***DBMS PROJECT REPORT –***

***EVENT MANAGEMENT***

***Submitted by:*** ***1) Jeel Shah***

***2) Priyanka***

***3) Marthala Supraja Reddy***

***4) Mansi Mahesh singh***

***5) Jallepalli Prerna***



***CONTENT***

|  |
| --- |
| 1. INTRODUCTION |
| 1. ER MODEL |
| 1. ER DIAGRAM |
| 1. ER DIAGRAM TO RELATIONAL MODEL |
| 1. NORMALIZING THE DATABASE |
| 1. CONVERTING INTO RELATIONAL TABLES |
| 1. SQL CODE |
| 1. SOFTWARE USED |
| 1. OTHER DETAILS –TRIGGERS |

***INTRODUCTION***

Database management systems(DBMSs) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyse data. The event management database we have it for event managers. In this database manager will get every information about every event which is done and which are going on in future.

Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. Most 3NF tables are free of insertion, update, and deletion anomalies

This project includes the program included that helps us to check which normal form dependencies of a table satisfy.

**The objective of this project is to design the database** e**vent Management Database which manages and stores all possible event related informations. To find which Normal Form the dependencies for a particular relation lies in.**

***ESSENCE OF THE PROJECT***

This project is especially designed for event managers. the event manager will be able to feed in keep record and project reports of his work . it provides a user friendly approach for handling all the services .some of the important features of the project are

* Events: gives information about all the requirements of services and products of the event being organized.
* Reports: reports can be generated for tracking success/process at different levels .
* Vendor -Specific List: generate specialized list of products and services ordered from a specific vendor.
* Easy Tracking: keeps simplified track of employee /manager/ head appointed to an event.
* Scope of Improvement: allows for improvement by reviewing user feedback.

Present event managers (mostly small scale) face problems in tracking the requirements of an event when they are handling multiple events at a time. Special employees are also appointed for managing the records for the events. The project provides an easy and automated approach that saves time, money, and mind-memory.

The project ensures real time feeding of data in a secure place which avoids data loss in offline logging due to any physical mishap and saves time wasted in unnecessary paper-work.

This project has been developed to overcome difficulties in utilization of present existing systems in general. Here users to the system are the event management company heads, managers and employees.

***PROBLEM DESCRIPTION***

This project is designed for event managers. The event manager will be able to feed in, keep record, project reports of his work. It provides a user friendly approach for handling all the services. Every event has its own type, date, time, feedback, transportation charges ,budget and an unique event ID. It has customers who can sponsor various events and have first name, last name ,address, phone no, username ,password, discount and unique customer ID. Every event needs various Electronic Equipments containing name, cost, unique ID, which are supplied by various Electronic companies having their unique ID, Address, email id ,name. For decoration purpose there are different types of decoration and their respective costs which will be supplied by different decorators having name, unique ID, mail id and address. Since there are many events held, each of these events are headed by one Manager.

Various employees are appointed to an event having first name, last name,unique E\_ID, phone no, address, salary. These employees could be either managers or a part of the logistics team. Each Manager has an unique manager ID and department. Each of the logistics team members have work\_ allotment and unique ID.

For making events more interesting informals are conducted. Every informal has its own unique informal ID , name, mail id, type, address. Events orders for the food of different types and their respective cost which is supplied by caterers who have their unique id, name, address, mail id. Each and every event is located at a particular venue .Venue is specified by its unique Id, name, capacity, rent, phone, address. Transportation is provided to an event is defined by the type of vehicle customer requires,cost per km and the number of seats that vehicle has. Transportation is provided by Transport Company with name, id, phone number, address. Some events like stage plays, marriages require makeup. There are different make up services which will be provided and each of the makeup service has its own cost and type. The makeup artists are from parlours. Since there are different parlours, hence each of them has its own specific name, unique id ,mail id and address. Since moments in the event have to be captured, so these are done by the photographers who cover the entire event. Each photographer has mail id , address, pay, ID, first name,lastname for further contacting.

For entertaining the guests in the event, different performances are held. These performances are performed in the event by performers who have their unique ID, name, salary, mail id and address. Performers are classified into dancers, musicians, fillers and anchors. Each dancer has his/her unique ID and their forte. Similarly anchors have gender and unique ID. Fillers who help in entertaining the guests in the events appear in between the performances like mimicry, comedians etc who have their own unique ID and type. There will be music which will be performed and each different type of music has its unique music ID and type.

