

Start\_Date DATE,

End\_Date DATE,

CONSTRAINT fk\_Contracts\_Events FOREIGN KEY (Event\_ID) REFERENCES Events(Event\_ID)

);

-- Inserting data into the Contracts table

INSERT INTO Contracts (Event\_ID, Counterparty, Contract\_Type, Terms\_and\_Conditions, Start\_Date, End\_Date) VALUES

(1, 'ABC Catering', 'Catering Services', 'Agreed upon catering services for the event', '2024-04-01', '2024-04-03'),

(2, 'XYZ Rentals', 'Equipment Rental', 'Rental of audio-visual equipment for the event', '2024-03-15', '2024-03-16'),

(3, 'PQR Decorations', 'Decoration Services', 'Decoration services for the event venue', '2024-04-10', '2024-04-12'),

(4, 'EFG Sound Systems', 'Sound System Rental', 'Rental of sound systems for the event', '2024-05-01', '2024-05-03'),

(5, 'LMN Photography', 'Photography Services', 'Photography services coverage for the event', '2024-06-15', '2024-06-16');

show tables;

describe Events;

select \*from Venues;

**CREATING VIEWS:**

1. The "EventRegistrations" view combines data from the Events, Registrations, and Participants tables, providing details such as event name, participant name, registration date, and status for each registration.

**CREATE VIEW EventRegistrations AS**

**SELECT e.Event\_Name, p.Name, r.Registration\_Date, r.Status**

**FROM Events e**

**INNER JOIN Registrations r ON e.Event\_ID = r.Event\_ID**

**INNER JOIN Participants p ON r.Participant\_ID = p.Participant\_ID;**

**describe EventRegistrations;**

1. The "AvailableTickets" view displays event names, ticket types, prices, and availability for tickets with availability greater than zero, by joining the Events and Tickets tables.

**CREATE VIEW AvailableTickets AS**

**SELECT e.Event\_Name, t.Type, t.Price, t.Availability**

**FROM Events e**

**INNER JOIN Tickets t ON e.Event\_ID = t.Event\_ID**

**WHERE t.Availability > 0;**

**describe AvailableTickets;**

**USING JOINS :**

1. **-- Retrieve the names of participants registered for the event "Tech Conference 2024"**

SELECT p.Name

FROM Participants p

INNER JOIN Registrations r ON p.Participant\_ID = r.Participant\_ID

INNER JOIN Events e ON r.Event\_ID = e.Event\_ID

WHERE e.Event\_Name = 'Tech Conference 2024';

1. **-- Get details of Registrations including Event Name, Participant Name, and Registration Status**

SELECT Events.Event\_Name, Participants.Name AS Participant\_Name, Registrations.Status

FROM Registrations

JOIN Events ON Registrations.Event\_ID = Events.Event\_ID

JOIN Participants ON Registrations.Participant\_ID = Participants.Participant\_ID;

1. **-- Fetch Event Name, Venue Name, and Capacity along with respective Venues**

SELECT Events.Event\_Name, Venues.Venue\_Name, Venues.Capacity

FROM Events

JOIN Venues ON Events.Venue\_ID = Venues.Venue\_ID;

1. **-- Retrieve Sponsor details along with the Event they sponsored for**

SELECT Events.Event\_Name, Sponsors.Sponsor\_Name, Sponsors.Sponsorship\_Level

FROM Sponsors

JOIN Events ON Sponsors.Sponsor\_ID = Events.Event\_ID;

**USING SET OPERATORS:**

1. UNION-

SELECT Sponsor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Sponsorship\_Level, Amount\_Sponsored

FROM Sponsors

UNION

SELECT Vendor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Services\_Provided, NULL AS Amount\_Sponsored

FROM Vendors;

1. UNION ALL-

SELECT Sponsor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Sponsorship\_Level, Amount\_Sponsored

FROM Sponsors

UNION ALL

SELECT Vendor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Services\_Provided, NULL AS Amount\_Sponsored

FROM Vendors;

1. INTERSECT-

SELECT Sponsor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Sponsorship\_Level, Amount\_Sponsored

FROM Sponsors

INTERSECT

SELECT Vendor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Services\_Provided, NULL AS Amount\_Sponsored

FROM Vendors;

1. MINUS-

SELECT Sponsor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Sponsorship\_Level, Amount\_Sponsored

FROM Sponsors

MINUS

SELECT Vendor\_Name, Contact\_Person, Contact\_Email, Contact\_Phone, Services\_Provided, NULL AS Amount\_Sponsored

FROM Vendors;

**TRIGGERS:**

1. To create a trigger that gets invoked if the numbers entered in the **Phone** column are less than 10 characters long

**CREATE OR REPLACE TRIGGER check\_phone\_length**

**BEFORE INSERT ON Participants**

**FOR EACH ROW**

**BEGIN**

**IF LENGTH(:NEW.Phone) < 10 THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Phone number must be at least 10 characters long');**

**END IF;**

**END;**

**/**

Example:

-- Try to insert a participant with a wrong phone number

INSERT INTO Participants (Name, Email, Phone, Organization)

VALUES ('John Doe', 'john.doe@example.com', '123456', 'Tech Startup');

1. To create a trigger that gets invoked when the format of entering an email is incorrect, you can use a regular expression to validate the email format.

