

DATABASE DESIGN FOR CEDAR ELEGANCE EVENTS

by

BYTE BUILDERS GROUP

Ali El Hajj  
[ali.elhajj03@lau.edu](mailto:ali.elhajj03@lau.edu)

Maher Saadi  
[maher.saadi@lau.edu](mailto:maher.saadi@lau.edu)

Imad Mahmoud  
[imad.mahmoud@lau.edu](mailto:imad.mahmoud@lau.edu)Ali Rida  
[ali.reda03@lau.edu](mailto:ali.reda03@lau.edu)   
  
Reina Harake  
[reina.harake@lau.edu](mailto:reina.harake@lau.edu)

REPORT  
submitted to Dr. Ramzi R. Haraty in partial fulfillment of the requirements of the course “CSC 375: Database Management Systems” in Computer Science

April 30, 2025

**Table of Contents**

[**System Overview and Data Management: 6**](#_Toc194879771)

[**ER Diagram Symbols: 7**](#_Toc194879772)

[**ER diagram for Cedar Elegance Events: 8**](#_Toc194879773)

[**Entity Types: 9**](#_Toc194879774)

[**1. EVENT: 9**](#_Toc194879775)

[**2. VENUE: 10**](#_Toc194879776)

[**3. ATTENDEE (Weak entity): 11**](#_Toc194879777)

[**4. ORGANIZER: 12**](#_Toc194879778)

[**5. STAFF: 13**](#_Toc194879779)

[**6. CUSTOMER: 14**](#_Toc194879780)

[**7. SPECIAL OFFERS 15**](#_Toc194879781)

[**8. SPONSOR: 16**](#_Toc194879782)

[**9. SERVICE: 17**](#_Toc194879783)

[**10. SUPPLIER: 18**](#_Toc194879784)

[**Relationships Between Entity Types: 19**](#_Toc194879785)

[** HOLD (N:M): 19**](#_Toc194879786)

[** USED IN(M:N): 20**](#_Toc194879787)

[** ATTENDS (1:N): 21**](#_Toc194879788)

[** WORKS ON (M:N) 22**](#_Toc194879789)

[** SPONSORS (M:N) 23**](#_Toc194879790)

[** ORGANIZES (M:N) 24**](#_Toc194879791)

[** PROVIDES FOR (M:N): 25**](#_Toc194879792)

[** HAS (M:N): 26**](#_Toc194879793)

[** PROVIDES (M:N): 27**](#_Toc194879794)

[** CHOOSES (1:1): 28**](#_Toc194879795)

[**ER to Relational Mapping Algorithms 30**](#_Toc194879796)

[**STEP 1: Mapping of Regular Entity Types 31**](#_Toc194879797)

[**1. EVENT 31**](#_Toc194879798)

[**2. STAFF 31**](#_Toc194879799)

[**3. SPONSOR 32**](#_Toc194879800)

[**4. VENUE 32**](#_Toc194879801)

[**5. SERVICE 32**](#_Toc194879802)

[**6. SPECIAL OFFER 33**](#_Toc194879803)

[**7. CUSTOMER 33**](#_Toc194879804)

[**8. SUPPLIER 33**](#_Toc194879805)

[**9. ORGANIZER 34**](#_Toc194879806)

[**STEP 2: Mapping of Weak Entity Types 35**](#_Toc194879807)

[**1. ATTENDEE 35**](#_Toc194879808)

[**STEP 3: Mapping of Binary 1:1 Relationship Types 36**](#_Toc194879809)

[**1. CHOOSES 36**](#_Toc194879810)

[**STEP 4: Mapping of Binary 1:N Relationship Types 37**](#_Toc194879811)

[**1. ATTENDEE (ATTENDS): 37**](#_Toc194879812)

[**2. STAFF (MANAGES): 37**](#_Toc194879813)

[**STEP 5: Mapping of M:N Relationship Types 38**](#_Toc194879814)

[**1. HOLDS 38**](#_Toc194879815)

[**2. USED IN 38**](#_Toc194879816)

[**3. WORKS ON 39**](#_Toc194879817)

[**5. ORGANIZES 39**](#_Toc194879818)

[**6. PROVIDES FOR 40**](#_Toc194879819)

[**7. HAS 40**](#_Toc194879820)

[**8. PROVIDES 40**](#_Toc194879821)

[**1. Event\_Types 41**](#_Toc194879822)

[**2. Staff\_Role 41**](#_Toc194879823)

[**3. Service\_Decoration 41**](#_Toc194879824)

[**4. Service\_Types 42**](#_Toc194879825)

[**5. Attendee\_Preferences 42**](#_Toc194879826)

[**6. Offer\_Types 42**](#_Toc194879827)

[**7. Customer\_Phone\_Number 42**](#_Toc194879828)

[**8. Supplier\_Phone\_Number 43**](#_Toc194879829)

[**9. Organizer\_Contact\_Information 43**](#_Toc194879830)

[**10. Organizer\_Role 43**](#_Toc194879831)

[**11. Venue\_Owner 43**](#_Toc194879832)

[**STEP 7: Mapping of Nary Relationship Types 44**](#_Toc194879833)

[**STEP 8: Final Displays 44**](#_Toc194879833)

**Creation of Database in SQL ………………………………………………..…..……………49**

**Regular Entity Types Decomposition ……………………...……………………………...…138**

**Introduction**:

Founded in 2003, Cedar Elegance Events is one of the leading event planning companies in Lebanon. Known for its meticulous attention to detail and unparalleled creativity, the company has established itself as a premier destination for organizing luxurious weddings, corporate gatherings, and high-profile social events. With a mission to transform visions into reality, Cedar Elegance Events has built a reputation for delivering bespoke experiences that reflect the unique desires of each client.

Initially a small family-owned business, the company rapidly expanded under the leadership of its founder, Lina Khoury. Her background in hospitality management and her passion for creating unforgettable experiences propelled Cedar Elegance Events to the forefront of the industry. By 2010, the company had successfully organized events across the Middle East, becoming synonymous with elegance, precision, and innovation.

However, in late 2024, Cedar Elegance Events became the target of a sophisticated cyberattack. An unidentified hacking group infiltrated the company’s internal systems, deploying a malicious rootkit known as "Shadow Lurker." This advanced malware was designed to grant unauthorized access to confidential client information, including financial records, guest lists, and proprietary event designs. The rootkit operated covertly, bypassing conventional cybersecurity defenses and compromising sensitive data without immediate detection.

The cyberattack threatened not only the company’s operational integrity but also the trust and confidentiality it had cultivated with its high-profile clientele. In response to this crisis, Cedar Elegance Events reached out to "Byte Builders" to assist in restoring their systems and rebuilding their database.

Byte Builders are helping Cedar Elegance Events rebuild their database, focusing on enhancing data integrity and ensuring the proper organization and recovery of lost records. This collaborative effort aims to restore essential business operations and maintain the high standards Cedar Elegance Events is known for.

**System Overview and Data Management:**

Cedar Elegance Events operates through a centralized and comprehensive database system that manages all aspects of event planning and daily business operations. This system is critical for organizing and retrieving essential information, including client profiles, event details, staff assignments, financial records, and supplier partnerships. The updated database is built using a relational model, ensuring seamless information flow across departments and accurate tracking of business processes.

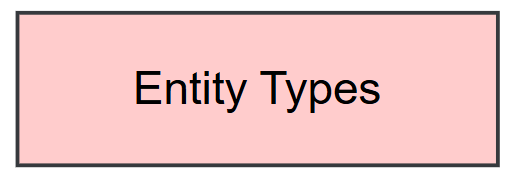
The database includes the following core components: Event data, Venue information, Client profiles, Staff records, Organizer details, Attendee information, Service management, Supplier data, Special offers, and Sponsorship records. Each of these components interact through defined relationships, enabling the system to maintain accurate and comprehensive data tracking.

The updated database enforces several key constraints to maintain data accuracy, consistency, and security. Personal emails must follow the format *@*.\*, while corporate emails must use the domain '@cedarelegance.com'. Primary and foreign keys ensure unique and consistent identification across all records. Payment records undergo regular reconciliation to identify and correct discrepancies. Access to sensitive information is restricted to authorized personnel only. Automated calculations compute staff wages and event costs using predefined formulas. Each event must be uniquely associated with a specific client, while services must be linked to both suppliers and events.

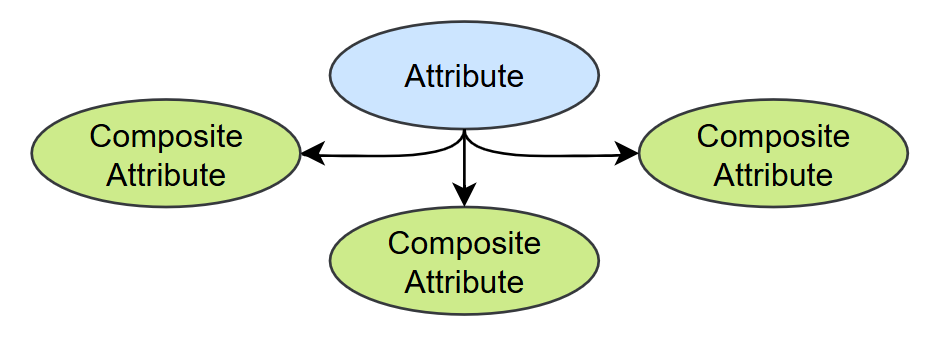
Additionally, the database supports advanced reporting capabilities, including financial summaries, event budget monitoring, and client payment analysis. Real-time data access enhances operational efficiency, ensuring timely responses to client needs and internal demands.

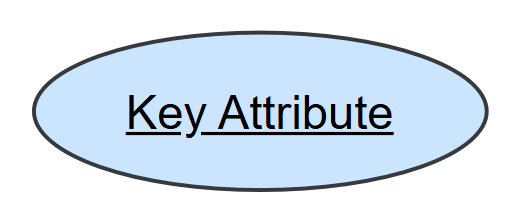
Through its partnership with Byte Builders, Cedar Elegance Events now operates a robust and scalable database that supports future growth while maintaining the highest standards of service delivery and data protection. This modernized system not only resolves past vulnerabilities but also provides a comprehensive framework for managing complex event operations with improved security and operational excellence.

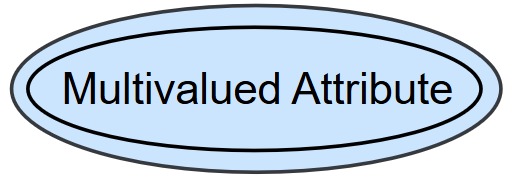
**ER Diagram Symbols:**

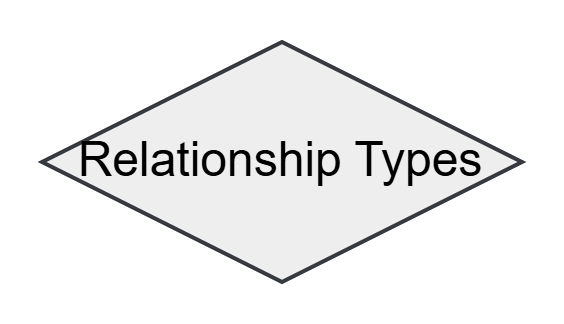


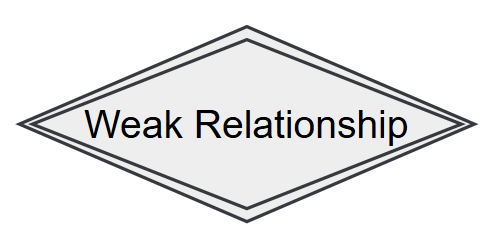






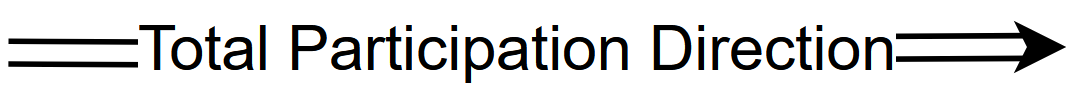










****

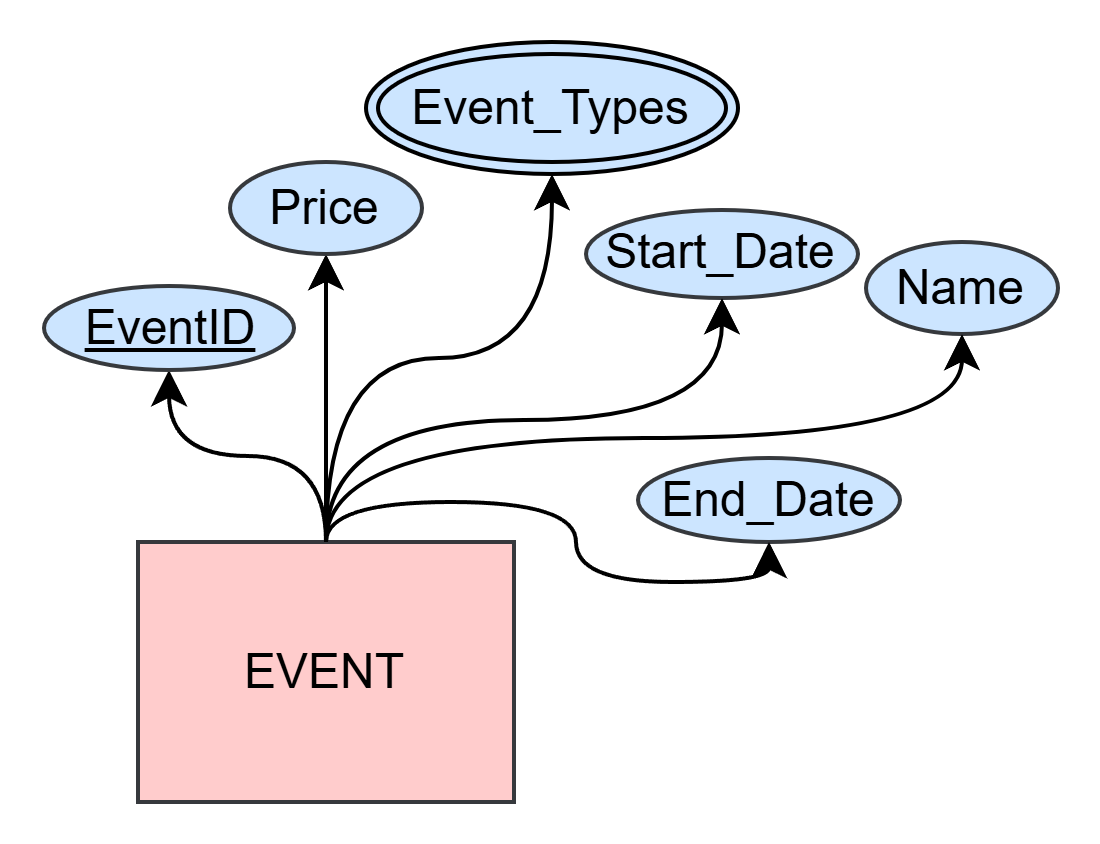
## **ER diagram for Cedar Elegance Events:**

A diagram of a flowchart

Description automatically generated

# **Entity Types:**

1. **EVENT:**

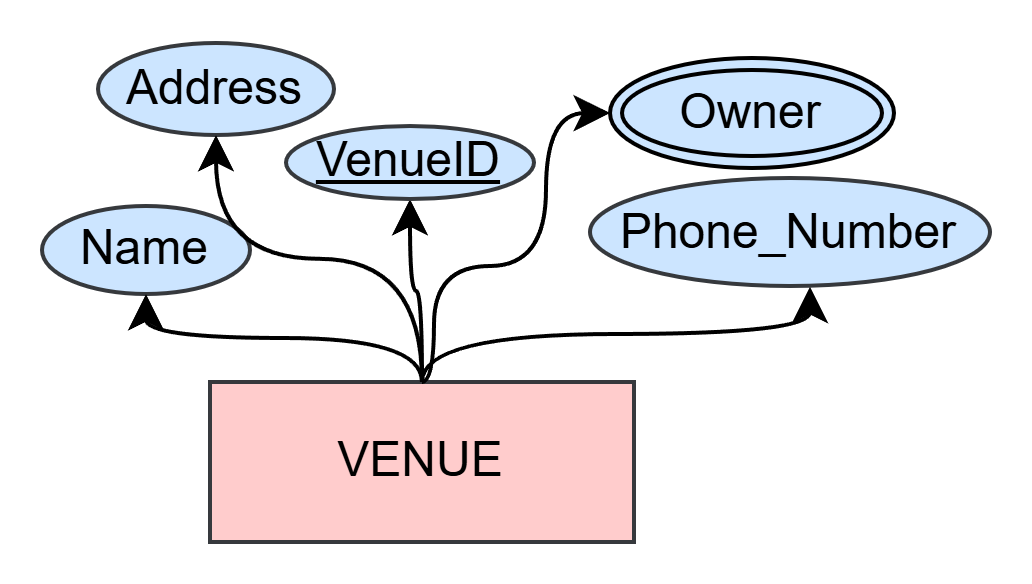


**Description:** Events are scheduled gatherings that take place for a specific purpose that Cedar Elegance Events plans for. They have a defined name, type, price, duration, and an expected number of attendees. Events the bread and butter of what Cedar Elegance Events does.

**Attributes**:

* **EventID**: A unique identifier for each event for record-keeping and reference.
* **Name**: The official title of the event.
* **Start** **Date**: The date and time when the event begins.
* **End** **Date**: The date and time when the event concludes.
* **Event** **Types**: The category or purpose of the event (e.g., wedding, corporate meeting, concert, birthday, graduation).
* **Price**: The total price of the event.

1. **VENUE:**

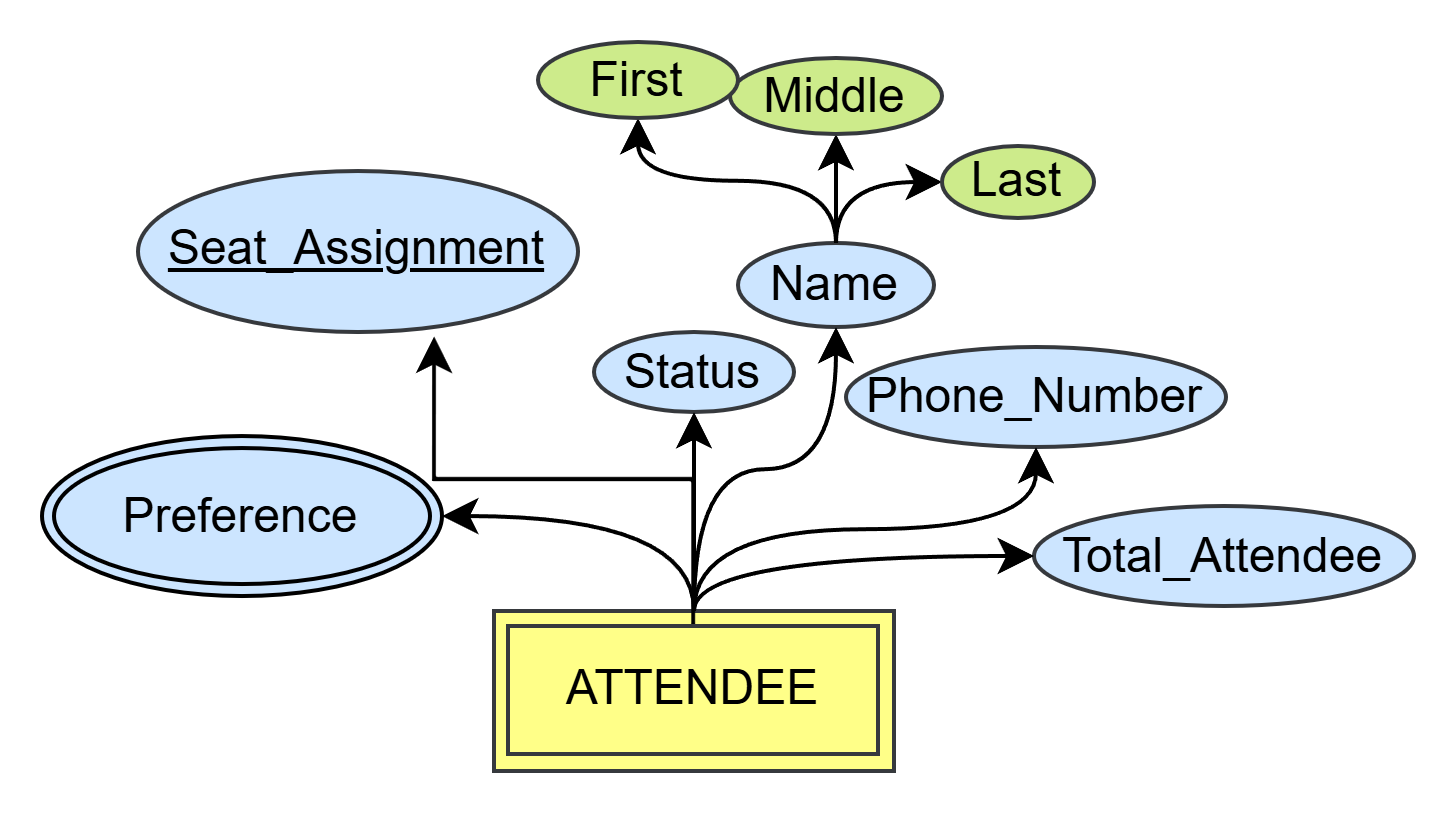


**Description:** A venue refers to the physical location where the events take place. Selecting the right venue is crucial for the success of an event, as it influences guest experience, logistics, and overall event planning. Each Venue has a unique identification such as VenueID, name, address and phone numbers. Venues are where Cedar Elegance Events makes their magic.