***ER MODEL***  
**Our ER Model contains the following entities:**

**Entity Scheme:-**

1. managers(emp\_id, department)
2. logistics(emp\_id,workallotted)
3. dancers(per\_id,forte)
4. transportation(cost\_per\_km,no\_of\_seats,vehicle\_name)
5. musician(per\_id,type)
6. fillers(per\_id,type)
7. anchor(per\_id,gender)
8. parlour (par\_id,par\_name,par\_mail\_id,par\_add)
9. makeup(type,charges)
10. venue(v\_id,v\_name,capacity,v\_add,v\_rent,v\_phn)
11. t\_company(t\_id,t\_phn,t\_\_add,t\_name)
12. caterers(cat\_id,cat\_name,cat\_mail\_id,cat\_add)
13. food(cost,type)
14. decorators(dec\_id,dec\_add,dec\_mail\_id,dec\_name)
15. decoration(type,charge)
16. Ecompany(e\_id,e\_add,e\_name,e\_mail\_id)
17. Elec\_equip(el\_id,el\_name,cost)
18. photographers(P\_Id,P\_Fname,P\_Lname,P\_Add,P\_mail\_id,P\_charges)
19. informals(I\_id,I\_name,type,I\_add,I\_mail\_id,cost)
20. event(eve\_id,eve\_date,eve\_time,eve\_feedback,eve\_type,eve\_budget,eve\_trans\_charges)
21. performer(per\_id,per\_fname,per\_lname,per\_mail\_id,per\_add,per\_salary
22. employees(emp\_id,emp\_fname,emp\_lname,emp\_add,emp\_phn,emp\_salary)
23. customer(cust\_id, first\_name, last\_name, email\_id, password,     username,discount,c\_phone,c\_address)
24. venueaddress  (v\_id, pincode , city, state, line1 , line2)

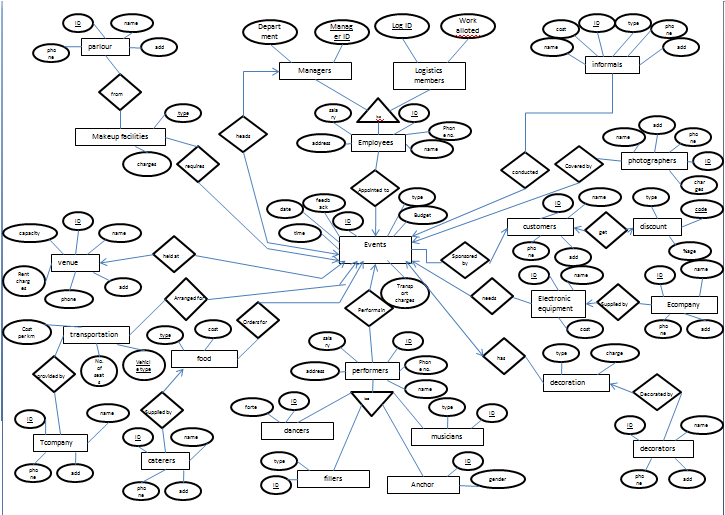
**Relations scheme:-**

1. decorated\_by(dec\_id,type) :-   one-one relation
2. supplied\_by(e\_id,el\_id) :-  many-many relation
3. from(par\_id,type) :- one-one relation
4. provided\_by(t\_id,vehicle\_name) :- many-many relation
5. served\_by(type,cat\_id) :- many-many relation
6. requires(Eve\_id,type) :- one-many relation
7. held\_at(Eve\_id,v\_id) :-  many-one relation

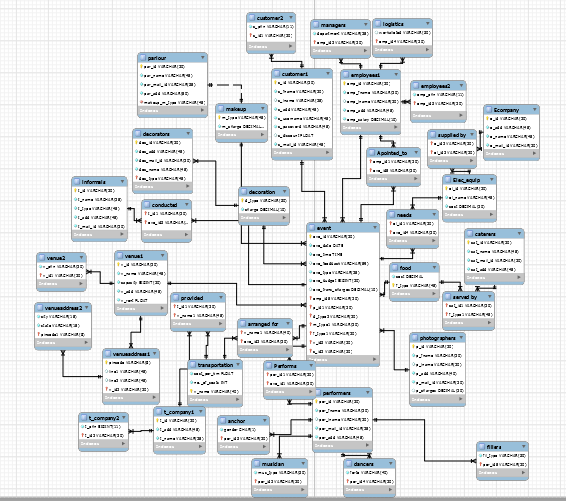
32. arranged\_for(Eve\_id,vehicle\_name) :- many-many relation

1. order\_for(Eve\_id,type) :- many-one relation
2. needs(Eve\_id,el\_id) :- many-many relation
3. has(Eve\_id,type) :- many-one relation
4. sponsored\_by(c\_id,Eve\_id) :- many-one relation
5. covered\_by(Eve\_id,P\_id) :- many-one relation
6. conducted(Eve\_id,I\_id) :- many-many relation
7. performs\_in(Eve\_id,per\_id) :- many-many relation
8. appointed\_to(Eve\_id,emp\_id) :- many-many relation
9. heads(Eve\_id,emp\_id) :- many-one relation
10. located\_at(v\_id, v\_code) :- one-one relation