**CREATE OR REPLACE TRIGGER check\_email\_format**

**BEFORE INSERT ON Staff**

**FOR EACH ROW**

**DECLARE**

**email\_pattern VARCHAR(100) := '[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}';**

**BEGIN**

**IF NOT REGEXP\_LIKE(:NEW.Email, email\_pattern) THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Invalid email format');**

**END IF;**

**END;**

**/**

Example:

-- Try to insert a participant with an incorrect email format

INSERT INTO Participants (Name, Email, Phone, Organization)

VALUES ('John Doe', 'invalidemail.com', '555-123-4567', 'Tech Startup');

1. To create a trigger that checks the date and time format entered in the **Events** table and invokes when the format is incorrect

**CREATE OR REPLACE TRIGGER DateTimeFormatTrigger**

**BEFORE INSERT ON Events**

**FOR EACH ROW**

**DECLARE**

**v\_error\_message VARCHAR2(200);**

**BEGIN**

**-- Check if the date format is correct**

**IF NOT REGEXP\_LIKE(:NEW.Event\_Date, '^[0-9]{4}-[0-9]{2}-[0-9]{2}$') THEN**

**v\_error\_message := 'Invalid date format. Please enter a date in YYYY-MM-DD format.';**

**RAISE\_APPLICATION\_ERROR(-20001, v\_error\_message);**

**END IF;**

**-- Check if the time format is correct**

**IF NOT REGEXP\_LIKE(:NEW.Event\_Time, '^[0-9]{2}:[0-9]{2}:[0-9]{2}$') THEN**

**v\_error\_message := 'Invalid time format. Please enter a time in HH:MM:SS format.';**

**RAISE\_APPLICATION\_ERROR(-20002, v\_error\_message);**

**END IF;**

**END;**

**/**

Example:

INSERT INTO Events VALUES (6, 'Invalid Date Event', TO\_DATE('2024/06/15', 'YYYY/MM/DD'), TIMESTAMP '2024-06-15 09:00:00', 1, 'This event has an

invalid date format.');

INSERT INTO Events VALUES (6, 'Invalid Date Event', TO\_DATE('2024/06/15', 'YYYY/MM/DD'), TIMESTAMP '2024-06-15 09:00:00', 1, 'This event has an invalid date format.')

**CURSOR IMPLEMENTATION:**

1. This declares a cursor named event\_cursor for the Events table. It then opens the cursor, fetches each row of data from the Events table one by one, processes it and continues until there are no more rows to fetch. Finally, it closes the cursor.

DECLARE

-- Declare the cursor

CURSOR event\_cursor IS

SELECT \* FROM Events;

-- Declare variables to store column values

event\_id Events.Event\_ID%TYPE;

event\_name Events.Event\_Name%TYPE;

event\_date Events.Event\_Date%TYPE;

event\_time Events.Event\_Time%TYPE;

venue\_id Events.Venue\_ID%TYPE;

event\_description Events.Description%TYPE;

BEGIN

-- Open the cursor

OPEN event\_cursor;

-- Fetch rows from the cursor one by one

FETCH event\_cursor INTO event\_id, event\_name, event\_date, event\_time, venue\_id, event\_description;

-- Loop through the cursor to fetch all rows

WHILE event\_cursor%FOUND LOOP

-- Print or use the fetched values as needed

DBMS\_OUTPUT.PUT\_LINE(event\_id || ', ' || event\_name || ', ' || TO\_CHAR(event\_date, 'DD-MON-YY') || ', ' || TO\_CHAR(event\_time, 'DD-MON-YY HH:MI:SS AM') || ', ' || venue\_id || ', ' || event\_description);

-- Fetch the next row

FETCH event\_cursor INTO event\_id, event\_name, event\_date, event\_time, venue\_id, event\_description;

END LOOP;

-- Close the cursor

CLOSE event\_cursor;

END;

/

1. This cursor fetches data from the `Participants` table row by row and prints each row's details, including participant ID, name, email, phone, and organization, using `DBMS\_OUTPUT.PUT\_LINE` in a PL/SQL environment.

DECLARE

-- Declare the cursor

CURSOR participant\_cursor IS

SELECT \* FROM Participants;

-- Declare variables to store column values

participant\_id Participants.Participant\_ID%TYPE;

participant\_name Participants.Name%TYPE;

participant\_email Participants.Email%TYPE;

participant\_phone Participants.Phone%TYPE;

participant\_organization Participants.Organization%TYPE;

BEGIN

-- Open the cursor

OPEN participant\_cursor;

-- Fetch rows from the cursor one by one

FETCH participant\_cursor INTO participant\_id, participant\_name, participant\_email, participant\_phone, participant\_organization;

-- Loop through the cursor to fetch all rows

WHILE participant\_cursor%FOUND LOOP

-- Print or use the fetched values as needed

DBMS\_OUTPUT.PUT\_LINE(participant\_id || ', ' || participant\_name || ', ' || participant\_email || ', ' || participant\_phone || ', ' || participant\_organization);

-- Fetch the next row

FETCH participant\_cursor INTO participant\_id, participant\_name, participant\_email, participant\_phone, participant\_organization;

END LOOP;

-- Close the cursor

CLOSE participant\_cursor;

END;

/

**EMPATHY MAP**