Attributes:

* **VenueID**: A unique identifier assigned to each venue for record-keeping and reference.
* **Name**: The official name of the venue.
* **Address**: provides a detailed address information.
* **Owner**: The individual or organization that owns the venue and is responsible for its management and operations.
* **Phone** **Number**: A contact number for inquiries, reservations, and customer support.

1. **ATTENDEE** (Weak entity):

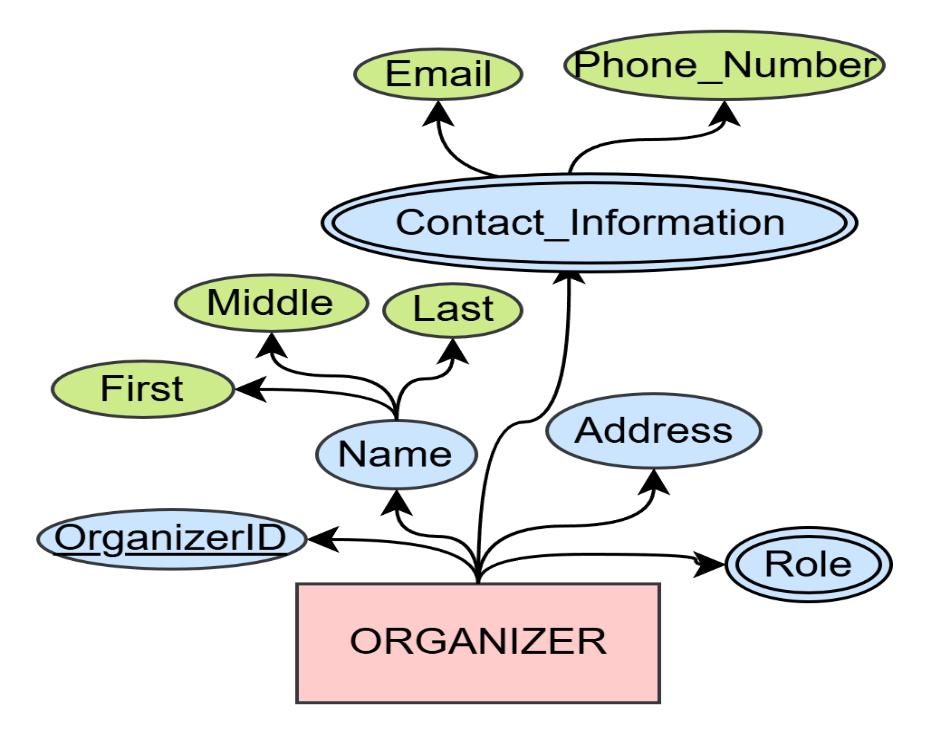


**Description:** Attendees are individuals who participate in an event. They can be invited guests, registered participants, or general audience members. Each attendee has a status indicating their participation, contact details for communication, name, and presences.

**Attributes:**

* **Name:** A composite attribute that consists of:
  + **First Name:** The given name of the attendee.
  + **Middle Name:** The middle name of the attendee (optional).
  + **Last Name:** The family name or surname of the attendee.
* **Status**: The participation status of the attendee (e.g., confirmed, pending, canceled).
* **Phone Number:** The contact number of the attendee.
* **Preference:** Specific requests or choices made by the attendee, such as seating arrangements, meal preferences, or accessibility needs.
* **Total Attendee:** The total amount of attendees attending the event.
* **Seat Assignment**: designates the guest’s assigned seating location at the wedding (ex: "Table-4-Seat-A" or "Section-B-12"). It serves as a **partial key** and is not unique on its own, but becomes unique when combined with the associated EventID.

1. **ORGANIZER:**

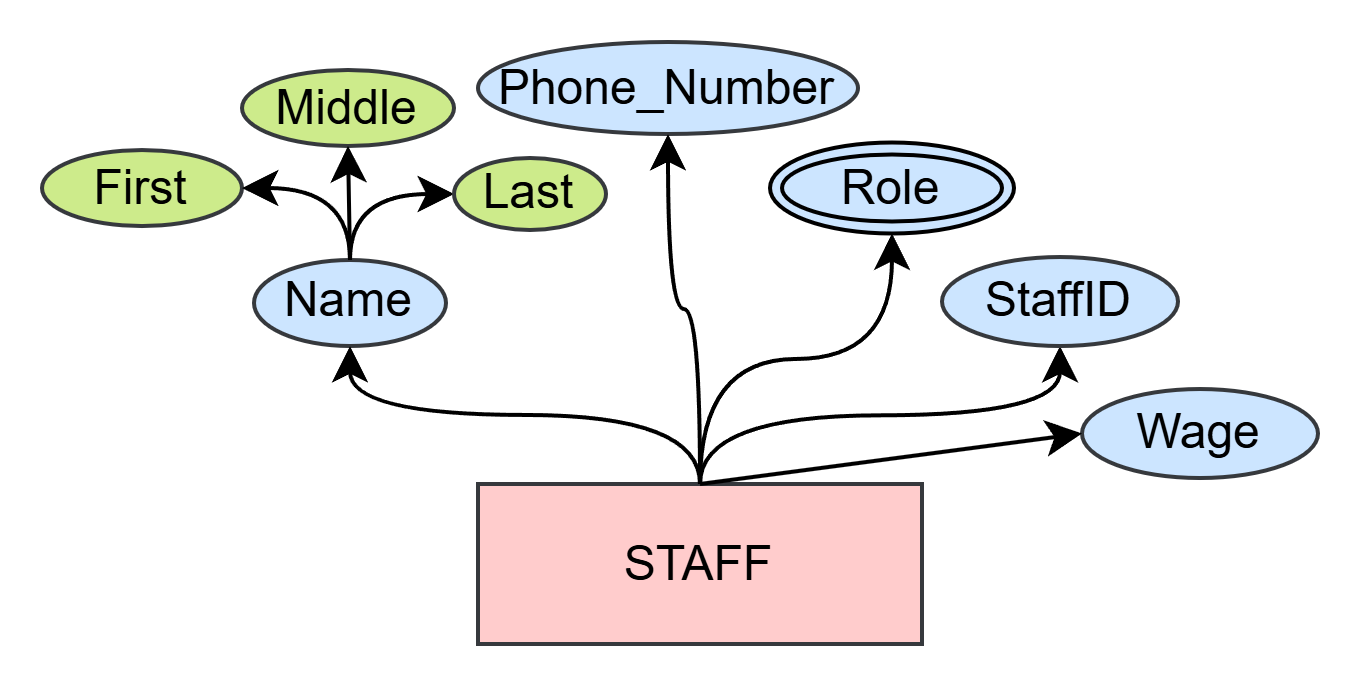


**Description:** Organizers are individuals or groups that can be hired by Cedar Elegance Events or already work for them to manage and coordinate various aspects of an event. They ensure smooth execution by handling logistics, managing staff, and overseeing activities. Organizers play a critical role in making sure the event runs according to plan.

**Attributes:**

* **OrganizerID**: A unique identifier for each organizer for record-keeping and reference.
* **Name**: A composite attribute that consists of:
  + **First Name**: The given name of the organizer.
  + **Middle** **Name**: The middle name of the organizer (optional).
  + **Last** **Name**: The family name or surname of the organizer.
* **Role**: The specific position or responsibility the organizer holds within the event (e.g., coordinator, supervisor).
* **Address**: The physical location or contact address of the organizer.
* **Contact Information**: A multi-valued attribute for communication purposes that includes the attributes:
* **Phone** **Number**: The contact number of the organizer.
* **Email** **Address**: The email address of the organizer.

1. **STAFF:**

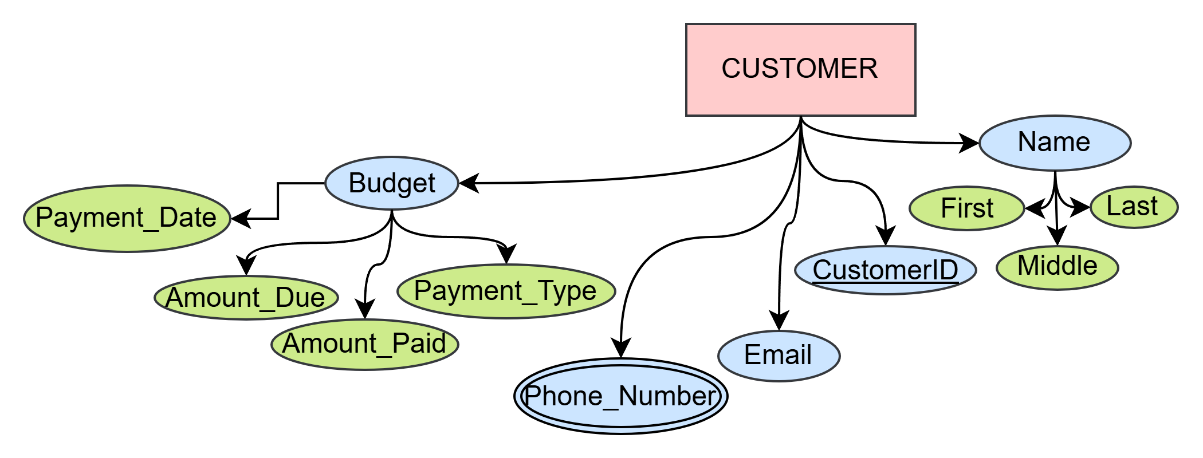


**Description:** Staff members are individuals employed or assigned by Cedar Elegance Events to support and assist during events. They are responsible for various operational tasks, including assisting guests, managing logistics, and ensuring the event runs smoothly.

**Attributes:**

* **StaffID:** A unique identifier for each staff member for record-keeping and reference.
* **Name:** A composite attribute that consists of:
  + **First Name:** The given name of the staff member.
  + **Middle Name:** The middle name of the staff member(optional).
  + **Last Name:** The family name or surname of the staff member.
* **Role:** A multi-valued attribute representing the different positions or responsibilities the staff member may hold during the event (e.g., waiter, security, coordinator).
* **Phone Number:** The contact number of the staff member for communication purposes.
* **Wage**: The amount of money a staff member gets paid.

1. **CUSTOMER:**

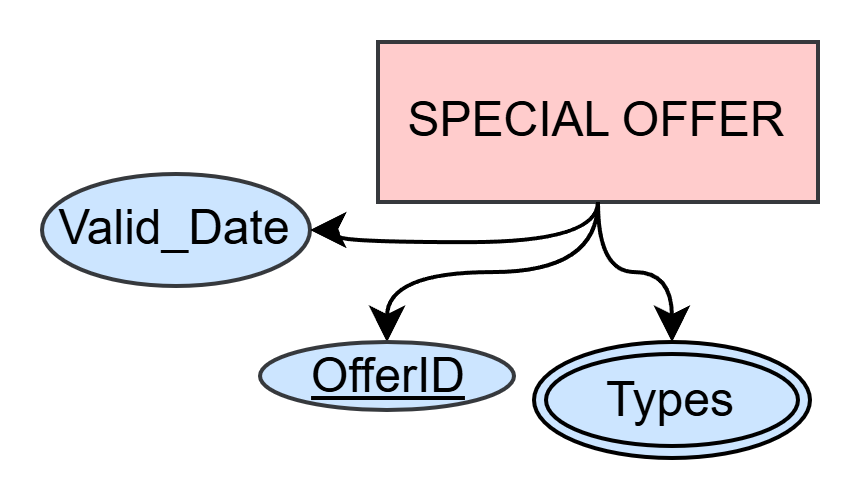


**Description:** A customer represents an individual who books an event to be organized by Cedar Elegance Events. They can choose a special offer and pay the planners for the event they want to book.

**Attributes:**

* **CustomerID:** A unique identifier assigned to each customer for record-keeping and reference.
* **Name:** A composite attribute that consists of:
  + **First Name:** The given name of the customer.
  + **Middle Name:** The middle name of the customer (optional).
  + **Last Name:** The family name or surname of the customer.
* **Email:** The primary email address used for communication and notifications.
* **Phone Number:** A contact number for communication and support.
* **Budget:** A composite and derived attribute detailing the financial aspects of the customer’s transaction, including:
  + Payment Method: The method the customer uses to pay for the event.
  + Amount Due: The remaining balance to be paid by the customer.
  + Amount Paid: The amount already paid by the customer.
  + Payment Date: The date when the customer is meant to pay.

## **SPECIAL OFFERS**

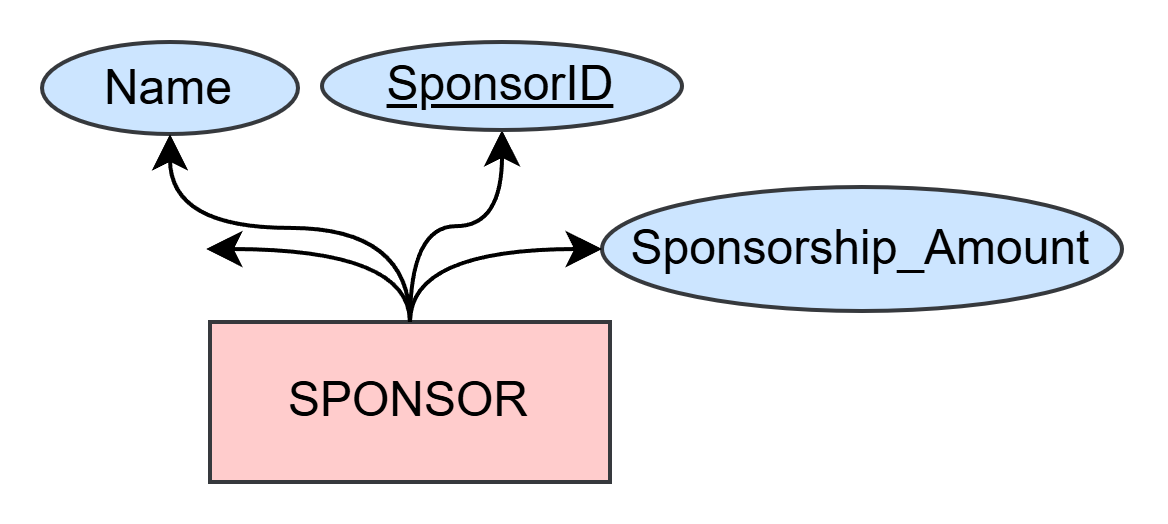


**Description:** Special offers refer to promotional deals, discounts, and incentives provided for events. These offers enhance customer engagement, encourage early bookings, and provide cost-saving opportunities for customers.

**Attributes:**

* **OfferID:** A unique identifier assigned to each special offer for record-keeping and reference.
* **Valid Date:** The expiration or validity period of the offer.
* **Types:** A classification of the offer which helps manage various promotional strategies and ensures that discounts and offers are systematically applied to events.

1. **SPONSOR:**

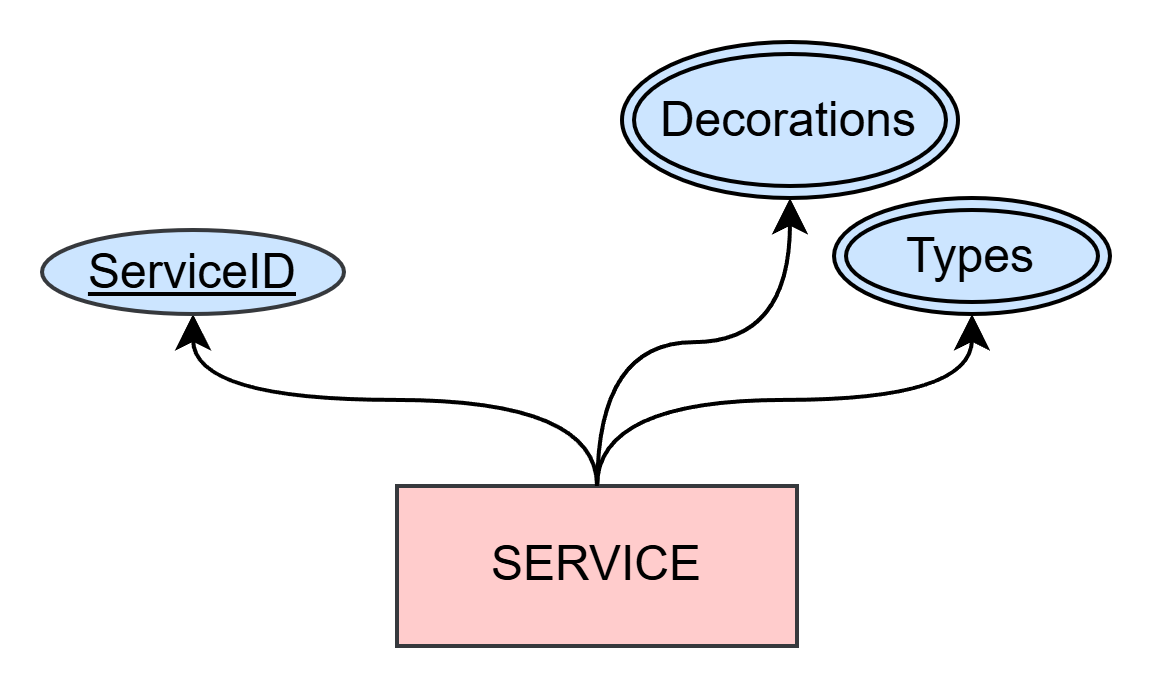


**Description**: Sponsors are individuals or organizations that provide financial or material support for an event. Their contributions help cover event expenses and enhance the overall experience. Sponsorships are essential for large-scale or high-profile events. Their unique

**Attributes**:

* **SponsorID**: A unique identifier assigned to each sponsor for record-keeping and reference.
* **Name**: The official name of the sponsor (individual or organization).
* **Phone** **Number**: A contact number for communication with the sponsor.
* **Sponsorship** **Amount**: The total amount or value of the support provided by the sponsor.

1. **SERVICE:**

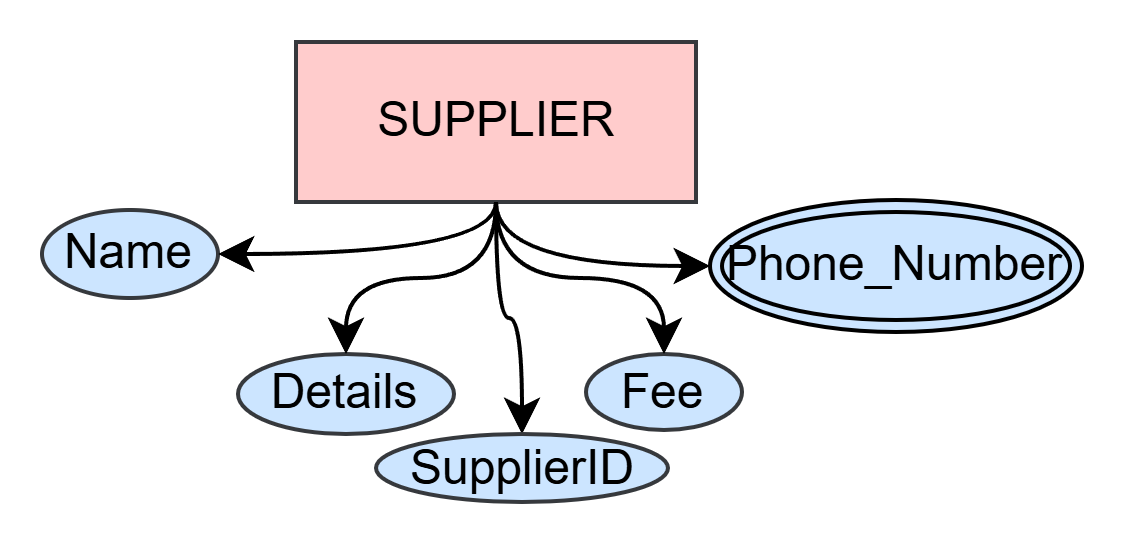


**Description**: Services refer to the various offerings required to execute an event successfully. These can include decorations, catering, entertainment, and other specialized services provided by vendors.

Attributes:

* **ServiceID**: A unique identifier assigned to each service for tracking and reference.
* **Types**: The category of service offered (e.g., catering, photography, decoration).
* **Decorations**: Specific details about the decoration service, if applicable.

**10. SUPPLIER:**



**Description**: Suppliers are external entities that provide goods or services needed for the event. They play a crucial role in delivering necessary items, ensuring smooth event operations.