**Conversion of ER to Relational Model:**

****

**ER Diagram to Relational Model:**

****

***Table Schema Description:***

1. **managers(emp\_id, department)**

**Primary key:** emp\_id

**Candidate key:** emp\_id

**Prime attributes:** emp\_id

**Functional dependencies:**

    emp\_id→department

There is only 1 FD so the managers is in BCNF

**2) logistics( emp\_id, workalloted)**

**Primary key:** emp\_id

**Candidate key:** emp\_id

**Prime attributes:** emp\_id

**Functional dependencies:**

    emp\_id→workalloted

There is only 1 FD so the logistics is in BCNF

**3) customer(cust\_id, first\_name, last\_name, email\_id, password,     username,discount,c\_phone,c\_address)**

**Primary key:** c\_id

c\_phone can be multivalued thus the table needs to be divided.Therefore:

i) **customer1(cust\_id, c\_fname, c\_lname, email\_id, password, username,discount,c\_address)**

ii) **customer2(c\_id,c\_phone)**

**i) customer1**(c\_id, c\_fname, c\_lname, c\_mail\_id, password,  username,discount)

**Primary key : c\_id**

Candidate keys/Prime attributes : c\_id, username, c\_mail\_id,( c\_fname c\_lname c\_address)

**Functional dependencies -**

c\_id→  c\_fname, c\_lname, c\_mail\_id, password, username, discount

c\_mail\_id→ c\_fname, c\_lname, c\_id, password, username, discount

username→ c\_fname, c\_lname, c\_mail\_id, password, c\_id, discount

c\_fname c\_lname c\_address →  c\_mail\_id, password, c\_id, discount, username

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**ii) customer2(c\_id,c\_phone)**

**Primary key:** c\_id

**Candidate key:** c\_id

**Prime attributes:** c\_id

**Functional dependencies:**

    c\_id→phone

There is only 1 FD so the customer2 is in BCNF

**4). venue(v\_id,v\_name,capacity,v\_add,v\_rent,v\_phn)**

**Primary Key :** v\_id

           v\_phn can have multivalued data because of phone number hence table                  needs to be divided

**venue1(v\_id, v\_name,capacity,v\_add,v\_rent)**

**venue2(v\_id,v\_phn)**

**venue1(v\_id,v\_name,capacity,v\_add,v\_rent)**

**Primary key:**v\_id

**Candidate key:**v\_id,  v\_name v\_add,

**Prime attributes:-**v\_id, v\_name, v\_add

**Functional  Dependencies:-**

v\_id →v\_name, capacity, v\_add, v\_rent

                     v\_name v\_add →v\_id, capacity, v\_rent

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**venue2(v\_id,v\_phn)**

**Primary key:**v\_id

**Candidate key:**v\_id

**Prime attributes:-**v\_id

**Functional  Dependencies:-**

v\_id →v\_phn

There is only one functional dependency so it is in BCNF

**5) dancers( per\_id, forte)**

**Primary key:** per\_id

**Candidate key:** per\_id

**Prime attributes:** per\_id

**Functional dependencies:**

    per\_id→forte

There is only one functional dependency so it is in BCNF

**6)  musician( per\_id, mus\_type)**

**Primary key:** per\_id

**Candidate key:** per\_id

**Prime attributes:** per\_id

**Functional dependencies:**

    Per\_id→ mus\_type

There is only one functional dependency so it is in BCNF

**7). fillers( per\_id, fil\_type)**

**Primary key:** per\_id

**Candidate key:** per\_id

**Prime attributes:** per\_id

**Functional dependencies:**

    per\_id→fil\_type

There is only one functional dependency so it is in BCNF

**8). anchor( per\_id, gender)**

**Primary key:** per\_id

**Candidate key:** per\_id

**Prime attributes:** per\_id

**Functional dependencies:**

    per\_id→gender

There is only one functional dependency so it is in BCNF

**9). transportation(cost\_per\_km,no\_of\_seats,vehicle\_name)**

**Primary key:** vehicle\_name

**Candidate key:** vehicle\_name

**Prime attributes:** vehicle\_name

**Functional dependencies:**

    vehicle\_name → cost\_per\_km, no\_of\_seats

There is only one functional dependency so it is in BCNF

**10). parlour(**par\_id,par\_name,par\_mail\_id,par\_add**)**

**Primary key:** par\_id

**Candidate key:** par\_id, (par\_name par\_add), par\_mail\_id

**Prime attributes:** par\_id, par\_mail\_id, par\_add, par\_name

**Functional dependencies:**

    par\_id→ par\_name, par\_mail\_id, par\_add

               par\_mail\_id → par\_id, par\_add, par\_name

               par\_name par\_add → par\_id, par\_mail\_id

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**11). food(cost,f\_type)**

**Primary key:  f\_** type

**Candidate key:  f\_**type

**Prime attributes:-  f\_**type

**Functional  Dependencies:-**

                    f\_type → cost

There is only one functional dependency so it is in BCNF

**12). caterers(cat\_id,cat\_name,cat\_mail\_id,cat\_add)**

**Primary key:**cat\_id,

**Candidate key:**cat\_id,   cat\_mail\_id,   (cat\_name cat\_add)

**Prime attributes:-**cat\_id, cat\_name,  cat\_mail\_id,  cat\_add

**Functional  Dependencies:-**

cat\_id → cat\_name, cat\_mail\_id, cat\_add

                    cat\_mail\_id → cat\_id, cat\_name, cat\_add

            cat\_name cat\_add → cat\_id, cat\_mail\_id

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**13) performer(per\_id,per\_fname,per\_lname,per\_mail\_id,per\_add,per\_salary)**

**Primary key:** per\_id

**Candidate key:** per\_id, per\_mail\_id, (per\_fname per\_lname per\_add)

**Prime attributes:** per\_id, per\_mail\_id, per\_fname, per\_lname, per\_add

**Functional dependencies:**

per\_id  → per\_fname,per\_lname,per\_mail\_id,per\_add,per\_salary

   per\_mail\_id  → per\_fname,per\_lname,per\_id,per\_add,per\_salary

per\_fname per\_lname per\_add → per\_mail\_id,per\_id,per\_salary

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**14) makeup(m\_type, charge)**

**Primary key:** m\_type

**Candidate key:** m\_type

**Prime attributes:** m\_type

**Functional dependencies:**

                   m\_type → charge

There is only one functional dependency so it is ini BCNF

**15) decorators(dec\_id,dec\_add,dec\_mail\_id,dec\_name)**

**Primary key:** dec\_id

**Candidate key:** dec\_id, (dec\_name dec\_add), dec\_mail\_id

**Prime attributes:** dec\_id, dec\_mail\_id, dec\_name, dec\_add

**Functional dependencies:**

    dec\_id→ dec\_name, dec\_mail\_id, dec\_add

               dec\_mail\_id → dec\_id, dec\_add, dec\_name

               dec\_name dec\_add → dec\_id, dec\_mail\_id

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**16) decoration(d\_type,d\_charge)**

**Primary key:** d\_type

**Candidate key:** d\_type

**Prime attributes:** d\_type

**Functional dependencies:**

                   d\_type → d\_charge

As there is only one functional dependency so it is in BCNF.

**17) Ecompany(e\_id,e\_add,e\_name,e\_mail\_id)**

**Primary key:** e\_id

**Candidate key:** e\_id, (e\_name e\_add), e\_mail\_id

**Prime attributes:** e\_id, e\_mail\_id, e\_add, e\_name

**Functional dependencies:**

    e\_id→ e\_name, e\_mail\_id, e\_add

               e\_mail\_id → e\_id, e\_add, e\_name

               e\_name e\_add → e\_id, e\_mail\_id

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**18) Elec\_equip(el\_id,el\_name,cost)**

**Primary key:** el\_id

**Candidate key:** el\_id, el\_name

**Prime attributes:** el\_id, el\_name

**Functional dependencies:**

                   el\_id→ el\_name, cost

                   el\_name→ el\_id, cost

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**19). photographers(p\_id,p\_fname,p\_lname,p\_add,p\_mail\_id,p\_charges)**

**Primary key:**p\_id

**Candidate key:**p\_mail\_id, p\_id,( p\_fname p\_lname p\_add)

**Prime attributes:-**p\_mail\_id, p\_fname, p\_lname, p\_add, p\_id

**Functional  Dependencies:-**

 p\_id  →   p\_mail\_id, p\_fname, p\_lname, p\_add,p\_charges

                          p\_mail\_id  →  p\_id, p\_fname, p\_lname,p\_add,p\_charges

                          p\_fname p\_lname p\_add →  p\_id, p\_mail\_id, p\_charges

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**20) informals(I\_id,I\_name,I\_type,I\_add,I\_mail\_id,I\_cost)**

**Primary key:** I\_id

**Candidate key:** I\_id,( I\_name I\_add), I\_mail\_id, I\_type

**Prime attributes:** I\_id, I\_mail\_id, I\_name, I\_add,I\_type

**Functional dependencies:**

    I\_id→ I\_name, I\_mail\_id, I\_add, type, I\_cost

               I\_mail\_id → I\_id, I\_add, I\_name, type, I\_cost

               I\_name I\_add → I\_id, I\_mail\_id, type, I\_cost

               type → I\_id, I\_mail\_id, I\_add, I\_name, I\_cost

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**21). t\_company(t\_id,t\_phn,t\_\_add,t\_name)**