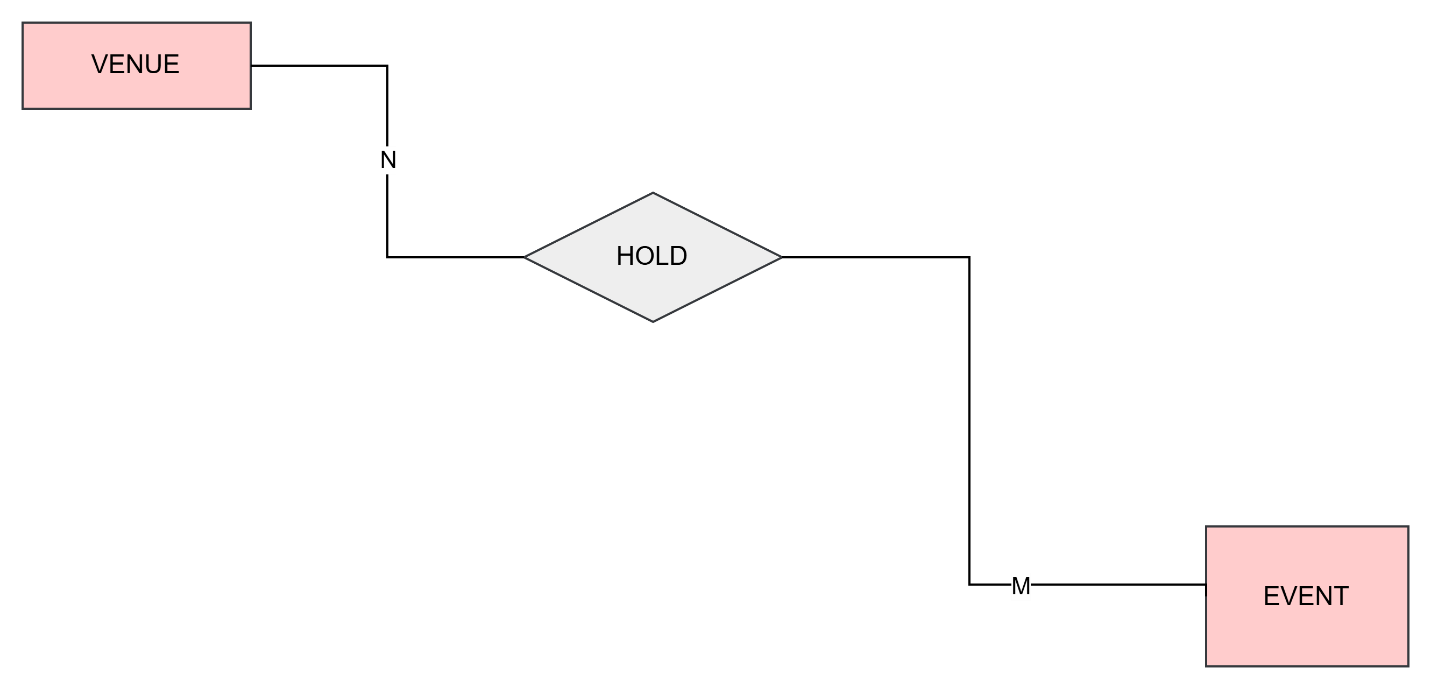
Attributes:

* **SupplierID**: A unique identifier assigned to each supplier for tracking and reference.
* **Name**: The official name of the supplier it could be either the company name or the owner's.
* **Phone** **Number**: A contact number for communication with the supplier.
* **Details**: Information about the goods or services provided by the supplier.

# **Relationships Between Entity Types:**

The relationships between these entities will be very important for the functionality of the entities. Here are some entity relationships:

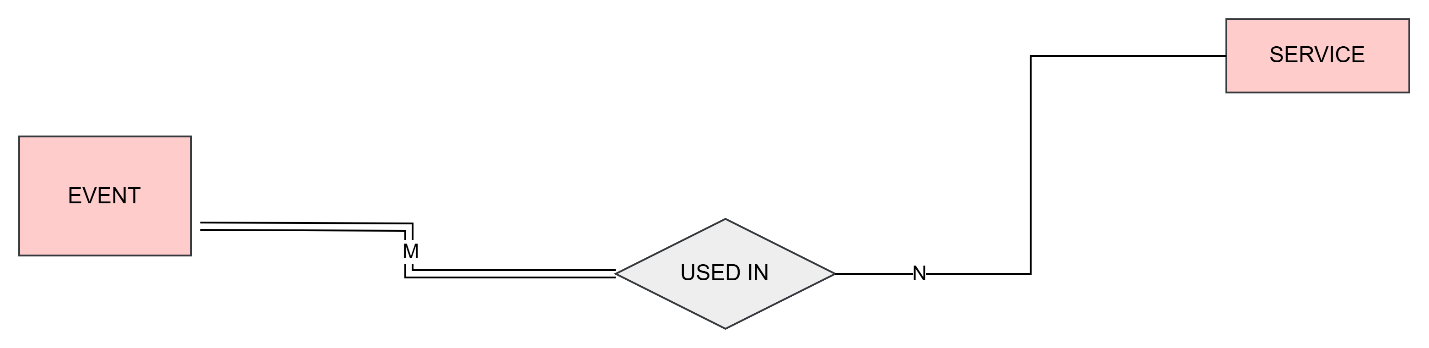
* HOLD (N:M):



EVENTS are held at a VENUE, and a VENUE can host multiple EVENTS over time. Thus, a HOLD relation was made between the entity EVENTS and entity VENUE. This relationship is many-to-many (N:M). For example, a conference hall may be used for a business seminar today and a wedding reception tomorrow. This structure allows for effective venue utilization and scheduling. Another example is that there are multiple EVENTS happening at multiple VENUES.

EVENTS may HOLD many VENUES and many VENUES may HOLD EVENTS. Additionally, there is a **PARTIAL RELATION** in which some EVENTS may be linked to a specific VENUE only under certain conditions or constraints, such as limited availability or specialized requirements for the EVENT. This partial relation helps in refining scheduling constraints and managing exceptions more effectively.

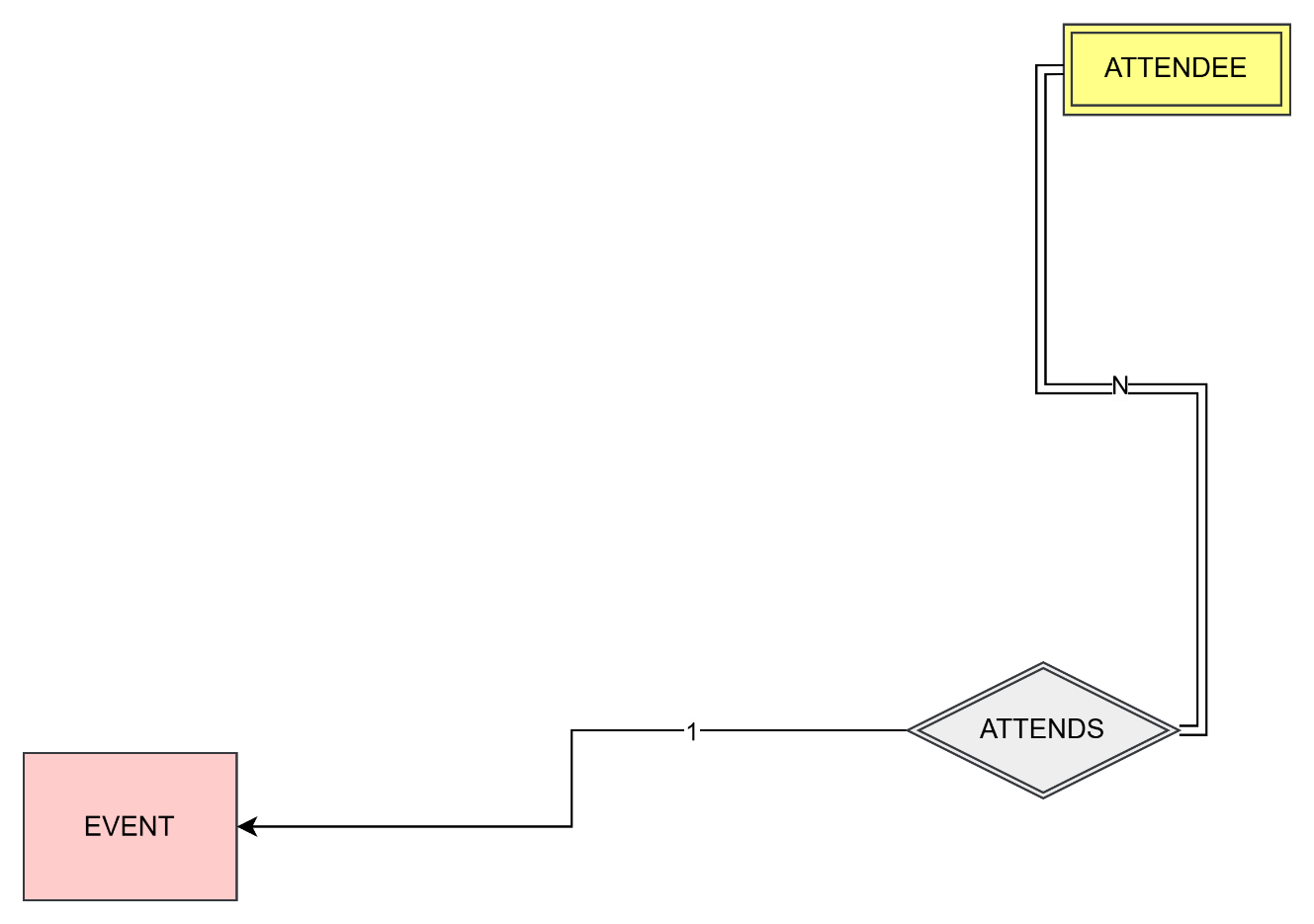
* USED IN(M:N):



An EVENT can require multiple SERVICES (e.g., catering, decorations, security), and many SERVICES can be USED IN multiple events. Therefore, the relation between the EVENT entity and the SERVICES entity was formed called USED IN. This relationship is many-to-many (M:N). For example, a wedding may require catering, music, and photography services, while the same catering SERVICE may be used for corporate events, conferences, and birthday parties. This ensures efficient resource allocation and service availability for different EVENTS. The participation for EVENTS is total but for SERVICES its partial.

SERVICES may be Used IN EVENTS but EVENTS are not USED IN SERVICES instead EVENTS have SERVICES.

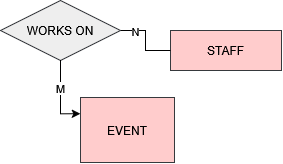
* ATTENDS (1:N):



Multiple ATTENDEES can participate in an EVENT. Thus, the relation called ATTENDS was made for the weak entity ATTENDEES and the entity EVENT. This relationship is one-to-many (1:M). For example, A wedding can have the mother, father and other family members attending the weddings. This relationship ensures accuracy in event participation.

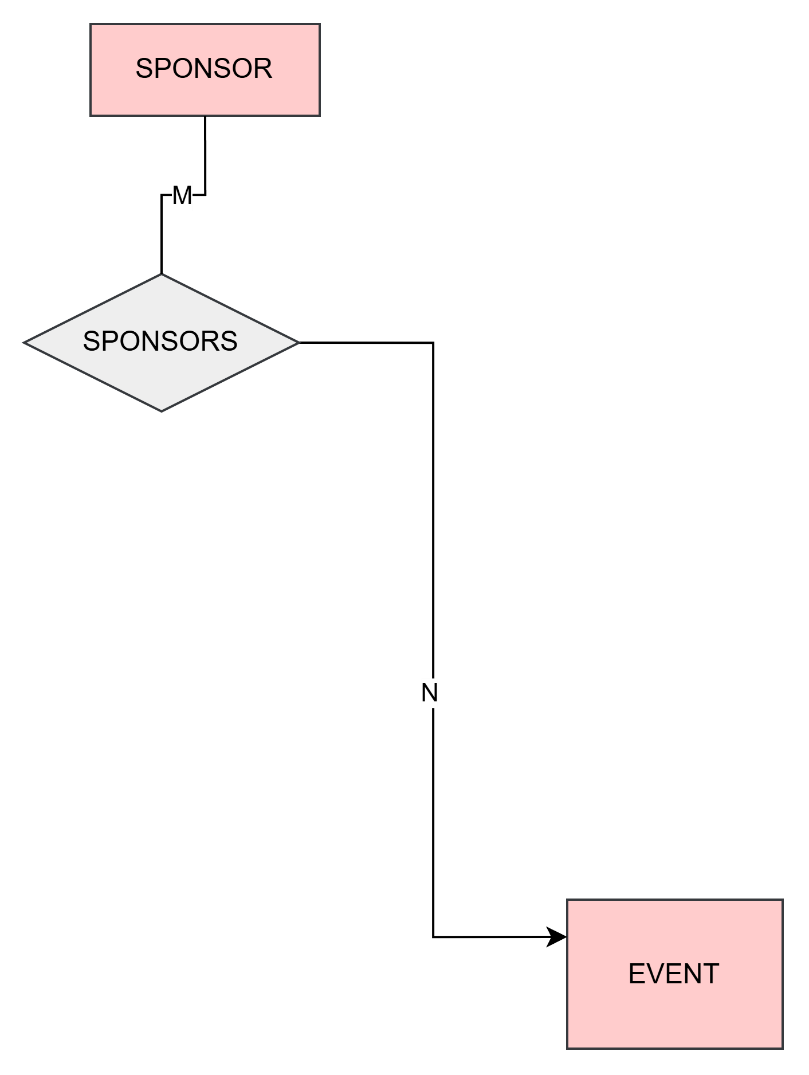
ATTENDEES May ATTEND an EVENT while an EVENT has many ATTENDEES.

* WORKS ON (M:N)



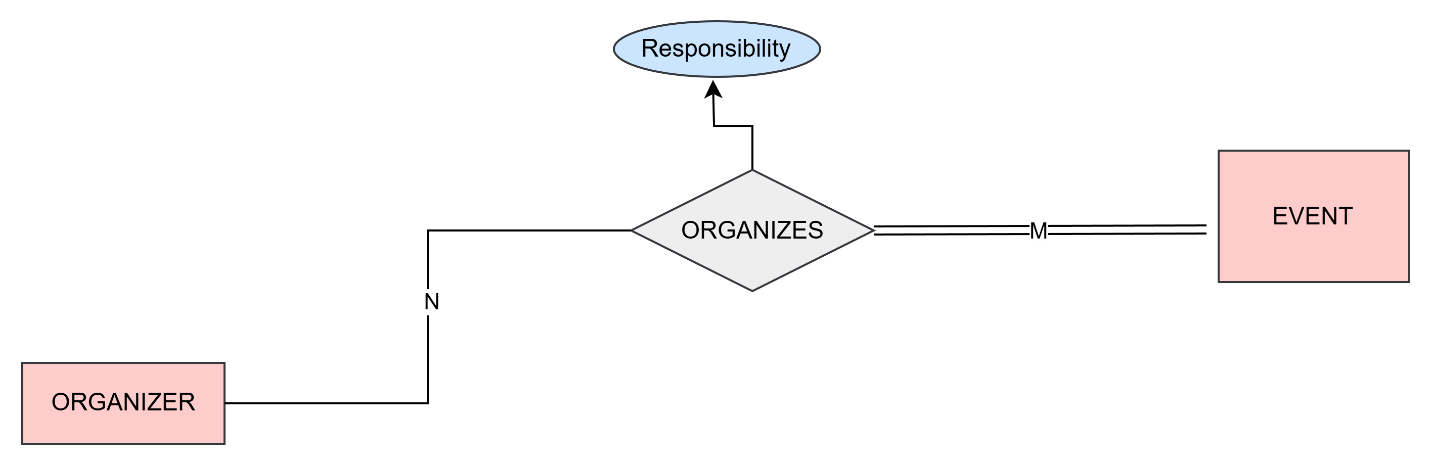
STAFFS WORKS ON multiple EVENTS, and an EVENT may require multiple STAFFS for different roles. A relation called WORKS ON was made for the entities STAFFS and EVENTS. This relationship is many-to-many (M:N). For example, a lighting technician might work at a concert one day and a corporate seminar the next. Similarly, an event might need security personnel, catering staff, and managers working together. This allows for efficient staffing based on expertise and availability. Additionally, there is a **PARTIAL RELATION** where certain STAFFS may only work on specific EVENTS based on their skillset, availability, or event requirements.

* SPONSORS (M:N)



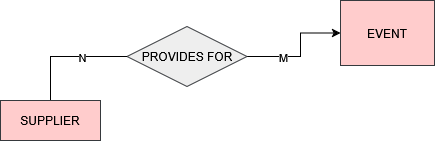
EVENTS can have multiple SPONSORS, and many SPONSORS SPONSORS EVENTS. Thus, the relation between the entity SPONSOR and EVENTS entity was called SPONSORS. This relationship is many-to-many (M:N). For example, a beverage company may sponsor both a sports event and a music festival, while a tech company may sponsor a business conference and a startup pitch competition. This ensures that a SPONSOR can reach their target audience effectively while EVENTS secure the necessary funding. Additionally, there is a **PARTIAL RELATION** where some SPONSORS may only be linked to specific EVENTS under certain conditions, such as the sponsor's niche interests or specific EVENT criteria. This partial relation helps to handle exceptions and manage unique SPONSOR - EVENT matches effectively.

* ORGANIZES (M:N)



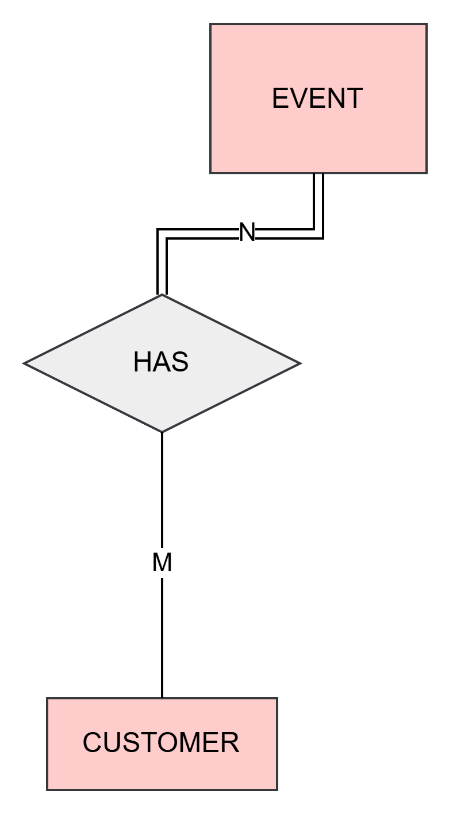
EVENTS are managed by one or more ORGANIZERS, and ORGANIZERS ORGANIZES multiple EVENTS. Thus, the relation called ORGANIZES was mode for the entity EVENTS and entity ORGANIZERS. This relationship is many-to-many (M:N). For example, an event planning company may handle corporate EVENTS, concerts, and weddings, while a charity organization ORGANIZES multiple fundraising EVENTS annually. This ensures EVENTS are professionally managed by experienced ORGANIZERS. Moreover, this relation has the attribute Responsibility This attribute can specify what specific responsibility or role the organizer has in relation to the event, such as 'coordinator', 'financial manager', or 'logistics'.

* PROVIDES FOR (M:N):



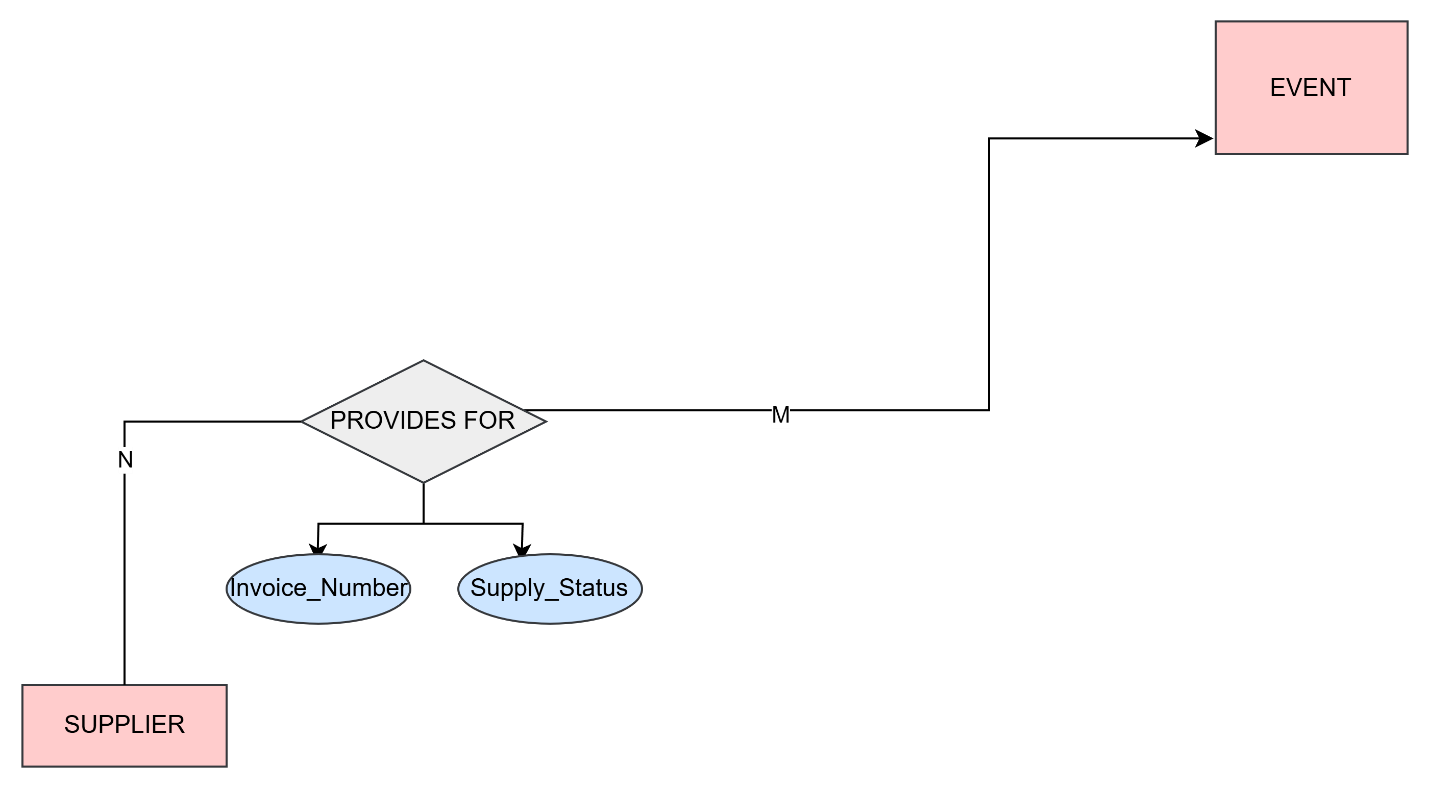
A SUPPLIER PROVIDES FOR multiple EVENTS, and EVENTS may require multiple SUPPLIERS for different needs. Therefore, the relationship called PROVIDES FOR was made for the entity SUPPLIER and the entity EVENTS. This relationship is many-to-many (M:N). For example, a flower SUPPLIER might provide decorations for weddings and gala dinners, while an audiovisual SUPPLIER may provide equipment for business conferences and concerts. This allows events to source materials from various specialized vendors. Additionally, there is a **PARTIAL RELATION** where some SUPPLIERS may only provide for specific EVENTS under certain conditions, such as EVENT type or SUPPLIER availability. This partial relation helps manage exceptions where certain SUPPLIERS are not needed for all events but are crucial for certain specialized requirements. SUPPLIERS PROVIDES FOR EVENTS, and EVENTS only get supplies from SUPPLIERS.