**Primary key:** t\_id

t\_phn can have multivalued data because of phone number hence table

            needs to be divided

**t\_company1(t\_id,t\_\_add,t\_name)**

**t\_company2(t\_id,t\_phn)**

      i).  **t\_company1(t\_id,t\_\_add,t\_name)**

**Primary key:**t\_id

**Candidate key:**t\_id,  t\_name t\_add,

**Prime attributes:-**t\_id, t\_name, t\_add

**Functional  Dependencies:-**

t\_id →t\_name, t\_add

                   t\_name t\_add →t\_id

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

ii). **t\_company2(t\_id,t\_phn)**

**Primary key:**t\_id

**Candidate key:**t\_id

**Prime attributes:-**t\_id

**Functional  Dependencies:-**

t\_id →t\_phn

There is only one functional dependency so it is in BCNF

**22) venueaddress(v\_id, pincode, city, state, line)**

**Primary key** : (v\_id, pincode)

**Candidate key** : (v\_id, pincode)

**Prime attributes** : v\_id, pincode

**Functional Dependencies -**

              v\_id → pincode,  city, state, line1, line2

          pincode → city

          pincode → state

          city → state

**Canonical cover -**

          v\_id pincode →  city, state, line1, line2

          pincode → city, state

         Since (pincode) is a subset of the primary key the table is not even in 2NF. Hence

         we will decompose the table as follows -

**venueaddress1**(v\_id, pincode, line1, line2)

**venueaddress2**(pincode, city, state)

          As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**23) employees(emp\_id,emp\_fname,emp\_lname,emp\_add,emp\_phn,emp\_salary)**

**Primary key:** emp\_id

            emp\_phn has multivalued data and can be divided further. Therefore:

    **i)  employees1(emp\_id,emp\_fname,emp\_lname,emp\_add,emp\_salary)**

**ii) employees2(emp\_id, emp\_phn)**

**i) employees1(emp\_id,emp\_fname,emp\_lname,emp\_add,emp\_salary)**

            Primary key: emp\_id

**Candidate key:** emp\_id, (emp\_fname emp\_lname emp\_add), emp\_mail\_id

**Prime attributes:** emp\_id, emp\_mail\_id, emp\_fname, emp\_lname, emp\_add

**Functional dependencies:**

    emp\_id→ emp\_fname,emp\_lname,emp\_mail\_id,emp\_add,emp\_salary

               emp\_mail\_id → emp\_id,emp\_add, emp\_fname,emp\_lname,emp\_salary

               emp\_fname emp\_lname emp\_add → emp\_id, emp\_mail\_id,emp\_salary

As left side dependencies are a superkey in each FD. Hence, it is in BCNF. If the relational schema is in BCNF then all the redundancy based on functional dependencies has been removed, although other types of redundancy may still exist. A relational schema R is in Boyce-codd normal form if and only if for every one of its dependencies is a trivial functional dependency, provided the determinant is a super key.

**ii) employees2(emp\_id,emp\_phn)**

**Primary key:** emp\_id

**Candidate key:** emp\_id, emp\_phn

**Prime attributes:** emp\_id

**Functional dependencies:**

                   emp\_id→ emp\_phn

There is only one functional dependency therefore it is in BCNF

**24) event (eve\_id, eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,    eve\_trans\_charges)**

**Primary key:** eve\_id

**Candidate key:** eve\_id

**Prime attributes:** eve\_id

**Functional dependencies:**

          eve\_id → eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,

                           eve\_trans\_charges

There is only one FD so it is in BCNF

**Relations :-**

**1). decorated\_by(dec\_id,d\_type)**

         This relationship connects decoration with decorators and it says,

            one decoration can be decorated by one decorator and one decorator can do

            one type of decoration.

            There are different types of decorations that are available for example

             Flower, balloon, etc and only one decorator can decorate any one of them.

**Primary key:** dec\_id,d\_type

**Candidate key:** dec\_id, d\_type

**Prime attributes:** dec\_id, d\_type

**Functional dependencies:**

         No Functional Dependencies

    Hence it is in **BCNF.**

**2). supplied\_by(e\_id,el\_id)**

This relation connects Electronic equipment with ECompany,

           Many electronic

**Primary key:** e\_id,el\_id

**Candidate key:** e\_id,el\_id

**Prime attributes:** e\_id,el\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**3).from(par\_id,m\_type)**

**Primary key:** par\_id,m\_type

**Candidate key:** par\_id,m\_type

**Prime attributes:** par\_id,m\_type

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**4). provided\_by(t\_id,v\_name)**

**Primary key:** t\_id, **v\_name**

**Candidate key:**  t\_id, **v\_name**

**Prime attributes:**  t\_id, **v\_name**

**Functional dependencies:**

No Functional Dependencies

      Hence it is in **BCNF.**

**5). served\_by(type,cat\_id)**

**Primary key:** type, cat\_id

**Candidate key:**  type, cat\_id

**Prime attributes:**  type, cat\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**6). requires(Eve\_id,type)**

**Primary key:** Eve\_id, type

**Candidate key:**  Eve\_id, type

**Prime attributes:**  Eve\_id, type

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**7). held(Eve\_id, v\_id)**

**Primary key:** Eve\_id, v\_id

**Candidate key:**  Eve\_id, v\_id

**Prime attributes:**  Eve\_id, v\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**8). arranged\_for(Eve\_id, type\_of\_vehicle)**

**Primary key:** Eve\_id, type\_of\_vehicle

**Candidate key:**  Eve\_id, type\_of\_vehicle

**Prime attributes:**  Eve\_id, type\_of\_vehicle

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**9).  order\_for(Eve\_id, type)**