* HAS (M:N):



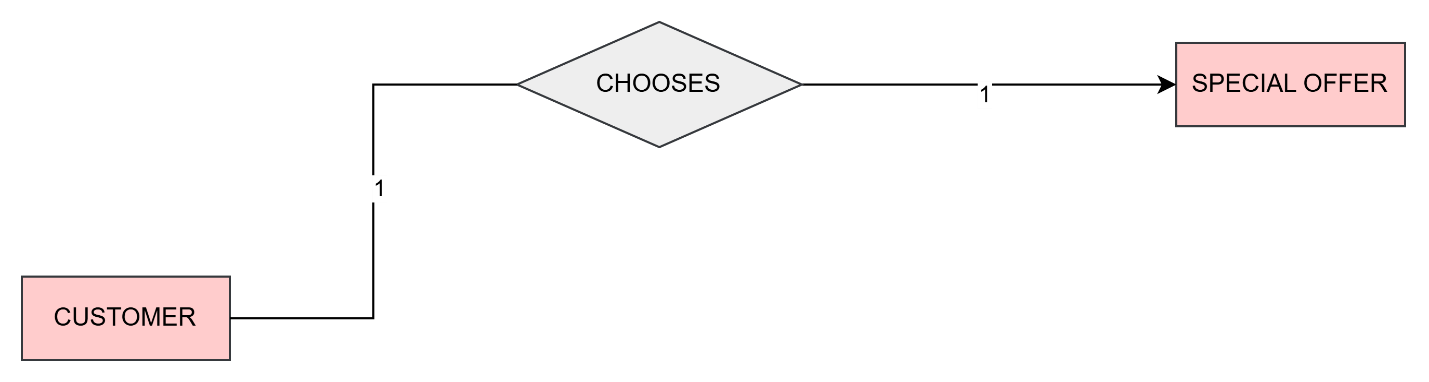
An EVENT HAS CUSTOMERS (such as ticket buyers or clients), and a CUSTOMERS may participate in a number of EVENTS. A relationship between the entities EVENT and CUSTOMERS were made called HAS. This relationship is many-to-many (M:N). For example, an engaged person could book for a wedding event and an outdoor party after the event. Moreover, a big public EVENT could have multiple CUSTOMERS pinching in to make the EVENT. This ensures that Cedar Elegance Events can provide multiple events to multiple customers. CUSTOMERS Have multiple EVENT and an EVENT has many CUSTOMERS.

* PROVIDES (M:N):

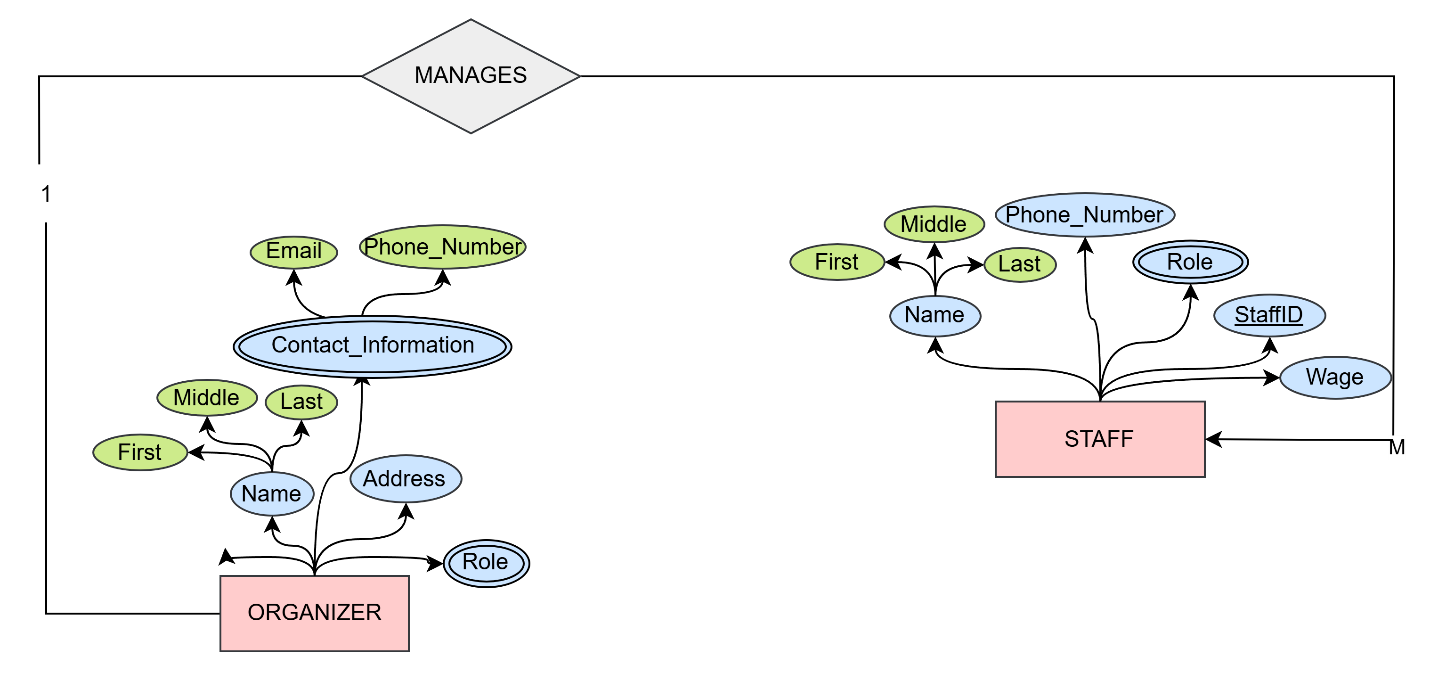


A SUPPLIER PROVIDES multiple SERVICES (such as catering, lighting, or audiovisual support), and SERVICES can be provided by multiple SUPPLIERS. Thus, a relation to SUPPLIER entity and SERVICES entity was made called PROVIDES. This relationship is many-to-many (M:N). For example, a catering company might offer food SERVICES for various events, while multiple SUPPLIERS could provide decoration SERVICES. This ensures that events have access to a diverse range of SERVICE providers that best meet their specific needs. Moreover, this relation has two attributes one is Invoice Number which tracks payments and references for financial records and the second one is Supply Status which checks if the supply is being delivered, pending or cancelled. Additionally, there is a **PARTIAL RELATION** where some SUPPLIERS may only provide specific SERVICES under certain conditions, such as their specialization or availability. For instance, a supplier might specialize only in audiovisual services and not in catering, or a decoration supplier may only offer certain types of decorations. This **PARTIAL RELATION** ensures that the right SUPPLIERS are matched to the right SERVICES for an event’s requirements. SUPPLIER may PROVIDE multiple SERVICES and SERVICES come from SUPPLIERS.

* CHOOSES (1:1):



Many CUSTOMERS need SPECAIL OFFER to lower the price thus a relation called **CHOOSES** was made between CUSTOMER and SPECAIL OFFER. A CUSTOMER can choose only one SPECAIL OFFER, and each SPECAIL OFFER is associated with only one CUSTOMER. This relationship is one-to-one (1:1). For example, if a CUSTOMER selects a "10% early bird discount" for an event, they cannot choose another SPECAIL OFFER, for the same event. However, SPECAIL OFFERS can’t CHOOSE a CUSTOMER. Additionally, there is a **PARTIAL RELATION** where some CUSTOMERS may not be eligible for certain SPECAIL OFFERS based on their status, event type, or other restrictions. This partial relation allows for more flexibility in managing offers and ensuring that only eligible CUSTOMERS are linked to specific discounts. However, SPECAIL OFFERS can’t **CHOOSE** a CUSTOMERS.

MANAGES (1:M): 

An ORGANIZER is responsible for managing multiple STAFF members thus a MANAGES relation was made, but each STAFF member reports to only one ORGANIZER. This relation is one to many (1:M). This structure ensures clear hierarchy and efficient delegation of tasks. For example, an event planning company may have one ORGANIZER overseeing different employees, assigning them roles in various events. Additionally, there is a **PARTIAL RELATION** where some STAFF members may report to different ORGANIZERS based on event needs, shifts, or special roles. This partial relation helps to account for exceptions where specific STAFF might work under multiple ORGANIZERS for particular projects or events. ORGANIZERS may MANAGES STAFF but STAFF can't MANAGE **ORGANIZERS**.

# ER to Relational Mapping Algorithms

After designing the ER schema and having displayed the database for Cedar Elegance Events as a system of entities, attributes, and relationships, this high level design must be translated into a relational database design. In order to map the ER design to a relational database design, a seven step algorithm needs to be followed. Relational database design ultimately produces a set of relations. The implicit goals of the design activity are information preservation and minimum redundancy. The following is a detailed description on applying the different steps to our database design. The steps in brief are as follows:

* **Step 1:** All regular entity types are mapped into relations schemas. By regular, we mean that only non­weak entities will be mapped in this step. For every regular entity, only the simple attributes are encoded into the relation schemas. Composite attributes are broken down into their simple attribute components. Multivalued and derived attributes are not encoded in this step. Multivalued attributes will be added in the Step 6.
* **Step 2:** All weak entity types are mapped into relation schemas. As in Step 1, only the simple attributes are encoded into the relation schemas. Composite attributes are broken down into their simple attribute components. Multivalued and derived attributes are not encoded in this step. Multivalued attributes will be added in the Step 6
* **Step 3:** All binary 1:1 relationship types are mapped into relation schemas. Specifically, in this step, we apply the foreign key approach where we choose the entity on the total participation side of the relation, then we add as a foreign key the primary key of the other entity participating in this relation.
* **Step 4:** All binary 1:N relationship types are mapped into relation schemas. As in Step 3, we apply the foreign key approach. We add a foreign key in the entity type at the many sides of the relationship. This foreign key will be the primary key of the other entity type participating in this relationship.
* **Step 5:** All binary M:N relationship types are mapped into relation schemas. Unlike Step 3 and 4, we encode the relationships by creating a new relation which includes, as foreign keys, the primary keys of all participating relations. Their combination will form the primary key of this newly created relation.
* **Step 6:** All multivalued attributes that were left over from the previous steps are mapped. Specifically, a relation is created for every multivalued attribute. It will contain the primary key of each entity has a multi­valued attribute.
* **Step 7:** All N­ary relationship types are mapped in this step. But, there are no N­ary relationships in the database so nothing is done for this step in that database system.

## STEP 1: Mapping of Regular Entity Types

In the first step, the regular entity types must be mapped into relations. Each regular entity is going to have its own relation that includes all of its simple attributes and a single primary key which is underlined. The regular (strong) entities in this database design for Cedar Elegance Events are: EVENT, STAFF, SPONSOR, VENUE, SERVICE, SPECIAL OFFER, CUSTOMER, SUPPLIER, and ORGANIZER.

1. **EVENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EventID | **Price** | **Start\_Date** | **End\_Date** | **Name** |

The EVENT entity contains simple, and multivalued attributes. The multivalued attribute Event\_Types is not represented in this relation. This relation only includes all simple attributes and the primary key EventID which is underlined. The other attributes of EVENT entity are:

* Price
* Start\_Date
* End\_Date
* Name

1. **STAFF**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| StaffID | **Phone\_Number** | **Wage** | **First** | **Middle** | **Last** |

The STAFF entity contains simple, composite, and multivalued attributes. The multivalued attribute Role is not represented in this relation. The STAFF entity has Name as a composite attribute of which only the simple attributes: First, Last, and Middle are included in the relation. This relation only includes all simple attributes and the primary key StaffID which is underlined. The other attributes of STAFF entity are:

* Phone\_Number
* Wage

1. **SPONSOR**

|  |  |  |  |
| --- | --- | --- | --- |
| SponsorID | **Name** | **Phone\_Number** | **Sponsorship\_Amount** |

The SPONSOR entity contains only simple attributes. This relation only includes all simple attributes and the primary key SponsorID which is underlined. The other attributes of EVENT entity are:

* Name
* Phone\_Number
* Sponsorship\_Amount

1. **VENUE**

|  |  |  |  |
| --- | --- | --- | --- |
| VenueID | **Name** | **Phone\_Number** | **Address** |

The VENUE entity contains simple, and multivalued attributes. The multivalued attribute Owner is not represented in this relation. This relation only includes all simple attributes and the primary key VenueID which is underlined. The other attributes of VENUE entity are:

* Name
* Phone\_Number
* Address

1. **SERVICE**

|  |
| --- |
| ServiceID |

The SERVICE entity contains multivalued attributes only. This relation only includes the primary key ServiceID which is underlined. SERVICE entity doesn’t have any other simple attributes.

1. **SPECIAL OFFER**

|  |  |
| --- | --- |
| OfferID | **Valid\_Date** |

The SPECIAL OFFER entity contains simple, and multivalued attributes. The multivalued attribute Types is not represented in this relation. This relation only includes the simple attribute and the primary key VenueID which is underlined. The other attribute of VENUE entity is:

* Valid\_Date

1. **CUSTOMER**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CustomerID | **Email** | **First** | **Middle** | **Last** | **Payment\_Types** | **Amount\_Paid** | **Amount\_Due** | **Payment\_Day** |

The CUSTOMER entity contains simple, multivalued, and composite attributes. The multivalued attribute Phone\_Number is not represented in this relation. This relation only includes all simple attributes and the primary key CustomerID which is underlined. The CUSTOMER entity has Name and Budget as a composite attribute of which only the simple attributes: First, Last, and Middle for Name and Payment\_Types, Amount\_Paid, Amount\_Due, and Payment\_Day are included in the relation. The other attribute of CUSTOMER entity is:

* Email

1. **SUPPLIER**

|  |  |  |  |
| --- | --- | --- | --- |
| SupplierID | **Name** | **Details** | **Fee** |

The SUPPLIER entity contains simple, and multivalued attributes. The multivalued attribute Phone\_Number is not represented in this relation. This relation only includes the simple attribute and the primary key SupplierID which is underlined. The other attribute of SUPLIER entity is:

* Name
* Details
* Fee

1. **ORGANIZER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrganizerID | **Address** | **First** | **Middle** | **Last** |

The ORGANIZER entity contains simple, and multivalued attributes. The multivalued attributes Role and Contact\_Information are not represented in this relation. The ORGANIZER entity has Name as a composite attribute of which only the simple attributes: First, Last, and Middle are included in the relation. This relation only includes the simple attribute and the primary key SupplierID which is underlined. The other attribute of SUPLIER entity is:

* Address

## STEP 2: Mapping of Weak Entity Types

In this step, the weak entity types are mapped into relations. As in Step 1, only the simple attributes are included in the relations and not multivalued or derived attributes. Furthermore, weak entity relations have a foreign key attribute which is the primary key of the owner entity type. The combination of the foreign key added, and the partial key of the weak entity type represents the primary key of the relation. The weak entity in our database design is: ATTENDEE.

1. **ATTENDEE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Status** | **Total\_Attendee** | **First** | **Middle** | **Last** |

The weak entity ATTENDEE contains simple, composite, and multivalued attributes. The multivalued attribute Preferences is not included in the relation. The ATTENDEE entity has Name as a composite attribute of which only the simple attributes: First, Last, and Middle are included in the relation. Moreover, the EventID, the primary key of the owner entity EVENT, is included. EventID and the partial key Attendant\_Number are combined to represent the primary key of this relation. The other attributes of ATTENDEE entity are:

* Status
* Total\_Attendee

## STEP 3: Mapping of Binary 1:1 Relationship Types

In this step, we are going to map the binary one-to­-one relationships. To accomplish our goal, we can follow one of three approaches. The first approach, called foreign key approach, is where we choose the entity on the total participation side of the relation, then we add as a foreign key the primary key of the other entity participating in this relation. The second approach, called merged relation approach, is where we merge the two entities participating in the relationship into a single relation. This is only used when both participations are total and thus not useful in our case. The third approach, called cross-reference or relationship relation approach, is where we create a third relation which will include the primary keys of both entities participating in the relationship. We are going to follow the foreign key approach because it is the most useful in our case. The binary one-to-one relationship that needs to be mapped is: CHOOSES

1. **CHOOSES**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **SO\_ID** | **Email** | **First** | **Middle** | **Last** | **Payment\_Types** | **Amount\_Paid** | **Amount\_Due** | **Payment\_Day** |

The entity CUSTOMER contains simple attributes, composite attributes, and one multivalued attribute. Indeed, the multivalued attribute will not be shown in this table here. The CUSTOMER entity has Name and Budget as a composite attribute of which only the simple attributes: First, Last, and Middle for Name and Payment\_Types, Amount\_Paid, Amount\_Due, and Payment\_Day are included in the relation. The simple attribute is:

* Email

The “CHOOSES” relationship was created as a link between CUSTOMER and SPECIAL OFFER. On both sides, participation is partial; thus it becomes our choice on where to place the foreign key. We chose to place the foreign key on the CUSTOMER relation in which we added, as a foreign key, the primary key OfferID of the SPECIAL OFFER entity and we renamed it SO\_ID.

## STEP 4: Mapping of Binary 1:N Relationship Types

In this step, we are going to map the binary one­-to-­many relationships. We add a foreign key in the entity type at the many sides of the relationship. This foreign key is the primary key of the other entity type participating in this relationship. We must also include any other simple attribute of the one-­to-­many relationship. The one-­to-­many relationships that need to be mapped are: ATTENDS, and MANAGES

1. **ATTENDEE (ATTENDS):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Status** | **Total\_Attendee** | **First** | **Middle** | **Last** |

Every EVENT has one or more ATTENDEE who attend it. That is why the “Attends” relationship was created. The relationship links the EVENT entity and the ATTENDEE entity. The ATTENDEE entity is on the “many” side. Thus, we add to its relation the foreign key EventID which is the primary key of the EVENT entity.

1. **STAFF (MANAGES):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StaffID | **OrganizerID** | **Phone\_Number** | **Wage** | **First** | **Middle** | **Last** |

Every ORGANIZER has one or more STAFF members to manage. That is why the “MANAGES” relationship was created. The relationship links the ORGANIZER entity and the STAFF entity. The STAFF entity is on the “many” side. Thus, we add to its relation the foreign key OrganizerID which is the primary key of the ORGANIZER entity.

## STEP 5: Mapping of M:N Relationship Types

In this step, we are going to map the binary many-to-many relationships. For each many-to-many relationship we are going to create a new relation which includes, as foreign keys, the primary keys of all participating relations. Their combination will form the primary key of this newly created relation. We must also include any other simple attribute of the many­-to­-many relationship. The many-to-many relationships needed to be mapped are: HOLD, USED IN, WORKS ON, SPONSORS, ORGANIZES, PROVIDES FOR, HAS, and PROVIDES.

1. **HOLDS**

|  |  |
| --- | --- |
| **EventID** | **VenueID** |

Every VENUE “Holds” at least one EVENT. An EVENT can be held at many VENUES, and one VENUE can hold many EVENTS. The “Holds” relationship links the EVENT entity and the VENUE entity. We create a new relation called “Holds” that includes the primary keys of the EVENT and VENUE entities. The primary key of the EVENT entity, EventID, is added to the “Hold” relation. Also the primary key of the Venue entity, VenueID, is added. The combination of both added keys represents the primary key of the “Holds” relation and they are consequently underlined.

1. **USED IN**

|  |  |
| --- | --- |
| **EventID** | **ServiceID** |

Every SERVICE is “Used in” at least one EVENT. An EVENT can use many SERVICES, and one SERVICE can be used in many EVENTS. The “Used in” relationship links the EVENT entity and the SERVICE entity. We create a new relation called “Used in” that includes the primary keys of the EVENT and SERVICE entities. The primary key of the EVENT entity, EventID, is added to the “Used in” relation. Also the primary key of the SERVICE entity, ServiceID, is added. The combination of both added keys represents the primary key of the “Used in” relation and they are consequently underlined.

1. **WORKS ON**

|  |  |
| --- | --- |
| **StaffID** | **EventID** |

Every STAFF member “Works on” at least one EVENT. An EVENT have many STAFF members working on it, and one STAFF can work on many EVENTS. The “Works on” relationship links the EVENT entity and the STAFF entity. We create a new relation called “Works on” that includes the primary keys of the EVENT and STAFF entities. The primary key of the EVENT entity, EventID, is added to the “Works on” relation. Also the primary key of the STAFF entity, StaffID, is added. The combination of both added keys represents the primary key of the “Works on” relation and they are consequently underlined.

1. **SPONSORS**

|  |  |
| --- | --- |
| **SponsorID** | **EventID** |

Every SPONSOR member “Sponsors” at least one EVENT. An EVENT has many SPONSORS sponsoring on it, and one SPONSOR can sponsor many EVENTS. The “Sponsors” relationship links the EVENT entity and the SPONSOR entity. We create a new relation called “Sponsors” that includes the primary keys of the EVENT and SPONSOR entities. The primary key of the EVENT entity, EventID, is added to the “Sponsors” relation. Also the primary key of the SPONSOR entity, SponsorID, is added. The combination of both added keys represents the primary key of the “Sponsors” relation and they are consequently underlined.

1. **ORGANIZES**

|  |  |  |
| --- | --- | --- |
| **OrganizerID** | **EventID** | **Responsibility** |

Every ORGANIZER “Organizes” at least one EVENT. An EVENT can have many ORGANIZERS organizing it, and one ORGANIZER can organize many EVENTS. The “Organizes” relationship links the EVENT entity and the ORGANIZER entity. We create a new relation called “Organizes” that includes the primary keys of the EVENT and ORGANIZER entities. The primary key of the EVENT entity, EventID, is added to the “Organizes” relation. Also the primary key of the ORGANIZER entity, OrganizerID, is added. The combination of both added keys represents the primary key of the “Organizes” relation and they are consequently underlined.

1. **PROVIDES FOR**

|  |  |  |  |
| --- | --- | --- | --- |
| **SupplierID** | **EventID** | **Invoice\_Number** | **Supply\_Status** |

Every SUPPLIER “Provides for” at least one EVENT. An EVENT can have many SUPPLIERS providing for it, and one SUPPLIER can provide for many EVENTS. The “Provides for” relationship links the EVENT entity and the SUPPLIER entity. We create a new relation called “Provides for” that includes the primary keys of the EVENT and SUPPLIER entities. The primary key of the EVENT entity, EventID, is added to the “Provides for” relation. Also the primary key of the SUPPLIER entity, SupplierID, is added. The combination of both added keys represents the primary key of the “Provides for” relation and they are consequently underlined.

1. **HAS**

|  |  |
| --- | --- |
| **CustomerID** | **EventID** |

Every EVENT “Has” at least one CUSTOMER. An EVENT can have many CUSTOMERS, and one CUSTOMER can have many EVENTS. The “Has” relationship links the EVENT entity and the CUSTOMER entity. We create a new relation called “Has” that includes the primary keys of the EVENT and CUSTOMER entities. The primary key of the EVENT entity, EventID, is added to the “Has” relation. Also the primary key of the CUSTOMER entity, CustomerID, is added. The combination of both added keys represents the primary key of the “Has” relation and they are consequently underlined.

1. **PROVIDES**

|  |  |
| --- | --- |
| **SupplierID** | **ServiceID** |

Every SUPPLIER “Provides” at least one SERVICE. A SUPPLIER can provide many SERVICES, and one SERVICE can be provided by many SUPPLIERS. The “Provides” relationship links the SUPPLIER entity and the SERVICE entity. We create a new relation called “Provides” that includes the primary keys of the SERVICE and SUPPLIER entities. The primary key of the SUPPLIER entity, SupplierID, is added to the “Provides” relation. Also the primary key of the Service entity, ServiceID, is added. The combination of both added keys represents the primary key of the “Provides” relation and they are consequently underlined.STEP 6: Mapping of Multivalued Attributes

In this step, we are going to map the multivalued attributes which we ignored before. For each multivalued attribute we create a new relation containing the related attribute and the primary key of the entity to which it belongs. Their combination will represent the primary key of the newly created relation. Our multivalued attributes are:

1. **Event\_Types**

|  |  |
| --- | --- |
| **EventID** | **Type** |

The multivalued attribute Type belongs to the EVENT entity. We represent it by creating a relation “Event\_Types”. Its primary key is composed of EventID, the primary key of the EVENT entity, and the attribute Type which represents the multiple types that an event can have.

1. **Staff\_Role**

|  |  |
| --- | --- |
| **StaffID** | **Role** |

The multivalued attribute Role belongs to the STAFF entity. We represent it by creating a relation “Staff\_Role”. Its primary key is composed of StaffID, the primary key of the STAFF entity, and the attribute Role which represents the multiple roles that a staff can have.

1. **Service\_Decoration**

|  |  |
| --- | --- |
| **ServiceID** | **Decoration** |

The multivalued attribute Decoration belongs to the SERVICE entity. We represent it by creating a relation “Service\_Decoration”. Its primary key is composed of ServiceID, the primary key of the SERVICE entity, and the attribute Decoration which represents the multiple decorations that a service can bring.

1. **Service\_Types**

|  |  |
| --- | --- |
| **ServiceID** | **Type** |

The multivalued attribute Type belongs to the SERVICE entity. We represent it by creating a relation “Service\_Types”. Its primary key is composed of ServiceID, the primary key of the SERVICE entity, and the attribute Type which represents the multiple Types that a service can have.

1. **Attendee\_Preferences**

|  |  |  |
| --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Preference** |

The multivalued attribute Preference belongs to the ATTENDEE entity. We represent it by creating a relation “Attendee\_Preference”. Its primary key is composed of EventID and Seat\_Assignment, the partial key of the ATTENDEE entity, and the attribute Preference which represents the multiple Preferences that an attendee can have.

1. **Offer\_Types**

|  |  |
| --- | --- |
| **OfferID** | **Type** |

The multivalued attribute Type belongs to the SPECIAL\_OFFER entity. We represent it by creating a relation “Offer\_Types”. Its primary key is composed of OfferID, the primary key of the SPECIAL\_OFFER entity, and the attribute Type which represents the multiple Types that an offer can have.

1. **Customer\_Phone\_Number**

|  |  |
| --- | --- |
| **CustomerID** | **Phone\_Number** |

The multivalued attribute Phone\_Number belongs to the CUSTOMER entity. We represent it by creating a relation “Customer\_Phone\_Number”. Its primary key is composed of CustomerID, the primary key of the CUSTOMER entity, and the attribute Phone\_Number which represents the multiple phone numbers that a customer can have.

1. **Supplier\_Phone\_Number**

|  |  |
| --- | --- |
| **SupplierID** | **Phone\_Number** |

The multivalued attribute Phone\_Number belongs to the SUPPLIER entity. We represent it by creating a relation “Supplier\_Phone\_Number”. Its primary key is composed of SupplierID, the primary key of the SUPPLIER entity, and the attribute Phone\_Number which represents the multiple phone numbers that a supplier can have.

1. **Organizer\_Contact\_Information**

|  |  |
| --- | --- |
| **OrganizerID** | **Contact\_Information** |

The multivalued attribute Contact\_Information belongs to the ORGANIZER entity. We represent it by creating a relation “Organizer\_Contact\_Information”. Its primary key is composed of OrganizerID, the primary key of the ORGANIZER entity, and the attribute Contact\_Information which represents the multiple contact info that an Organizer can have.

1. **Organizer\_Role**

|  |  |
| --- | --- |
| **OrganizerID** | **Role** |

The multivalued attribute Role belongs to the ORGANIZER entity. We represent it by creating a relation “Organizer\_Role”. Its primary key is composed of OrganizerID, the primary key of the ORGANIZER entity, and the attribute Role which represents the multiple Roles that an organizer can have.

1. **Venue\_Owner**

|  |  |
| --- | --- |
| **VenueID** | **Owner** |

The multivalued attribute Owner belongs to the VENUE entity. We represent it by creating a relation “Venue\_Owner”. Its primary key is composed of VenueID, the primary key of the VENUE entity, and the attribute Owner which represents the multiple Owners that own a venue.

## **STEP 7: Mapping of N­ary Relationship Types**

In this step, we were supposed to map the N­ary Relationship types. We were supposed to create a new relation containing the primary keys of all participating entities and any simple attributes of the relationship type. However, we didn’t find any N­ary relationship types, so this step is not applicable here.

# Final Step: Final Displays

**EVENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EventID | **Price** | **Start\_Date** | **End\_Date** | **Name** |

**STAFF**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| StaffID | **Phone\_Number** | **Wage** | **First** | **Middle** | **Last** |

**SPONSOR**

|  |  |  |  |
| --- | --- | --- | --- |
| SponsorID | **Name** | **Phone\_Number** | **Sponsorship\_Amount** |

**VENUE**

|  |  |  |  |
| --- | --- | --- | --- |
| VenueID | **Name** | **Phone\_Number** | **Address** |

**SERVICE**

|  |
| --- |
| ServiceID |

**SPECIAL OFFER**

|  |  |
| --- | --- |
| OfferID | **Valid\_Date** |

**CUSTOMER**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CustomerID | **Email** | **First** | **Middle** | **Last** | **Payment\_Types** | **Amount\_Paid** | **Amount\_Due** | **Payment\_Day** |

**SUPPLIER**

|  |  |  |  |
| --- | --- | --- | --- |
| SupplierID | **Name** | **Details** | **Fee** |

**ORGANIZER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrganizerID | **Address** | **First** | **Middle** | **Last** |

**ATTENDEE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Status** | **Total\_Attendee** | **First** | **Middle** | **Last** |

**CHOOSES**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **SO\_ID** | **Email** | **First** | **Middle** | **Last** | **Payment\_Types** | **Amount\_Paid** | **Amount\_Due** | **Payment\_Day** |

**ATTENDEE (ATTENDS):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Status** | **Total\_Attendee** | **First** | **Middle** | **Last** |

**STAFF (MANAGES):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StaffID | **OrganizerID** | **Phone\_Number** | **Wage** | **First** | **Middle** | **Last** |

**HOLDS**

|  |  |
| --- | --- |
| **EventID** | **VenueID** |

**USED IN**

|  |  |
| --- | --- |
| **EventID** | **ServiceID** |

**WORKS ON**

|  |  |
| --- | --- |
| **StaffID** | **EventID** |

**SPONSORS**

|  |  |
| --- | --- |
| **SponsorID** | **EventID** |

**ORGANIZES**

|  |  |
| --- | --- |
| **OrganizerID** | **EventID** |

**PROVIDES FOR**

|  |  |
| --- | --- |
| **SupplierID** | **EventID** |

**HAS**

|  |  |
| --- | --- |
| **CustomerID** | **EventID** |

**PROVIDES**

|  |  |
| --- | --- |
| **SupplierID** | **ServiceID** |

**Event\_Types**

|  |  |
| --- | --- |
| **EventID** | **Type** |

**Staff\_Role**

|  |  |
| --- | --- |
| **StaffID** | **Role** |

**Service\_Decoration**

|  |  |
| --- | --- |
| **ServiceID** | **Decoration** |

**Service\_Types**

|  |  |
| --- | --- |
| **ServiceID** | **Type** |

**Attendee\_Preferences**

|  |  |  |
| --- | --- | --- |
| **EventID** | **Attendee\_Number** | **Preference** |

**Offer\_Types**

|  |  |
| --- | --- |
| **OfferID** | **Type** |

**Customer\_Phone\_Number**

|  |  |
| --- | --- |
| **CustomerID** | **Phone\_Number** |

**Supplier\_Phone\_Number**

|  |  |
| --- | --- |
| **SupplierID** | **Phone\_Number** |

**Organizer\_Contact\_Information**

|  |  |
| --- | --- |
| **OrganizerID** | **Contact\_Information** |

**Organizer\_Role**

|  |  |
| --- | --- |
| **OrganizerID** | **Role** |

**Venue\_Owner**

|  |  |
| --- | --- |
| **VenueID** | **Owner** |

# Creation of Database

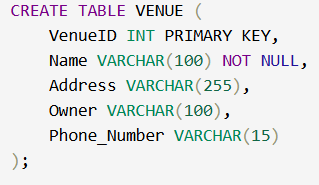
After designing the ER diagram for Cedar Elegance Events and mapping this diagram into relational database design, now it is time to start creating the actual tables for our database on the Oracle Database Server. We will start by creating all the tables and then inserting data into these tables. Finally, we will execute some queries to display the importance of the database and especially in a hospital.

1. EVENT:

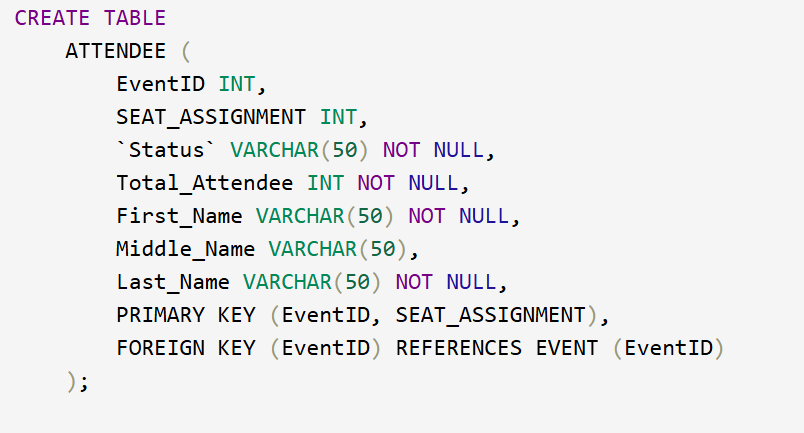
A screen shot of a computer code

AI-generated content may be incorrect.

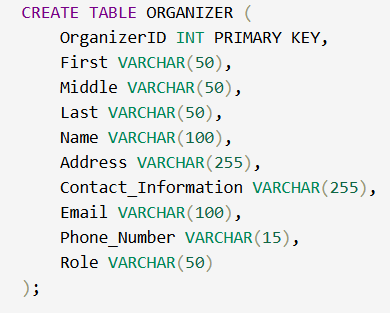
1. VENUE:



1. ATTENDEE:



1. ORGANIZER:



1. STAFF:

A screenshot of a computer

AI-generated content may be incorrect.

1. CUSTOMER:

A screenshot of a computer code

AI-generated content may be incorrect.

1. SPECIAL\_OFFER

A close-up of a screen

AI-generated content may be incorrect.

1. SPONSOR

A screen shot of a computer code

AI-generated content may be incorrect.

1. SERVICE

A close-up of a computer screen

AI-generated content may be incorrect.

1. SUPPLIER

A screenshot of a computer

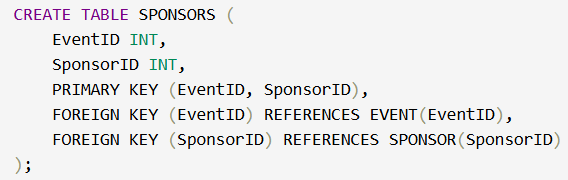
AI-generated content may be incorrect.

1. WORKS\_ON

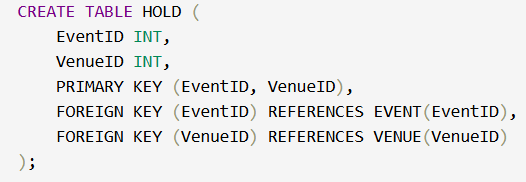
A screenshot of a computer program

AI-generated content may be incorrect.

1. SPONSORS



1. HOLD



1. ORGANIZES

A screenshot of a computer code

AI-generated content may be incorrect.

1. HAS

A computer screen shot of a computer code

AI-generated content may be incorrect.

1. USED\_IN

A computer screen shot of a computer code

AI-generated content may be incorrect.

1. ATTENDS

A computer screen shot of black text

AI-generated content may be incorrect.

1. CHOOSES

A close-up of a computer code

AI-generated content may be incorrect.

1. PROVIDES\_FOR

A screen shot of a computer code

AI-generated content may be incorrect.

1. PROVIDES

A computer screen shot of a computer code

AI-generated content may be incorrect.

1. MANAGES

A computer code with black text

AI-generated content may be incorrect.

# TABLE DESCRIPTION

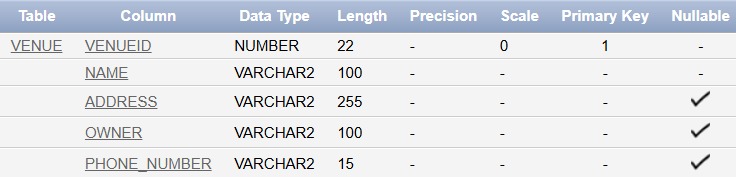
After creating all the tables on the oracle database server, we can view the description of each table in order to make sure everything is fine and no mistakes were made during creation of table.

1. EVENT

A screenshot of a computer

AI-generated content may be incorrect.

1. VENUE



1. ATTENDEE

A screenshot of a computer

AI-generated content may be incorrect.

1. ORGANIZER

A screenshot of a computer

AI-generated content may be incorrect.

1. STAFF

A screenshot of a data table

AI-generated content may be incorrect.

1. CUSTOMER

A screenshot of a data table

AI-generated content may be incorrect.

1. SPECIAL\_OFFER

A screenshot of a phone

AI-generated content may be incorrect.

1. SPONSOR

A screenshot of a computer

AI-generated content may be incorrect.

1. SERVICE

A screenshot of a phone

AI-generated content may be incorrect.

1. SUPPLIER

A screenshot of a computer

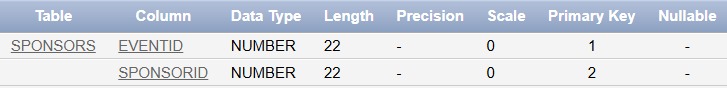
AI-generated content may be incorrect.

1. WORKS\_ON

A screenshot of a computer

AI-generated content may be incorrect.

1. SPONSORS



1. HOLD

A screenshot of a phone

AI-generated content may be incorrect.

1. ORGANIZES

A screenshot of a computer

AI-generated content may be incorrect.

1. HAS

A screenshot of a facebook page

AI-generated content may be incorrect.

1. USED\_IN

A screenshot of a facebook page

AI-generated content may be incorrect.

1. ATTENDS

A screenshot of a phone

AI-generated content may be incorrect.

1. CHOOSES

A screenshot of a phone

AI-generated content may be incorrect.

1. PROVIDES\_FOR

A screenshot of a phone

AI-generated content may be incorrect.

1. PROVIDES

A screenshot of a phone

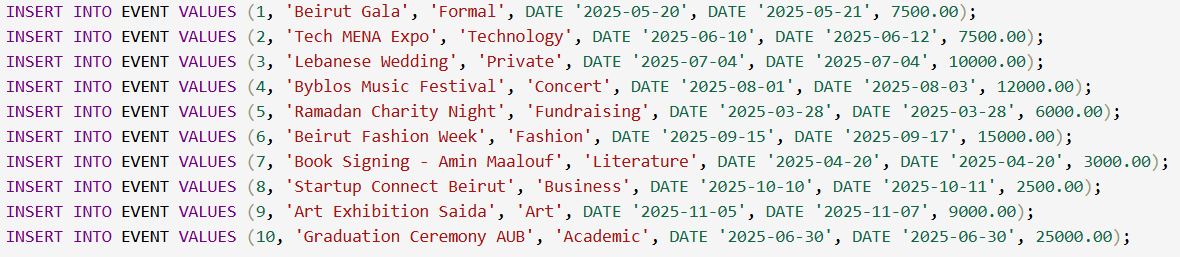
AI-generated content may be incorrect.