**Primary key:** Eve\_id, type

**Candidate key:**  Eve\_id, type

**Prime attributes:**  Eve\_id, type

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**10). needs(Eve\_id, el\_id)**

**Primary key:** Eve\_id, el\_id

**Candidate key:**  Eve\_id, el\_id

**Prime attributes:**  Eve\_id, el\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**11).  has(Eve\_id, type)**

**Primary key:** Eve\_id, type

**Candidate key:**  Eve\_id, type

**Prime attributes:**  Eve\_id, type

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**12). sponsored\_by(c\_id,Eve\_id)**

**Primary key:** Eve\_id, c\_id

**Candidate key:**  Eve\_id, c\_id

**Prime attributes:**  Eve\_id, c\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**13).  covered\_by(Eve\_id,P\_id)**

**Primary key:** Eve\_id,P\_id

**Candidate key:**  Eve\_id, P\_id

**Prime attributes:**  Eve\_id, P\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**14).  conducted(Eve\_id,I\_id)**

**Primary key:** Eve\_id, I\_id

**Candidate key:**  Eve\_id,I\_id

**Prime attributes:**  Eve\_id, I\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**15). performs\_in(Eve\_id,per\_id)**

**Primary key:** Eve\_id, per\_id

**Candidate key:**  Eve\_id, per\_id

**Prime attributes:**  Eve\_id, per\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**16).** appointed\_to(Eve\_id,emp\_id)

**Primary key:** Eve\_id, emp\_id

**Candidate key:**  Eve\_id, emp\_id

**Prime attributes:**  Eve\_id, emp\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

**17).** heads(Eve\_id,emp\_id)

**Primary key:** Eve\_id, emp\_id

**Candidate key:**  Eve\_id, emp\_id

**Prime attributes:**  Eve\_id, emp\_id

**Functional dependencies:**

No Functional Dependencies

       Hence it is in **BCNF.**

***CONVERTING INTO RELATIONAL TABLES:***

**Many to one relationships:**

1. events(**eve\_id** not null, eve\_date, eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,eve\_trans\_charge,primary key(eve\_id),FOREIGN KEY (m\_type) REFERENCES makeup(m\_type) on delete cascade on update cascade)
2. events(**eve\_id** not null, eve\_date, eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,eve\_trans\_charge,primary key(eve\_id),FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id) on delete cascade on update cascade)
3. events(**eve\_id** not null, eve\_date, eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,eve\_trans\_charge,primary key(eve\_id),FOREIGN KEY (p\_id) REFERENCES photographers(p\_id) on delete cascade on update cascade)
4. events(**eve\_id** not null, eve\_date, eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,eve\_trans\_charge,primary key(eve\_id),FOREIGN KEY (v\_id) REFERENCES venue1(v\_id) on delete cascade on update cascade)
5. events(**eve\_id** not null, eve\_date, eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,eve\_trans\_charge,primary key(eve\_id),FOREIGN KEY (c\_id) REFERENCES customer1(c\_id) on delete cascade on update cascade)
6. events(**eve\_id** not null, eve\_date, eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,eve\_trans\_charge,primary key(eve\_id),FOREIGN KEY (f\_type) REFERENCES food(f\_type) on delete cascade on update cascade)
7. events(**eve\_id** not null, eve\_date, eve\_date, eve\_time, eve\_feedback, eve\_type, eve\_budget,eve\_trans\_charge,primary key(eve\_id),FOREIGN KEY (d\_type) REFERENCES decoration(d\_type) on delete cascade on update cascade)

**Converting ISA hierarchy**

1. managers (emp\_id primary key not null, department, FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id))
2. logistics (emp\_id primary key not null, workallotted, FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id))
3. dancers (per\_id primary key not null, forte, FOREIGN KEY (per\_id) REFERENCES performers(per\_id))
4. fillers (per\_id primary key not null, fil\_type, FOREIGN KEY (per\_id) REFERENCES performers(per\_id))
5. anchors (per\_id primary key not null, gender, FOREIGN KEY (per\_id) REFERENCES performers(per\_id))
6. musicians (per\_id primary key not null, mus\_type, FOREIGN KEY (per\_id) REFERENCES performers(per\_id))