1. MANAGES

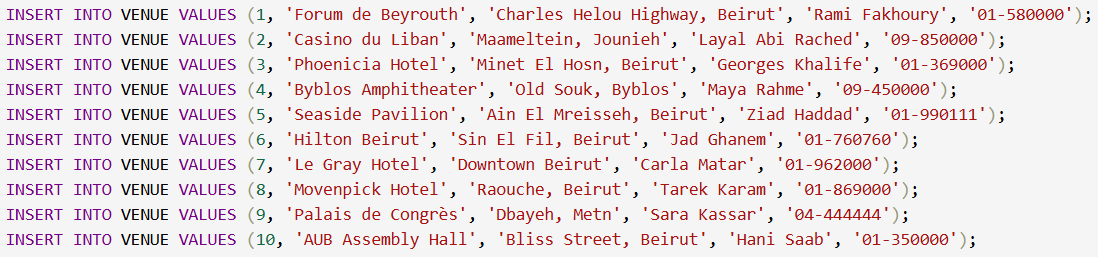


# Insertion into Tables

1. EVENT



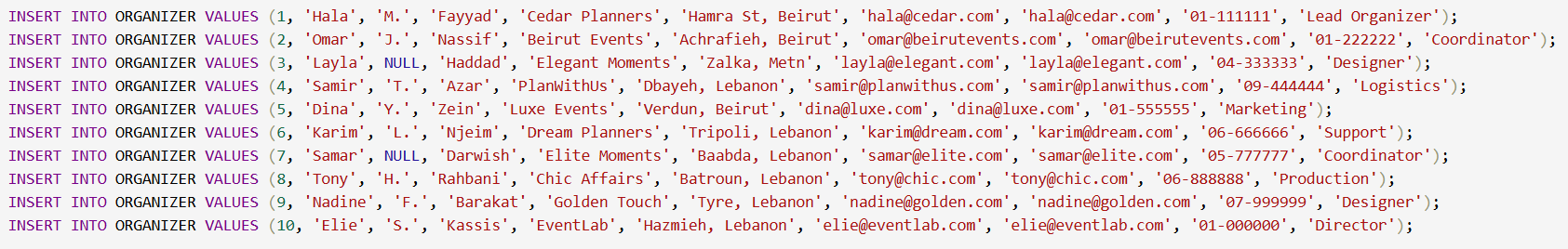
1. VENUE



1. ATTENDEE



1. ORGANIZER



1. STAFF

A screenshot of a computer screen

AI-generated content may be incorrect.

1. CUSTOMER

A close up of text

AI-generated content may be incorrect.

1. SPECIAL\_OFFER

A screenshot of a computer

AI-generated content may be incorrect.

1. SPONSOR

A screenshot of a computer screen

AI-generated content may be incorrect.

1. SERVICE

A screenshot of a computer

AI-generated content may be incorrect.

1. SUPPLIER

A close-up of a number

AI-generated content may be incorrect.

1. WORKS\_ON

A screenshot of a computer code

AI-generated content may be incorrect.

1. SPONSORS

A screenshot of a computer

AI-generated content may be incorrect.

1. HOLD

A screenshot of a computer

AI-generated content may be incorrect.

1. ORGANIZES

A screenshot of a computer screen

AI-generated content may be incorrect.

1. HAS

A screenshot of a computer

AI-generated content may be incorrect.

1. USED\_IN

A screenshot of a computer code

AI-generated content may be incorrect.

1. ATTENDS

A screenshot of a computer

AI-generated content may be incorrect.

1. CHOOSES

A screenshot of a computer

AI-generated content may be incorrect.

1. PROVIDES\_FOR

A screenshot of a computer code

AI-generated content may be incorrect.

1. PROVIDES

A screenshot of a computer

AI-generated content may be incorrect.

1. MANAGES

A screenshot of a computer

AI-generated content may be incorrect.

# Final Tables States

1. EVENT

A screenshot of a computer

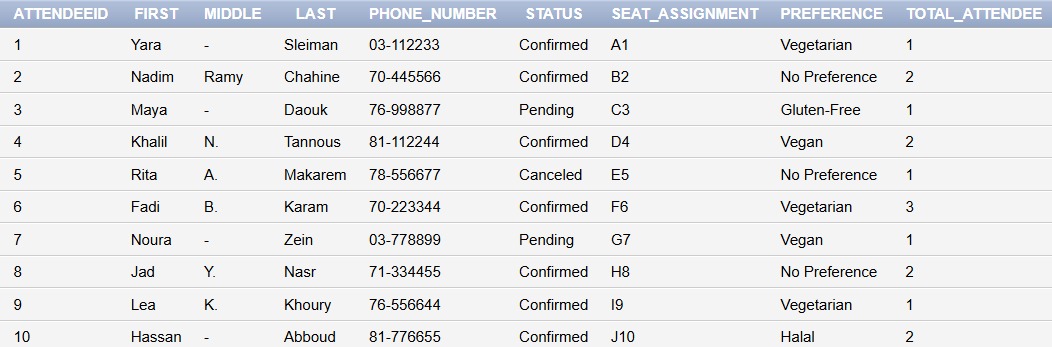
AI-generated content may be incorrect.

1. VENUE

A screenshot of a computer

AI-generated content may be incorrect.

1. ATTENDEE



1. ORGANIZER

A screenshot of a computer

AI-generated content may be incorrect.

1. STAFF

A screenshot of a computer

AI-generated content may be incorrect.

1. CUSTOMER

A screenshot of a computer

AI-generated content may be incorrect.

1. SPECIAL\_OFFER

A screenshot of a list of events

AI-generated content may be incorrect.

1. SPONSOR

A table with numbers and numbers

AI-generated content may be incorrect.

1. SERVICE

A table with text on it

AI-generated content may be incorrect.

1. SUPPLIER

A screenshot of a phone number

AI-generated content may be incorrect.

1. WORKS\_ON

A screenshot of a cell phone

AI-generated content may be incorrect.

1. SPONSORS

A screenshot of a cell phone

AI-generated content may be incorrect.

1. HOLD

A screenshot of a cell phone

AI-generated content may be incorrect.

1. ORGANIZES

A screenshot of a computer

AI-generated content may be incorrect.

1. HAS

A screenshot of a cell phone

AI-generated content may be incorrect.

1. USED\_IN

A white table with black numbers

AI-generated content may be incorrect.

1. ATTENDS

A white rectangular table with black numbers

AI-generated content may be incorrect.

1. CHOOSES

A screenshot of a cell phone

AI-generated content may be incorrect.

1. PROVIDES\_FOR



1. PROVIDES

A screenshot of a cell phone

AI-generated content may be incorrect.

1. MANAGES

A white rectangular object with black numbers

AI-generated content may be incorrect.

**Description** 1: Get most details about an event, like event IDs with their names, their venue names, sponsor names, and the total number of attendees for each event.

**QUERY 1:**SELECT

E.EventID,

E.Name AS EventName,

V.Name AS VenueName,

S.Name AS SponsorName,

COUNT(A.AttendeeID) AS TotalAttendees

FROM

EVENT E

JOIN

HOLD H ON E.EventID = H.EventID

JOIN

VENUE V ON H.VenueID = V.VenueID

JOIN

SPONSORS SP ON E.EventID = SP.EventID

JOIN

SPONSOR S ON SP.SponsorID = S.SponsorID

JOIN

ATTENDS A ON E.EventID = A.EventID

GROUP BY

E.EventID, E.Name, V.Name, S.Name;

A screenshot of a computer

Description automatically generated

**Description 2:** A customer wanted to plan an event outside of Beirut so we extract all venues that are not located in Beirut:

**QUERY 2:**

SELECT NAME, ADDRESS, OWNER, PHONE\_NUMBER

FROM VENUE

WHERE address NOT LIKE '%Beirut%';

A screenshot of a phone

Description automatically generated

Description 3: It’s the end of the month and your financial staff needs a spreadsheet of customers that owe you money with their contact information

**QUERY 3:**

SELECT

name AS customer\_name,

phone\_number,

email,

amount\_due

FROM CUSTOMER

WHERE amount\_due> 0;

A screenshot of a phone number

Description automatically generated

Description 4: I am feeling generous today and my event planning business is doing well, so I decided to give a little bonus to some of my staff member ( mainly the one that make under 300$ ) so I want to write a query that extracts staff member that make less than 300$ with their contact info

**Queries 4:**

SELECT

name AS staff\_name,

phone\_number,

role,

wage

FROM STAFF

WHERE wage < 300;

A screenshot of a number

Description automatically generated

Description 5: You want a spreadsheet of attendees with pending status with their contact information and the event they’re going to attend.

**QUERY 5:**

SELECT

A.attendeeid,

A.first AS attendee\_first,

A.phone\_number,

E.name AS event\_name

FROM ATTENDEE A

JOIN ATTENDS B ON A.attendeeid = B.attendeeid

JOIN EVENT E ON B.eventid = E.eventid

WHERE LOWER(A.status) = 'pending';



Description 6: your financial team needs to calculate costs for events sponsored for less than 3000

**QUERY 6:**

SELECT

s.SponsorID,

s.Name AS Sponsor\_Name,

s.sponsorship\_Amount,

e.name,

e.price

FROM SPONSOR s

JOIN SPONSORS sp ON s.SponsorID = sp.SponsorID

JOIN EVENT e ON sp.EventID = e.EventID

WHERE s.sponsorship\_Amount < 3000;



Description 7: Mira Chamoun one of your customers has responded to your financial team and initiated her due payments

**QUERY 7:**

UPDATE CUSTOMER

SET amount\_due = 0,

amount\_paid = budget

WHERE first = 'Mira'

AND last = 'Chamoun';

SELECT \*

FROM CUSTOMER

WHERE first = ‘Mira’

AND last = ‘Chamoun’;



Description 8: You decided to give a 15% pay raise to your chef

**QUERY 8:**

UPDATE STAFF

SET Wage = Wge\* 1.15

WHERE Role = 'Chef';



Description 9: you want to assess your companies work so u write a query to check organized events in the next month

**QUERY 9:**

SELECT

E.Name AS Event\_Name,

E.Start\_Date,

O.First || ' ' || O.Last AS Organizer\_Name,

O.Email

FROM EVENT E

JOIN ORGANIZER O ON E.eventID = O.OrganizerID

WHERE E.Start\_Date BETWEEN SYSDATE AND SYSDATE + 30;



Description 10: Calculating your average staff wage

**QUERY 10:**

SELECT

ROUND(AVG(Wage), 2) AS Average\_Staff\_Wage

FROM

    STAFF;

A close up of a sign

Description automatically generated

Full Code:

CREATE DATABASE CedarEleganceEvents;

USE CedarEleganceEvents;

-- Table for EVENT

CREATE TABLE

`EVENT` (

EventID INT PRIMARY KEY,

`Name` VARCHAR(255) NOT NULL,

`Start\_Date` DATETIME NOT NULL,

End\_Date DATETIME NOT NULL,

Price DECIMAL(10, 2) NOT NULL

);

--Insert data into EVENT table

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (1, 'Enchanted Wedding', '2025-05-15 14:00:00', '2025-05-15 20:00:00', 5500.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (2, 'Innovators Summit', '2025-06-01 09:00:00', '2025-06-01 17:00:00', 3500.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (3, 'Magical Birthday Bash', '2025-07-10 18:00:00', '2025-07-10 23:00:00', 1800.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (4, 'Rock the Night Concert', '2025-08-20 19:00:00', '2025-08-20 23:59:00', 8500.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (5, 'Masterpiece Art Gala', '2025-09-05 10:00:00', '2025-09-05 18:00:00', 2500.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (6, 'Golden Hearts Charity Ball', '2025-10-15 19:00:00', '2025-10-15 23:00:00', 12000.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (7, 'Future Tech Expo', '2025-11-01 08:00:00', '2025-11-01 18:00:00', 7500.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (8, 'Runway Dreams Fashion Show', '2025-12-10 18:00:00', '2025-12-10 22:00:00', 6500.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (9, 'Literary Stars Book Launch', '2025-12-20 15:00:00', '2025-12-20 18:00:00', 3000.00);

INSERT INTO `EVENT` (EventID, `Name`, `Start\_Date`, End\_Date, Price)

VALUES (10, 'Glamorous New Year Gala', '2025-12-31 20:00:00', '2026-01-01 02:00:00', 13000.00);

-- Table for VENUE

CREATE TABLE

VENUE (

VenueID INT PRIMARY KEY,

`Name` VARCHAR(255) NOT NULL,

`Address` VARCHAR(255) NOT NULL,

Phone\_Number VARCHAR(15)

);

--Insert data into VENUE table

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (1, 'Crystal Palace', 'Beirut Central District, Beirut, Lebanon', '+961-1-123456');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (2, 'Grand Horizon Hall', 'Al-Mina, Tripoli, Lebanon', '+961-6-654321');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (3, 'The Enchanted Garden', 'Byblos Old Souk, Byblos, Lebanon', '+961-9-555123');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (4, 'Skyline Rooftop', 'Corniche Sidon, Sidon, Lebanon', '+961-7-444987');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (5, 'Golden Pavilion', 'Boulevard Zahle, Zahle, Lebanon', '+961-8-333222');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (6, 'Moonlit Terrace', 'Tyre Coastal Area, Tyre, Lebanon', '+961-7-666555');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (7, 'Emerald Ballroom', 'Baalbek Ruins Area, Baalbek, Lebanon', '+961-8-777888');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (8, 'The Velvet Lounge', 'Jounieh Bay Area, Jounieh, Lebanon', '+961-9-222333');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (9, 'Aurora Banquet Hall', 'Batroun Coastal Area, Batroun, Lebanon', '+961-6-888777');

INSERT INTO VENUE (VenueID, `Name`, `Address`, Phone\_Number)

VALUES (10, 'Starlight Amphitheater', 'Aley Mountain Area, Aley, Lebanon', '+961-5-999000');

-- Table for ATTENDEE (Weak Entity)

CREATE TABLE

ATTENDEE (

EventID INT,

SEAT\_ASSIGNMENT INT,

`Status` VARCHAR(50) NOT NULL,

Total\_Attendee INT NOT NULL,

First\_Name VARCHAR(50) NOT NULL,

Middle\_Name VARCHAR(50),

Last\_Name VARCHAR(50) NOT NULL,

PRIMARY KEY (EventID, SEAT\_ASSIGNMENT),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID)

);

--Insert data into ATTENDEE table

-- EventID 1

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (1, 1, 'Confirmed', 2, 'Clark', 'Joseph', 'Kent');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (1, 2, 'Confirmed', 2, 'Ramzi', NULL, 'Harati');

-- EventID 2

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (2, 1, 'Confirmed', 3, 'Bruce', NULL, 'Wayne');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (2, 2, 'Pending', 3, 'Diana', 'Themyscira', 'Prince');

-- EventID 3

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (3, 1, 'Confirmed', 2, 'Barry', NULL, 'Allen');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (3, 2, 'Pending', 2, 'Iris', NULL, 'West');

-- EventID 4

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (4, 1, 'Confirmed', 3, 'Arthur', NULL, 'Curry');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (4, 2, 'Confirmed', 3, 'Mera', NULL, 'Queen');

-- EventID 5

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (5, 1, 'Confirmed', 2, 'Victor', NULL, 'Stone');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (5, 2, 'Pending', 2, 'Silas', NULL, 'Stone');

-- EventID 6

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (6, 1, 'Confirmed', 2, 'Hal', NULL, 'Jordan');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (6, 2, 'Pending', 2, 'Carol', NULL, 'Ferris');

-- EventID 7

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (7, 1, 'Confirmed', 2, 'Oliver', NULL, 'Queen');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (7, 2, 'Pending', 2, 'Felicity', NULL, 'Smoak');

-- EventID 8

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (8, 1, 'Confirmed', 2, 'John', NULL, 'Constantine');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (8, 2, 'Pending', 2, 'Zatanna', NULL, 'Zatara');

-- EventID 9

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (9, 1, 'Confirmed', 2, 'Billy', 'William', 'Batson');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (9, 2, 'Pending', 2, 'Freddy', 'Eugene', 'Freeman');

-- EventID 10

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (10, 1, 'Confirmed', 2, 'Lex', NULL, 'Luthor');

INSERT INTO ATTENDEE (EventID, SEAT\_ASSIGNMENT, `Status`, Total\_Attendee, First\_Name, Middle\_Name, Last\_Name)

VALUES (10, 2, 'Pending', 2, 'Mercy', NULL, 'Graves');

-- Table for ORGANIZER

CREATE TABLE

ORGANIZER (

OrganizerID INT PRIMARY KEY,

First\_Name VARCHAR(50) NOT NULL,

Middle\_Name VARCHAR(50),

Last\_Name VARCHAR(50) NOT NULL,

`Address` VARCHAR(255)

);

--Insert data into ORGANIZER table

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (1, 'Ramzi', NULL, 'Haraty', 'Beirut, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (2, 'Imad', NULL, 'Mahmoud', 'Tripoli, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (3, 'Ali', NULL, 'Rida', 'Byblos, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (4, 'Reina', NULL, 'Harake', 'Sidon, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (5, 'Ali', NULL, 'El Hajj', 'Zahle, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (6, 'Maher', NULL, 'Saadi', 'Tyre, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (7, 'Cesar', NULL, 'Al Ayache', 'Baalbek, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (8, 'Jean', 'Claude', 'Cherfane', 'Jounieh, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (9, 'Alaa', 'Al Dine', 'Doumit', 'Batroun, Lebanon');

INSERT INTO ORGANIZER (OrganizerID, First\_Name, Middle\_Name, Last\_Name, `Address`)

VALUES (10, 'Sara', 'Lea', 'Fares', 'Aley, Lebanon');

-- Table for STAFF

CREATE TABLE

STAFF (

StaffID INT PRIMARY KEY,

First\_Name VARCHAR(50) NOT NULL,

Middle\_Name VARCHAR(50),

Last\_Name VARCHAR(50) NOT NULL,

Phone\_Number VARCHAR(15),

Wage DECIMAL(10, 2) NOT NULL,

OrganizerID INT,

FOREIGN KEY (OrganizerID) REFERENCES ORGANIZER (OrganizerID)

);

--Insert data into STAFF table

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (1, 'Ahmad', 'Fouad', 'Hassan', '+961-1-123456', 1200.00, 1);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (2, 'Layla', 'Marie', 'Khalil', '+961-6-654321', 1500.00, 2);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (3, 'Jad', 'Elie', 'Nasr', '+961-9-555123', 1100.00, 3);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (4, 'Rana', 'Joseph', 'Saad', '+961-7-444987', 1300.00, 4);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (5, 'Fouad', 'Michel', 'Ghanem', '+961-8-333222', 1400.00, 5);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (6, 'Maya', 'Rita', 'Haddad', '+961-7-666555', 1250.00, 6);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (7, 'Karim', 'Nour', 'Doumit', '+961-8-777888', 1350.00, 7);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (8, 'Rita', 'Sami', 'Fares', '+961-9-222333', 1450.00, 8);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (9, 'Elie', 'George', 'Cherfane', '+961-6-888777', 1500.00, 9);

INSERT INTO STAFF (StaffID, First\_Name, Middle\_Name, Last\_Name, Phone\_Number, Wage, OrganizerID)

VALUES (10, 'Sara', 'Lea', 'Harb', '+961-5-999000', 1600.00, 10);

-- Table for CUSTOMER

CREATE TABLE

CUSTOMER (

CustomerID INT PRIMARY KEY,

First\_Name VARCHAR(50) NOT NULL,

Middle\_Name VARCHAR(50),

Last\_Name VARCHAR(50) NOT NULL,

Email VARCHAR(255) NOT NULL,

Payment\_Types VARCHAR(50),

Amount\_Paid DECIMAL(10, 2),

Amount\_Due DECIMAL(10, 2),

Payment\_Day DATE,

SO\_ID INT,

FOREIGN KEY (SO\_ID) REFERENCES SPECIAL\_OFFER (OfferID)

);

--Insert data into CUSTOMER table

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (1, 'Rami', 'Joseph', 'Haddad', 'rami.haddad@example.com', 'Credit Card', 500.00, 0.00, '2025-04-01', NULL);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (2, 'Lina', 'Marie', 'Karam', 'lina.karam@example.com', 'Cash', 300.00, 200.00, '2025-04-05', 1);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (3, 'Fadi', 'George', 'Saliba', 'fadi.saliba@example.com', 'Bank Transfer', 1000.00, 0.00, '2025-04-10', 2);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (4, 'Nadine', 'Elie', 'Chahine', 'nadine.chahine@example.com', 'Credit Card', 700.00, 300.00, '2025-04-15', 3);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (5, 'Jad', 'Michel', 'Abou Khalil', 'jad.khalil@example.com', 'Cash', 250.00, 50.00, '2025-04-20', NULL);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (6, 'Maya', 'Rita', 'Nasr', 'maya.nasr@example.com', 'Bank Transfer', 1200.00, 0.00, '2025-04-25', 4);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (7, 'Karim', 'Fouad', 'Ghanem', 'karim.ghanem@example.com', 'Credit Card', 800.00, 200.00, '2025-04-28', 5);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (8, 'Rita', 'Nour', 'Saad', 'rita.saad@example.com', 'Cash', 400.00, 100.00, '2025-04-30', NULL);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (9, 'Elie', 'Sami', 'Doumit', 'elie.doumit@example.com', 'Bank Transfer', 1500.00, 0.00, '2025-05-01', 6);

INSERT INTO CUSTOMER (CustomerID, First\_Name, Middle\_Name, Last\_Name, Email, Payment\_Types, Amount\_Paid, Amount\_Due, Payment\_Day, SO\_ID)

VALUES (10, 'Sara', 'Lea', 'Fares', 'sara.fares@example.com', 'Credit Card', 1000.00, 500.00, '2025-05-05', 7);

-- Table for SPECIAL OFFER

CREATE TABLE

SPECIAL\_OFFER (

OfferID INT PRIMARY KEY,

Valid\_Date DATE NOT NULL

);

--Insert data into SPECIAL\_OFFER table

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (1, '2025-05-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (2, '2025-06-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (3, '2025-07-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (4, '2025-08-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (5, '2025-09-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (6, '2025-10-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (7, '2025-11-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (8, '2025-12-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (9, '2026-01-01');

INSERT INTO SPECIAL\_OFFER (OfferID, Valid\_Date)

VALUES (10, '2026-02-01');

-- Table for SPONSOR

CREATE TABLE

SPONSOR (

SponsorID INT PRIMARY KEY,

`Name` VARCHAR(255) NOT NULL,

Phone\_Number VARCHAR(15),

Sponsorship\_Amount DECIMAL(10, 2) NOT NULL

);

--Insert data into SPONSOR table

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (1, 'Lebanon Bank', '+961-1-123456', 10000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (2, 'Cedars Insurance', '+961-6-654321', 8000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (3, 'Phoenicia Hotels', '+961-9-555123', 12000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (4, 'Beirut Traders', '+961-7-444987', 7000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (5, 'Golden Olive Oil', '+961-8-333222', 5000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (6, 'Tyre Tourism Board', '+961-7-666555', 6000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (7, 'Baalbek Cultural Association', '+961-8-777888', 9000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (8, 'Jounieh Marina', '+961-9-222333', 11000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (9, 'Batroun Brewery', '+961-6-888777', 4000.00);

INSERT INTO SPONSOR (SponsorID, `Name`, Phone\_Number, Sponsorship\_Amount)

VALUES (10, 'Aley Mountain Resorts', '+961-5-999000', 15000.00);

-- Table for SERVICE

CREATE TABLE

`SERVICE` (

ServiceID INT PRIMARY KEY

);

--Insert data into SERVICE table

INSERT INTO `SERVICE` (ServiceID)

VALUES (1);

INSERT INTO `SERVICE` (ServiceID)

VALUES (2);

INSERT INTO `SERVICE` (ServiceID)

VALUES (3);

INSERT INTO `SERVICE` (ServiceID)

VALUES (4);

INSERT INTO `SERVICE` (ServiceID)

VALUES (5);

INSERT INTO `SERVICE` (ServiceID)

VALUES (6);

INSERT INTO `SERVICE` (ServiceID)

VALUES (7);

INSERT INTO `SERVICE` (ServiceID)

VALUES (8);

INSERT INTO `SERVICE` (ServiceID)

VALUES (9);

INSERT INTO `SERVICE` (ServiceID)

VALUES (10);

-- Table for SUPPLIER

CREATE TABLE

SUPPLIER (

SupplierID INT PRIMARY KEY,

`Name` VARCHAR(255) NOT NULL,

Fee VARCHAR(15),

Details TEXT

);

--Insert data into SUPPLIER table

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (1, 'Lebanon Catering Co.', '5000', 'Provides catering services for events.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (2, 'Elite Sound Systems', '3000', 'Offers sound and lighting equipment.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (3, 'Floral Creations', '2000', 'Specializes in floral decorations.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (4, 'Luxury Rentals', '4000', 'Provides luxury furniture and decor.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (5, 'Event Security Services', '2500', 'Offers professional security personnel.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (6, 'Tech Solutions', '3500', 'Provides audiovisual and technical support.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (7, 'Gourmet Desserts', '1500', 'Specializes in custom desserts and cakes.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (8, 'Photo Memories', '3000', 'Offers professional photography and videography.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (9, 'Sparkling Clean', '1000', 'Provides cleaning services before and after events.');

INSERT INTO SUPPLIER (SupplierID, `Name`, Fee, Details)

VALUES (10, 'Transport Experts', '5000', 'Offers transportation and logistics services.');

-- Table for HOLD (M:N Relationship between EVENT and VENUE)

CREATE TABLE

HOLD (

EventID INT,

VenueID INT,

PRIMARY KEY (EventID, VenueID),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID),

FOREIGN KEY (VenueID) REFERENCES VENUE (VenueID)

);

-- Insert data into HOLD table

INSERT INTO HOLD (EventID, VenueID)

VALUES (1, 1);

INSERT INTO HOLD (EventID, VenueID)

VALUES (2, 2);

INSERT INTO HOLD (EventID, VenueID)

VALUES (3, 3);

INSERT INTO HOLD (EventID, VenueID)

VALUES (4, 4);

INSERT INTO HOLD (EventID, VenueID)

VALUES (5, 5);

INSERT INTO HOLD (EventID, VenueID)

VALUES (6, 6);

INSERT INTO HOLD (EventID, VenueID)

VALUES (7, 7);

INSERT INTO HOLD (EventID, VenueID)

VALUES (8, 8);

INSERT INTO HOLD (EventID, VenueID)

VALUES (9, 9);

INSERT INTO HOLD (EventID, VenueID)

VALUES (10, 10);

-- Table for USED\_IN (M:N Relationship between EVENT and SERVICE)

CREATE TABLE

USED\_IN (

EventID INT,

ServiceID INT,

PRIMARY KEY (EventID, ServiceID),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID),

FOREIGN KEY (ServiceID) REFERENCES SERVICE (ServiceID)

);

-- Insert data into USED\_IN table

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (1, 1);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (1, 2);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (2, 3);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (2, 4);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (3, 5);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (3, 6);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (4, 7);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (4, 8);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (5, 9);

INSERT INTO USED\_IN (EventID, ServiceID)

VALUES (5, 10);

-- Table for WORKS\_ON (M:N Relationship between STAFF and EVENT)

CREATE TABLE

WORKS\_ON (

StaffID INT,

EventID INT,

PRIMARY KEY (StaffID, EventID),

FOREIGN KEY (StaffID) REFERENCES STAFF (StaffID),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID)

);

-- Insert data into WORKS\_ON table

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (1, 1);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (2, 1);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (3, 2);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (4, 2);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (5, 3);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (6, 3);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (7, 4);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (8, 4);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (9, 5);

INSERT INTO WORKS\_ON (StaffID, EventID)

VALUES (10, 5);

-- Table for SPONSORS (M:N Relationship between SPONSOR and EVENT)

CREATE TABLE

SPONSORS (

SponsorID INT,

EventID INT,

PRIMARY KEY (SponsorID, EventID),

FOREIGN KEY (SponsorID) REFERENCES SPONSOR (SponsorID),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID)

);

-- Insert data into SPONSORS table

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (1, 1);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (2, 2);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (3, 3);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (4, 4);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (5, 5);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (6, 6);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (7, 7);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (8, 8);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (9, 9);

INSERT INTO SPONSORS (SponsorID, EventID)

VALUES (10, 10);

-- Table for ORGANIZES (M:N Relationship between ORGANIZER and EVENT)

CREATE TABLE

ORGANIZES (

OrganizerID INT,

EventID INT,

Resposibility VARCHAR(250),

PRIMARY KEY (OrganizerID, EventID),

FOREIGN KEY (OrganizerID) REFERENCES ORGANIZER (OrganizerID),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID)

);

-- Insert data into ORGANIZES table

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (1, 1, 'Coordinate catering and decorations');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (2, 2, 'Manage guest list and invitations');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (3, 3, 'Oversee event setup and logistics');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (4, 4, 'Handle entertainment and performances');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (5, 5, 'Supervise art displays and auctions');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (6, 6, 'Plan charity activities and fundraising');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (7, 7, 'Coordinate technology and presentations');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (8, 8, 'Organize fashion show logistics');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (9, 9, 'Manage book launch and author interactions');

INSERT INTO ORGANIZES (OrganizerID, EventID, Resposibility)

VALUES (10, 10, 'Plan New Year celebrations and fireworks');

-- Table for PROVIDES\_FOR (M:N Relationship between SUPPLIER and EVENT)

CREATE TABLE

PROVIDES\_FOR (

SupplierID INT,

EventID INT,

Invoice\_Number VARCHAR(50),

Supply\_Status VARCHAR(50),

PRIMARY KEY (SupplierID, EventID),

FOREIGN KEY (SupplierID) REFERENCES SUPPLIER (SupplierID),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID)

);

-- Insert data into PROVIDES\_FOR table

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (1, 1, 'INV-1001', 'Delivered');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (2, 2, 'INV-1002', 'Pending');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (3, 3, 'INV-1003', 'Delivered');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (4, 4, 'INV-1004', 'In Progress');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (5, 5, 'INV-1005', 'Delivered');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (6, 6, 'INV-1006', 'Pending');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (7, 7, 'INV-1007', 'Delivered');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (8, 8, 'INV-1008', 'In Progress');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (9, 9, 'INV-1009', 'Delivered');

INSERT INTO PROVIDES\_FOR (SupplierID, EventID, Invoice\_Number, Supply\_Status)

VALUES (10, 10, 'INV-1010', 'Pending');

-- Table for HAS (M:N Relationship between CUSTOMER and EVENT)

CREATE TABLE

HAS (

CustomerID INT,

EventID INT,

PRIMARY KEY (CustomerID, EventID),

FOREIGN KEY (CustomerID) REFERENCES CUSTOMER (CustomerID),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID)

);

-- Insert data into HAS table

INSERT INTO HAS (CustomerID, EventID)

VALUES (1, 1);

INSERT INTO HAS (CustomerID, EventID)

VALUES (2, 2);

INSERT INTO HAS (CustomerID, EventID)

VALUES (3, 3);

INSERT INTO HAS (CustomerID, EventID)

VALUES (4, 4);

INSERT INTO HAS (CustomerID, EventID)

VALUES (5, 5);

INSERT INTO HAS (CustomerID, EventID)

VALUES (6, 6);

INSERT INTO HAS (CustomerID, EventID)

VALUES (7, 7);

INSERT INTO HAS (CustomerID, EventID)

VALUES (8, 8);

INSERT INTO HAS (CustomerID, EventID)

VALUES (9, 9);

INSERT INTO HAS (CustomerID, EventID)

VALUES (10, 10);

-- Table for PROVIDES (M:N Relationship between SUPPLIER and SERVICE)

CREATE TABLE

PROVIDES (

SupplierID INT,

ServiceID INT,

PRIMARY KEY (SupplierID, ServiceID),

FOREIGN KEY (SupplierID) REFERENCES SUPPLIER (SupplierID),

FOREIGN KEY (ServiceID) REFERENCES SERVICE (ServiceID)

);

-- Insert data into PROVIDES table

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (1, 1);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (2, 2);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (3, 3);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (4, 4);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (5, 5);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (6, 6);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (7, 7);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (8, 8);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (9, 9);

INSERT INTO PROVIDES (SupplierID, ServiceID)

VALUES (10, 10);

-- Table for Event\_Types (Multivalued Attribute for EVENT)

CREATE TABLE

Event\_Types (

EventID INT,

`Type` VARCHAR(50),

PRIMARY KEY (EventID, Type),

FOREIGN KEY (EventID) REFERENCES EVENT (EventID)

);

-- Insert data into Event\_Types table

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (1, 'Wedding');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (2, 'Conference');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (3, 'Birthday');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (4, 'Concert');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (5, 'Art Gala');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (6, 'Charity');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (7, 'Expo');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (8, 'Fashion Show');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (9, 'Book Launch');

INSERT INTO Event\_Types (EventID, `Type`)

VALUES (10, 'New Year Celebration');

-- Table for Staff\_Role (Multivalued Attribute for STAFF)

CREATE TABLE

Staff\_Role (

StaffID INT,

`Role` VARCHAR(50),

PRIMARY KEY (StaffID, Role),

FOREIGN KEY (StaffID) REFERENCES STAFF (StaffID)

);

-- Insert data into Staff\_Role table

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (1, 'Event Manager');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (2, 'Catering Coordinator');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (3, 'Logistics Specialist');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (4, 'Entertainment Manager');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (5, 'Security Supervisor');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (6, 'Technical Support');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (7, 'Marketing Coordinator');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (8, 'Fashion Show Organizer');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (9, 'Photography Manager');

INSERT INTO Staff\_Role (StaffID, `Role`)

VALUES (10, 'Transportation Coordinator');

-- Table for Service\_Decoration (Multivalued Attribute for SERVICE)

CREATE TABLE

Service\_Decoration (

ServiceID INT,

Decoration VARCHAR(50),

PRIMARY KEY (ServiceID, Decoration),

FOREIGN KEY (ServiceID) REFERENCES SERVICE (ServiceID)

);

-- Insert data into Service\_Decoration table

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (1, 'Floral Arrangements');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (2, 'Balloon Decorations');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (3, 'Lighting Setup');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (4, 'Table Centerpieces');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (5, 'Stage Backdrops');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (6, 'Thematic Props');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (7, 'Ceiling Drapes');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (8, 'Photo Booth Setup');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (9, 'Outdoor Lighting');

INSERT INTO Service\_Decoration (ServiceID, Decoration)

VALUES (10, 'Custom Signage');

-- Table for Service\_Types (Multivalued Attribute for SERVICE)

CREATE TABLE

Service\_Types (

ServiceID INT,

`Type` VARCHAR(50),

PRIMARY KEY (ServiceID, Type),

FOREIGN KEY (ServiceID) REFERENCES SERVICE (ServiceID)

);

-- Insert data into Service\_Types table

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (1, 'Catering');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (2, 'Lighting');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (3, 'Floral Decoration');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (4, 'Furniture Rental');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (5, 'Security');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (6, 'Audio/Visual');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (7, 'Desserts');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (8, 'Photography');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (9, 'Cleaning');

INSERT INTO Service\_Types (ServiceID, `Type`)

VALUES (10, 'Transportation');

-- Table for Attendee\_Preferences (Multivalued Attribute for ATTENDEE)

CREATE TABLE

Attendee\_Preferences (

EventID INT,

Seat\_Assignment INT,

Preference VARCHAR(50),

PRIMARY KEY (EventID, Seat\_Assignment, Preference),

FOREIGN KEY (EventID, Seat\_Assignment) REFERENCES ATTENDEE (EventID, Seat\_Assignment)

);

-- Insert data into Attendee\_Preferences table

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (1, 1, 'Vegetarian Meal');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (1, 2, 'Window Seat');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (2, 1, 'Front Row Seat');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (2, 2, 'Vegan Meal');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (3, 1, 'Gluten-Free Meal');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (3, 2, 'Aisle Seat');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (4, 1, 'Extra Legroom');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (4, 2, 'Non-Alcoholic Drinks');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (5, 1, 'Quiet Zone');

INSERT INTO Attendee\_Preferences (EventID, Seat\_Assignment, Preference)

VALUES (5, 2, 'Kosher Meal');

-- Table for Offer\_Types (Multivalued Attribute for SPECIAL\_OFFER)

CREATE TABLE

Offer\_Types (

OfferID INT,

`Type` VARCHAR(50),

PRIMARY KEY (OfferID, Type),

FOREIGN KEY (OfferID) REFERENCES SPECIAL\_OFFER (OfferID)

);

-- Insert data into Offer\_Types table

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (1, 'Discount');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (2, 'Early Bird');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (3, 'Group Booking');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (4, 'Seasonal');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (5, 'Loyalty Reward');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (6, 'Flash Sale');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (7, 'Referral Bonus');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (8, 'Limited Time');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (9, 'Exclusive Access');

INSERT INTO Offer\_Types (OfferID, `Type`)

VALUES (10, 'VIP Package');

-- Table for Customer\_Phone\_Number (Multivalued Attribute for CUSTOMER)

CREATE TABLE

Customer\_Phone\_Number (

CustomerID INT,

Phone\_Number VARCHAR(15),

PRIMARY KEY (CustomerID, Phone\_Number),

FOREIGN KEY (CustomerID) REFERENCES CUSTOMER (CustomerID)

);

-- Insert data into Customer\_Phone\_Number table

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (1, '+961-1-123456');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (2, '+961-6-654321');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (3, '+961-9-555123');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (4, '+961-7-444987');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (5, '+961-8-333222');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (6, '+961-7-666555');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (7, '+961-8-777888');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (8, '+961-9-222333');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (9, '+961-6-888777');

INSERT INTO Customer\_Phone\_Number (CustomerID, Phone\_Number)

VALUES (10, '+961-5-999000');

-- Table for Supplier\_Phone\_Number (Multivalued Attribute for SUPPLIER)

CREATE TABLE

Supplier\_Phone\_Number (

SupplierID INT,

Phone\_Number VARCHAR(15),

PRIMARY KEY (SupplierID, Phone\_Number),

FOREIGN KEY (SupplierID) REFERENCES SUPPLIER (SupplierID)

);

-- Insert data into Supplier\_Phone\_Number table

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (1, '+961-1-123456');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (2, '+961-6-654321');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (3, '+961-9-555123');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (4, '+961-7-444987');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (5, '+961-8-333222');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (6, '+961-7-666555');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (7, '+961-8-777888');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (8, '+961-9-222333');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (9, '+961-6-888777');

INSERT INTO Supplier\_Phone\_Number (SupplierID, Phone\_Number)

VALUES (10, '+961-5-999000');

-- Table for Organizer\_Contact\_Information (Multivalued Attribute for ORGANIZER)

CREATE TABLE

Organizer\_Contact\_Information (

OrganizerID INT,

Contact\_Information VARCHAR(255),

PRIMARY KEY (OrganizerID, Contact\_Information),

FOREIGN KEY (OrganizerID) REFERENCES ORGANIZER (OrganizerID)

);

-- Insert data into Organizer\_Contact\_Information table

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (1, 'rharaty@lau.edu.lb');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (2, 'imad.mahmoud@lau.edu');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (3, 'ali.reda03@lau.edu ');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (4, 'reina.harake@lau.edu');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (5, 'ali.elhajj03@lau.edu');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (6, 'maher.saadi@lau.edu');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (7, 'cesar.ayache@example.com');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (8, 'jean.cherfane@example.com');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (9, 'alaa.doumit@example.com');

INSERT INTO Organizer\_Contact\_Information (OrganizerID, Contact\_Information)

VALUES (10, 'sara.fares@example.com');

-- Table for Organizer\_Role (Multivalued Attribute for ORGANIZER)

CREATE TABLE

Organizer\_Role (

OrganizerID INT,

`Role` VARCHAR(50),

PRIMARY KEY (OrganizerID, Role),

FOREIGN KEY (OrganizerID) REFERENCES ORGANIZER (OrganizerID)

);

-- Insert data into Organizer\_Role table

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (1, 'Event Planner');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (2, 'Logistics Manager');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (3, 'Marketing Specialist');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (4, 'Entertainment Coordinator');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (5, 'Venue Manager');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (6, 'Fundraising Coordinator');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (7, 'Technical Supervisor');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (8, 'Fashion Show Director');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (9, 'Book Launch Organizer');

INSERT INTO Organizer\_Role (OrganizerID, `Role`)

VALUES (10, 'New Year Event Coordinator');

-- Table for Venue\_Owner (Multivalued Attribute for VENUE)

CREATE TABLE

Venue\_Owner (

VenueID INT,

`Owner` VARCHAR(255),

PRIMARY KEY (VenueID, Owner),

FOREIGN KEY (VenueID) REFERENCES VENUE (VenueID)

);

-- Insert data into Venue\_Owner table

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (1, 'Beirut Holdings');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (2, 'Tripoli Events Group');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (3, 'Byblos Heritage Foundation');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (4, 'Sidon Skyline Ventures');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (5, 'Zahle Cultural Society');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (6, 'Tyre Coastal Development');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (7, 'Baalbek Historical Trust');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (8, 'Jounieh Bay Enterprises');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (9, 'Batroun Coastal Properties');

INSERT INTO Venue\_Owner (VenueID, `Owner`)

VALUES (10, 'Aley Mountain Resorts');

--This query finds the StaffID of staff members who worked on events

--sponsored by sponsors with a sponsorship amount greater than 10,000

--and provided services categorized as "Catering"

SELECT

St1.StaffID

FROM

STAFF AS St1

INNER JOIN WORKS\_ON AS W1 ON St1.StaffID = W1.StaffID

INNER JOIN SPONSORS AS S1 ON W1.EventID = S1.EventID

INNER JOIN SPONSOR AS SP1 ON S1.SponsorID = SP1.SponsorID

INNER JOIN USED\_IN AS U1 ON W1.EventID = U1.EventID

INNER JOIN SERVICE\_Types AS ST1 ON U1.ServiceID = ST1.ServiceID

WHERE

SP1.Sponsorship\_Amount > 10000

AND ST1.Type = 'Catering'

INTERSECT

SELECT

St2.StaffID

FROM

STAFF AS St2

INNER JOIN WORKS\_ON AS W2 ON St2.StaffID = W2.StaffID

INNER JOIN ORGANIZES AS O2 ON W2.EventID = O2.EventID

INNER JOIN ORGANIZER AS ORG2 ON O2.OrganizerID = ORG2.OrganizerID

WHERE

ORG2.Address LIKE '%Lebanon%'

AND St2.Wage > 1500;

--This query finds the CustomerID of customers who attended events

--organized by organizers located in Lebanon and where the

--event price is greater than 5000:

SELECT

C.CustomerID

FROM

CUSTOMER AS C

INNER JOIN HAS AS H ON C.CustomerID = H.CustomerID

INNER JOIN EVENT AS E ON H.EventID = E.EventID

INNER JOIN ORGANIZES AS O ON E.EventID = O.EventID

INNER JOIN ORGANIZER AS ORG ON O.OrganizerID = ORG.OrganizerID

WHERE

ORG.Address LIKE '%Lebanon%'

AND E.Price > 5000

INTERSECT

SELECT

C2.CustomerID

FROM

CUSTOMER AS C2

INNER JOIN HAS AS H2 ON C2.CustomerID = H2.CustomerID

INNER JOIN EVENT AS E2 ON H2.EventID = E2.EventID

INNER JOIN SPONSORS AS S ON E2.EventID = S.EventID

INNER JOIN SPONSOR AS SP ON S.SponsorID = SP.SponsorID

WHERE

SP.Sponsorship\_Amount > 10000;