**Converting Many to Many relationships:**

1. Appointed to(emp\_id , eve\_id ,primary key(emp\_id , eve\_id), FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id),  FOREIGN KEY (eve\_id) REFERENCES events(eve\_id));
2. conducted(I\_id , eve\_id ,primary key(I\_id , eve\_id), FOREIGN KEY (I\_id) REFERENCES informals(I\_id),  FOREIGN KEY (eve\_id) REFERENCES events(eve\_id));
3. Supplied by(e\_id , el\_id ,primary key(e\_id , el\_id), FOREIGN KEY (e\_id) REFERENCES ecompany(e\_id),  FOREIGN KEY (el\_id) REFERENCES electronic equipments(el\_id));
4. provided(t\_id , v\_name ,primary key(t\_id , v\_name, FOREIGN KEY (t\_id) REFERENCES tcompany1(t\_id),  FOREIGN KEY (v\_name) REFERENCES transportation(v\_name));
5. Served by(cat\_id , f\_type ,primary key(cat\_id , f\_type), FOREIGN KEY (cat\_id) REFERENCES caterers(cat\_id),  FOREIGN KEY (f\_type) REFERENCES food(f\_type));
6. Arranged for(v\_name , eve\_id ,primary key(v\_name , eve\_id), FOREIGN KEY (v\_name) REFERENCES transportation(v\_name),  FOREIGN KEY (eve\_id) REFERENCES events(eve\_id));
7. needs(el\_id , eve\_id ,primary key(el\_id , eve\_id), FOREIGN KEY (el\_id) REFERENCES electronic equipments(el\_id),  FOREIGN KEY (eve\_id) REFERENCES events(eve\_id));
8. Performs in(per\_id , eve\_id ,primary key(per\_id , eve\_id), FOREIGN KEY (per\_id) REFERENCES performers(per\_id),  FOREIGN KEY (eve\_id) REFERENCES events(eve\_id));

***SQL CODE:***

1 . CREATE TABLE `mydb`.`customer1` (

`c\_id` VARCHAR(20) NOT NULL,

`c\_fname` VARCHAR(20) NOT NULL,

`c\_lname` VARCHAR(25) NULL,

`c\_add` VARCHAR(45) NOT NULL,

`c\_username` VARCHAR(45) NOT NULL,

`c\_password` VARCHAR(45) NOT NULL,

`c\_discount` FLOAT NULL,

`c\_mail\_id` VARCHAR(45) NOT NULL,

UNIQUE INDEX `c\_username\_UNIQUE` (`c\_username` ),

UNIQUE INDEX `c\_mail\_id\_UNIQUE` (`c\_mail\_id` ),

UNIQUE INDEX `c\_id\_UNIQUE` (`c\_id` ),

PRIMARY KEY (`c\_id`))

1. CREATE TABLE `mydb`.`customer2` (

`c\_phn` VARCHAR(11) NOT NULL,

`c\_id1` VARCHAR(20) NOT NULL,

PRIMARY KEY (`c\_id1`),

UNIQUE INDEX `c\_id\_UNIQUE` (`c\_id1` ),

CONSTRAINT `c\_id1`

FOREIGN KEY (`c\_id1`)

REFERENCES `mydb`.`customer1` (`c\_id`)

1. CREATE TABLE `mydb`.`employees1` (

`emp\_id` VARCHAR(20) NOT NULL,

`emp\_fname` VARCHAR(20) NOT NULL,

`emp\_lname` VARCHAR(20) NULL,

`emp\_add` VARCHAR(45) NOT NULL,

`emp\_salary` DECIMAL(10) NOT NULL,

PRIMARY KEY (`emp\_id`),

UNIQUE INDEX `emp\_id\_UNIQUE` (`emp\_id` ))

1. CREATE TABLE IF `mydb`.`logistics` (

`workalloted` VARCHAR(30) NULL,

`emp\_id4` VARCHAR(20) NOT NULL,

PRIMARY KEY (`emp\_id4`),

CONSTRAINT `emp\_id4`

FOREIGN KEY (`emp\_id4`)

REFERENCES `mydb`.`employees1` (`emp\_id`)

1. CREATE TABLE `mydb`.`performers` (

`per\_id` VARCHAR(20) NOT NULL,

`per\_fname` VARCHAR(20) NOT NULL,

`per\_lname` VARCHAR(20) NULL,

`per\_mail\_id` VARCHAR(25) NOT NULL,

`per\_add` VARCHAR(45) NOT NULL,

PRIMARY KEY (`per\_id`),

UNIQUE INDEX `per\_id\_UNIQUE` (`per\_id` ),

UNIQUE INDEX `per\_mail\_id\_UNIQUE` (`per\_mail\_id` ))

1. CREATE TABLE `mydb`.`dancers` (

`forte` VARCHAR(40) NOT NULL,

`per\_id4` VARCHAR(20) NOT NULL,

PRIMARY KEY (`per\_id4`),

CONSTRAINT `per\_id4`

FOREIGN KEY (`per\_id4`)

REFERENCES `mydb`.`performers` (`per\_id`))

1. CREATE TABLE `mydb`.`venue1` (

`v\_id` VARCHAR(20) NOT NULL,

`v\_name` VARCHAR(45) NOT NULL,

`capacity` BIGINT(20) NOT NULL,

`v\_add` VARCHAR(45) NOT NULL,

`v\_rent` FLOAT NOT NULL,

PRIMARY KEY (`v\_id`),

UNIQUE INDEX `v\_id\_UNIQUE` (`v\_id` ))

1. CREATE TABLE `mydb`.`venue2` (

`v\_phn` VARCHAR(20) NOT NULL,

`v\_id1` VARCHAR(20) NOT NULL,

PRIMARY KEY (`v\_id1`),

CONSTRAINT `v\_id`

FOREIGN KEY (`v\_id1`)