--This query finds the SupplierID of suppliers who provided services

--for events held in venues located in Lebanon and where the event type is "Charity"

SELECT

S.SupplierID

FROM

SUPPLIER AS S

INNER JOIN PROVIDES AS P ON S.SupplierID = P.SupplierID

INNER JOIN USED\_IN AS U ON P.ServiceID = U.ServiceID

INNER JOIN EVENT AS E ON U.EventID = E.EventID

INNER JOIN HOLD AS H ON E.EventID = H.EventID

INNER JOIN VENUE AS V ON H.VenueID = V.VenueID

WHERE

V.Address LIKE '%Lebanon%'

INTERSECT

SELECT

S2.SupplierID

FROM

SUPPLIER AS S2

INNER JOIN PROVIDES AS P2 ON S2.SupplierID = P2.SupplierID

INNER JOIN USED\_IN AS U2 ON P2.ServiceID = U2.ServiceID

INNER JOIN EVENT AS E2 ON U2.EventID = E2.EventID

INNER JOIN Event\_Types AS ET ON E2.EventID = ET.EventID

WHERE

ET.Type = 'Charity';

--This query finds the EventID of events that have both a sponsorship

--amount greater than 15,000 and use services categorized as "Audio/Visual"

SELECT

E.EventID

FROM

EVENT AS E

INNER JOIN SPONSORS AS S ON E.EventID = S.EventID

INNER JOIN SPONSOR AS SP ON S.SponsorID = SP.SponsorID

WHERE

SP.Sponsorship\_Amount > 15000

INTERSECT

SELECT

E2.EventID

FROM

EVENT AS E2

INNER JOIN USED\_IN AS U ON E2.EventID = U.EventID

INNER JOIN SERVICE\_Types AS ST ON U.ServiceID = ST.ServiceID

WHERE

ST.Type = 'Audio/Visual';

--This query finds the CustomerID of customers who attended events

--where the venue owner is "Beirut Holdings" and the event type is "Wedding"

SELECT

C.CustomerID

FROM

CUSTOMER AS C

INNER JOIN HAS AS H ON C.CustomerID = H.CustomerID

INNER JOIN EVENT AS E ON H.EventID = E.EventID

INNER JOIN HOLD AS HO ON E.EventID = HO.EventID

INNER JOIN VENUE AS V ON HO.VenueID = V.VenueID

INNER JOIN Venue\_Owner AS VO ON V.VenueID = VO.VenueID

WHERE

VO.Owner = 'Beirut Holdings'

INTERSECT

SELECT

C2.CustomerID

FROM

CUSTOMER AS C2

INNER JOIN HAS AS H2 ON C2.CustomerID = H2.CustomerID

INNER JOIN EVENT AS E2 ON H2.EventID = E2.EventID

INNER JOIN Event\_Types AS ET ON E2.EventID = ET.EventID

WHERE

ET.Type = 'Wedding';

--This query finds the StaffID of staff members who worked on events

--where the service type is "Photography" and the event was

--sponsored by sponsors with a sponsorship amount greater than 12,000

SELECT

St.StaffID

FROM

STAFF AS St

INNER JOIN WORKS\_ON AS W ON St.StaffID = W.StaffID

INNER JOIN USED\_IN AS U ON W.EventID = U.EventID

INNER JOIN SERVICE\_Types AS ST ON U.ServiceID = ST.ServiceID

WHERE

ST.Type = 'Photography'

INTERSECT

SELECT

St2.StaffID

FROM

STAFF AS St2

INNER JOIN WORKS\_ON AS W2 ON St2.StaffID = W2.StaffID

INNER JOIN SPONSORS AS S ON W2.EventID = S.EventID

INNER JOIN SPONSOR AS SP ON S.SponsorID = SP.SponsorID

WHERE

SP.Sponsorship\_Amount > 12000;

--This query finds the CustomerID of customers who attended events,

-- used services, or were associated with special offers

-- Customers who attended events

(SELECT

C.CustomerID

FROM

CUSTOMER AS C

INNER JOIN HAS AS H ON C.CustomerID = H.CustomerID

INNER JOIN EVENT AS E ON H.EventID = E.EventID

WHERE

E.Price > 5000

)

UNION

(SELECT

C2.CustomerID

FROM

CUSTOMER AS C2

INNER JOIN HAS AS H2 ON C2.CustomerID = H2.CustomerID

INNER JOIN EVENT AS E2 ON H2.EventID = E2.EventID

INNER JOIN USED\_IN AS U ON E2.EventID = U.EventID

INNER JOIN SERVICE\_Types AS ST ON U.ServiceID = ST.ServiceID

WHERE

ST.Type = 'Catering'

)

UNION

(SELECT

C3.CustomerID

FROM

CUSTOMER AS C3

INNER JOIN SPECIAL\_OFFER AS SO ON C3.SO\_ID = SO.OfferID

WHERE

SO.Valid\_Date BETWEEN '2025-01-01' AND '2025-12-31'

);

--This query finds the EventID of events that were both organized

--by organizers located in "Beirut" and used services provided by

--suppliers with a fee greater than 4000

-- Events organized by organizers located in Beirut

(SELECT

E.EventID

FROM

EVENT AS E

INNER JOIN ORGANIZES AS O ON E.EventID = O.EventID

INNER JOIN ORGANIZER AS ORG ON O.OrganizerID = ORG.OrganizerID

WHERE

ORG.Address LIKE '%Beirut%'

)

INTERSECT

(SELECT

E2.EventID

FROM

EVENT AS E2

INNER JOIN USED\_IN AS U ON E2.EventID = U.EventID

INNER JOIN PROVIDES AS P ON U.ServiceID = P.ServiceID

INNER JOIN SUPPLIER AS S ON P.SupplierID = S.SupplierID

WHERE

CAST(S.Fee AS DECIMAL(10, 2)) > 4000

);

--query that used to find the EventID, Name, and total sponsorship amount

--for events that have more than 3 sponsors and use at least 2 different types of services

SELECT

E.EventID,

E.Name,

(SELECT SUM(SP.Sponsorship\_Amount)

FROM SPONSORS AS S

INNER JOIN SPONSOR AS SP ON S.SponsorID = SP.SponsorID

WHERE S.EventID = E.EventID) AS Total\_Sponsorship

FROM

EVENT AS E

WHERE

(SELECT COUNT(DISTINCT S.SponsorID)

FROM SPONSORS AS S

WHERE S.EventID = E.EventID) > 3

AND

(SELECT COUNT(DISTINCT ST.Type)

FROM USED\_IN AS U

INNER JOIN SERVICE\_Types AS ST ON U.ServiceID = ST.ServiceID

WHERE U.EventID = E.EventID) >= 2;

--This query finds the SupplierID and Name of suppliers who provided

--services for events attended by customers who have paid their full

--amount (i.e., Amount\_Due = 0) and where the event type is "Expo"

SELECT

DISTINCT S.SupplierID,

S.Name

FROM

SUPPLIER AS S

INNER JOIN PROVIDES AS P ON S.SupplierID = P.SupplierID

INNER JOIN USED\_IN AS U ON P.ServiceID = U.ServiceID

INNER JOIN EVENT AS E ON U.EventID = E.EventID

INNER JOIN HAS AS H ON E.EventID = H.EventID

INNER JOIN CUSTOMER AS C ON H.CustomerID = C.CustomerID

WHERE

C.Amount\_Due = 0

AND E.EventID IN (

SELECT ET.EventID

FROM Event\_Types AS ET

WHERE ET.Type = 'Expo'

);

##### **Normalizing Database up to BCNF normal form:**

After establishing all the necessary relations, the next step is to enhance the database structure through normalization. This process ensures that the data is efficiently organized by reducing redundancy and improving integrity. In this approach, we will normalize our database incrementally through the various levels of normalization, up to the Boyce-Codd Normal Form (BCNF), which is considered a stronger version of the third normal form. For each relation, we will systematically apply the normalization steps in sequence: starting with the First Normal Form (1NF), then moving to the Second (2NF), followed by the Third (3NF), and finally achieving BCNF. Before we begin, let us provide a brief overview of each of these normal forms.

|  |  |
| --- | --- |
| **Normal Form** | **Requirement** |
| **1NF** | All attributes are atomic (no sets, lists or nested structure) |
| **2NF** | Must be in 1NF, and no partial dependency of non-prime attributes on part of a composite key |
| **3NF** | Must be in 2NF, and removes transitive dependencies, meaning non-prime attributes cannot depend on other non-prime attributes. |
| **BCNF** | A stricter version of 3NF. For every functional dependency (X → Y), the determinant (X) must be a candidate key. |

**What are the normal Forms?**

**Regular Entity Types:**

**1.EVENT :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EventID | **Price** | **Start\_Date** | **End\_Date** | **Name** |

1NF: The EVENT relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF: The EVENT relation schema satisfies all conditions of the 2NF because every nonprime attribute is fully functionally dependent on the primary key “EVENTID.”

3NF: The EVENT relation schema does not satisfy all conditions of the 3NF because we have somewhat of a transitive dependency since **EventID->Start\_Date** and **Start\_Date->End\_Date** , in other terms End\_Date is defined through another non-prime attribute , to fix we further decompose.

**Event**:

|  |  |  |  |
| --- | --- | --- | --- |
| EventID | **Price** | **Start\_Date** | **Name** |

**Schedule**:

|  |  |  |
| --- | --- | --- |
| Start\_Date | **EventID** | **End\_Date** |

BCNF: After decomposing, BCNF was satisfied since its 3NF and for every functional dependency X → Y, X must be a candidate key.

**2.STAFF:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| StaffID | **Phone\_Number** | **Wage** | **First** | **Middle** | **Last** |

1NF: Satisfied, since even the composite attribute Name has been already flattened into atomic values, and Phone\_number and Wage are already such, so satisfies 1NF.

2NF: No partial dependency detected, since StaffID is a single primary key, therefore since 1NF is also satisfied, 2NF satisfied as well.

3NF: Satisfied, since 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “StaffID.”

BCNF: All attributes solely depend on the primary key “StaffID” as a determinant, and since its 3NF is also satisfied, then BCNF satisfied.

**3.SPONSOR:**

|  |  |  |  |
| --- | --- | --- | --- |
| SponsorID | **Name** | **Phone\_Number** | **Sponsorship\_Amount** |

1NF: The Sponsor relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF: No partial dependency detected, since SponsorID is a single primary key, therefore since 1NF is also satisfied, 2NF satisfied as well.

3NF: Satisfied, since 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “SponsorID.”

BCNF: All attributes solely depend on the primary key “SponsorID” as a determinant, and since its 3NF is also satisfied , then BCNF satisfied.

**4.VENUE:**

|  |  |  |  |
| --- | --- | --- | --- |
| VenueID | **Name** | **Phone\_Number** | **Address** |

1NF: The VENUE relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF: No partial dependency detected, since VenueID is a single primary key, therefore since 1NF is also satisfied, 2NF satisfied as well.

3NF: Satisfied, since 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “VenueID.”

BCNF:All attributes solely depend on the primary key “VenueID” as a determinant, and since its 3NF is also satisfied, then BCNF satisfied.

**5.SERVICE:**

|  |
| --- |
| ServiceID |

Well as we can see here this relational schema consists of one attribute, which is technically the primary key as well, therefore it satisfies all forms of normal forms (1NF, 2NF, 3NF, BCNF)

**6.SPECIAL OFFER:**

|  |  |
| --- | --- |
| OfferID | **Valid\_Date** |

Well, since this relational schema only consists of 2 attributes, one of which is the “primary key” we assume that it also satisfies all normal forms (1NF, 2NF, 3NF, BCNF)

**7.CUSTOMER:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CustomerID | **Email** | **First** | **Middle** | **Last** | **Payment\_Types** | **Amount\_Paid** | **Amount\_Due** | **Payment\_Day** |

1NF: The CUSTOMER relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF: No partial dependency detected, since CustomerID is a single primary key, therefore since 1NF is also satisfied, 2NF satisfied as well.

3NF: All though the multivalued attribute isn’t stated in the relational schema here, we can still assume that Amount\_due = Budget – Amount\_Paid , so we can likely assume as well a somewhat of a transitive dependency consisting of **CustomerID->Amount\_Paid->Amount\_Due which is a 3NF violation.** To fix that we can simply just remove Amount\_Due, which can be computed later via SQL.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CustomerID | **Email** | **First** | **Middle** | **Last** | **Payment\_Types** | **Amount\_Paid** | **Payment\_Day** |

This new schema satisfies the 3NF normal form.

BCNF: After removing Amount\_Due , we can safely say that this schema satisfies BCNF since all attributes solely depend on the primary key “CustomerID” as a determinant, and since its 3NF is now also satisfied, then BCNF satisfied.

**8.SUPPLIER:**

|  |  |  |  |
| --- | --- | --- | --- |
| SupplierID | **Name** | **Details** | **Fee** |

1NF: The Supplier relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF: No partial dependency detected, since SupplierID is a single primary key, therefore since 1NF is also satisfied, 2NF satisfied as well.

3NF: We can say that there is violation, assuming that Details->Fee where Fee is not a key nor Fee is a prime attribute, therefore, to fix that we can decompose the table into:

**Supplier:**

|  |  |  |
| --- | --- | --- |
| SupplierID | **Name** | **Details** |

**Supplier\_Fee:**

|  |  |
| --- | --- |
| Details | **Fee** |

Doing such would fix this transitive dependency and therefore satisfy the third normal form.

BCNF: After decomposing the schema, we can say that it also satisfies BCNF since all attributes solely depend on the primary key “ServiceID” as a determinant, and since its 3NF is now also satisfied, then BCNF satisfied.

**9.ORGANIZER:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrganizerID | **Address** | **First** | **Middle** | **Last** |

1NF: The ORGANIZER relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic. (Name composite attribute was decomposed already)

2NF: No partial dependency detected, since OrganizerID is a single primary key, therefore since 1NF is also satisfied, 2NF satisfied as well.

3NF: Satisfied, since 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “OrganizerID.”

BCNF:All attributes solely depend on the primary key “OrganizerID” as a determinant, and since its 3NF is also satisfied, then BCNF satisfied.

##### **Multivalued Attribute Tables:**

**1.Event\_Types:**

|  |  |
| --- | --- |
| **EventID** | **Type** |

**2.Staff\_roles:**

|  |  |
| --- | --- |
| **StaffID** | **Role** |

**3.Service\_Decoration:**

|  |  |
| --- | --- |
| **ServiceID** | **Decoration** |

**4.Service\_Types:**

|  |  |
| --- | --- |
| **ServiceID** | **Type** |

**5.Offer\_Types:**

|  |  |
| --- | --- |
| **OfferID** | **Type** |

**6.Customer\_Phone\_Number:**

|  |  |
| --- | --- |
| **CustomerID** | **Phone\_Number** |

**7.Supplier\_Phone\_Number:**

|  |  |
| --- | --- |
| **SupplierID** | **Phone\_Number** |

**8.Organizer\_Contact\_Information:**

|  |  |
| --- | --- |
| **OrganizerID** | **Contact\_Information** |

**9.Organizer\_Role:**

|  |  |
| --- | --- |
| **OrganizerID** | **Role** |

**10.Venue\_Owner:**

|  |  |
| --- | --- |
| **VenueID** | **Owner** |

Those Tables were already made in previous phases, to represent multivalued Attributes , thus eliminating some redundancy and confusion from lots of the Entity tables, and since those tables are structurally simple ( Based on 2 attributes of which one is a primary key ) thus all those relational schema satisfy all normal forms that were working on (1NF,2NF,3NF,BCNF)

##### **Weak Entity Tables:**

**ATTENDEE:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Status** | **Total\_Attendee** | **First** | **Middle** | **Last** |

1NF: The ATTENDEE relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic. (Name composite attribute was decomposed already)

2NF: A weak entity consists of a partial key, here its (EventID, Seat\_Assignement), we can assume that Total\_Attendee shouldn’t be tied with the seat assignment, thus creating an anomaly where there is partial dependency, since EventID->Total\_Attendee , so it doesn’t satisfy 2NF. To fix that we can break this weak entity into two tables:

**ATTENDEE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Status** | **First** | **Middle** | **Last** |

EVENT\_ATTENDEE\_TOTAL:

|  |  |
| --- | --- |
| **EventID** | **Total\_Attendee** |

Doing such would fix the anomaly, and therefore satisfy 2NF since all the other attributes don’t rely on partial dependencies.

3NF: All attributes directly and fully depend on the composite key (Seat\_Assignment and EVENTID) , and in the other tables its only 2 attributes so it already satisfies 3NF, and since we’ve satisfies 2NF by breaking down this weak entity, we can now say that 3NF is also satisfied.

BCNF: All attributes solely depend on the composite key “seat\_assignment, EventID” as a determinant, and since its 3NF is also satisfied, then BCNF satisfied. (as well as in the other new table)

##### **Weak Multivalued Attribute:**

**Attendee\_preference:**

|  |  |  |
| --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Preference** |

Like the other multivalued tables, since this table (all though technically consisting of 3 attributes, but EventID and Seat\_Assignement are considered to be a composite key) this tables satisfies normal forms since all its values are atomic (1NF satisfied) no partial dependency (2NF satisfied) no transitive (3NF satisfied) and Preference solely relies on to composite key as a determinant

##### **Relationship Tables:**

**With non-prime attributes (basically BCNF from the get-go):**

**Holds(M:N):**

|  |  |
| --- | --- |
| **EventID** | **VenueID** |

**USED\_IN(M:N):**

|  |  |
| --- | --- |
| **EventID** | **ServiceID** |

**WORKS\_ON(M:N):**

|  |  |
| --- | --- |
| **StaffID** | **EventID** |

**SPONSORS(M:N):**

|  |  |
| --- | --- |
| **SponsorID** | **EventID** |

**HAS(M:N):**

|  |  |
| --- | --- |
| **CustomerID** | **EventID** |

**PROVIDES(M:N):**

|  |  |
| --- | --- |
| **SupplierID** | **ServiceID** |

##### **Other Relations:**

**1.CHOOSES(1:1):**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Customer\_ID** | **SO\_ID** | **Email** | **First** | **Middle** | **Last** | **Payment\_Types** | **Amount\_Paid** | **Amount\_Due** | **Payment\_Day** |

1NF: This schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF, 3NF, BCNF: This table violates all 3, in fact it seems unnecessary to use this table, it was originally created to link between the special offer and customer, so all those unnecessary attributes should be refuted thus this tables becomes a 2 key table:

**CHOOSES(1:1):**

|  |  |
| --- | --- |
| **Customer\_ID** | **SO\_ID** |

With all natural forms being satisfied

**2.ORGANIZES(M:N):**

|  |  |  |
| --- | --- | --- |
| **OrganizerID** | **EventID** | **Responsibility** |

1NF: This schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF: Based on the description of the table, claiming a many to many relationships, thus there is no partial dependency where the non-prime attribute is partially dependent on one of the keys, and not on the composite key (OrganizerID, EventID) thus 2NF satisfied.

3NF: No transitive dependency, since responsibility relies only on the composite key, therefore 3NF satisfied.

BCNF: responsibility solely depend on the composite key “OrganizerID, EventID” as a determinant, and since its 3NF is also satisfied, then BCNF satisfied.

**3.PROVIDES FOR(M:N):**

|  |  |  |  |
| --- | --- | --- | --- |
| **SupplierID** | **EventID** | **Invoice\_Number** | **Supply\_Status** |

1NF: This schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF: Based on the description of the table, claiming a many to many relationships, thus there is no partial dependency where the non-prime attribute is partially dependent on one of the keys, and not on the composite key (SupplierID, EventID) thus 2NF satisfied.

3NF: violated, since we can assume that Invoice\_Number->Supply\_status , therefore transitive dependency violating 3NF, to fix that, we break the table here into two tables:

**PROVIDES FOR(M:N):**

|  |  |  |
| --- | --- | --- |
| **SupplierID** | **EventID** | **Invoice\_Number** |

**INVOICE\_STATUS:**

|  |  |
| --- | --- |
| **Invoice\_Number** | **Supply\_Status** |

Doing such would ensure no transitive dependency, thus 3NF satisfied.

BCNF: after breaking down the table into two, we can now say that attributes solely depend on their key as a determinant (whether the key is (SupplierID, EventID) or Invoice\_Number) thus BCNF satisfied.

**4.ATTENDS(1:N):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EventID** | **Seat\_Assignment** | **Status** | **Total\_Attendee** | **First** | **Middle** | **Last** |

The anomaly and the way to fix it was already addressed in “**weak entity table**” part

**5.MANAGES(1:N):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StaffID | **OrganizerID** | **Phone\_Number** | **Wage** | **First** | **Middle** | **Last** |

1NF: This schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

2NF, 3NF, BCNF: This table violates all 3, in fact it seems unnecessary to use this table, it was originally created to link between the STAFF and ORGANIZER, so all those unnecessary attributes should be refuted thus this tables becomes a 2 key table:

MANAGES(1:M):

|  |  |
| --- | --- |
| StaffID | **OrganizerID** |

With it now satisfying all normal forms since it’s a 2 key table