REFERENCES `mydb`.`venue1` (`v\_id`))

1. CREATE TABLE `mydb`.`managers` (

`department` VARCHAR(25) NOT NULL,

`emp\_id2` VARCHAR(20) NOT NULL,

PRIMARY KEY (`emp\_id2`),

CONSTRAINT `emp\_id2`

FOREIGN KEY (`emp\_id2`)

REFERENCES `mydb`.`employees1` (`emp\_id`))

1. CREATE TABLE `mydb`.`musician` (

`mus\_type` VARCHAR(30) NOT NULL,

`per\_id2` VARCHAR(20) NOT NULL,

PRIMARY KEY (`per\_id2`),

CONSTRAINT `per\_id2`

FOREIGN KEY (`per\_id2`)

REFERENCES `mydb`.`performers` (`per\_id`))

1. CREATE TABLE `mydb`.`fillers` (

`fil\_type` VARCHAR(30) NOT NULL,

`per\_id5` VARCHAR(20) NOT NULL,

PRIMARY KEY (`per\_id5`),

CONSTRAINT `per\_id5`

FOREIGN KEY (`per\_id5`)

REFERENCES `mydb`.`performers` (`per\_id`))

1. CREATE TABLE `mydb`.`anchor` (

`gender` CHAR(1) NOT NULL,

`per\_id3` VARCHAR(20) NOT NULL,

PRIMARY KEY (`per\_id3`),

CONSTRAINT `per\_id3`

FOREIGN KEY (`per\_id3`)

REFERENCES `mydb`.`performers` (`per\_id`))

1. CREATE TABLE `mydb`.`decoration` (

`d\_type` VARCHAR(30) NOT NULL,

`charge` DECIMAL(10) NOT NULL,

PRIMARY KEY (`d\_type`),

UNIQUE INDEX `d\_type\_UNIQUE` (`d\_type` ))

1. CREATE TABLE `mydb`.`decorators` (

`dec\_id` VARCHAR(20) NOT NULL,

`dec\_add` VARCHAR(45) NOT NULL,

`dec\_mail\_id` VARCHAR(20) NOT NULL,

`dec\_name` VARCHAR(45) NOT NULL,

`dec\_type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`dec\_id`, `dec\_type`),

UNIQUE INDEX `dec\_id\_UNIQUE` (`dec\_id` ),

UNIQUE INDEX `dec\_mail\_id\_UNIQUE` (`dec\_mail\_id` ),

INDEX `dec\_type\_idx` (`dec\_type` ),

CONSTRAINT `dec\_type`

FOREIGN KEY (`dec\_type`)

REFERENCES `mydb`.`decoration` (`d\_type`))

1. CREATE TABLE `mydb`.`transportation` (

`cost\_per\_km` FLOAT NOT NULL,

`no.\_of\_seats` INT NOT NULL,

`v\_name` VARCHAR(40) NOT NULL,

PRIMARY KEY (`v\_name`))

1. CREATE TABLE `mydb`.`makeup` (

`m\_type` VARCHAR(45) NOT NULL,

`m\_charge` DECIMAL(10) NOT NULL,

PRIMARY KEY (`m\_type`))

1. CREATE TABLE `mydb`.`parlour` (

`par\_id` VARCHAR(20) NOT NULL,

`par\_name` VARCHAR(45) NOT NULL,

`par\_mail\_id` VARCHAR(35) NOT NULL,

`par\_add` VARCHAR(50) NOT NULL,

`makeup\_m\_type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`par\_id`),

INDEX `fk\_parlour\_makeup1\_idx` (`makeup\_m\_type` ),

CONSTRAINT `makeup\_m\_type`

FOREIGN KEY (`makeup\_m\_type`)

REFERENCES `mydb`.`makeup` (`m\_type`)

1. CREATE TABLE `mydb`.`food` (

`cost` DECIMAL NOT NULL,

`f\_type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`f\_type`),

UNIQUE INDEX `f\_type\_UNIQUE` (`f\_type` ))

1. CREATE TABLE `mydb`.`caterers` (

`cat\_id` VARCHAR(20) NOT NULL,

`cat\_name` VARCHAR(45) NOT NULL,

`cat\_mail\_id` VARCHAR(20) NOT NULL,

`cat\_add` VARCHAR(45) NOT NULL,

PRIMARY KEY (`cat\_id`),

UNIQUE INDEX `cat\_mail\_id\_UNIQUE` (`cat\_mail\_id` ))

1. CREATE TABLE `mydb`.`Ecompany` (

`e\_id` VARCHAR(20) NOT NULL,

`e\_add` VARCHAR(45) NOT NULL,

`e\_name` VARCHAR(45) NOT NULL,

`e\_mail\_id` VARCHAR(20) NOT NULL,

PRIMARY KEY (`e\_id`),

UNIQUE INDEX `e\_mail\_id\_UNIQUE` (`e\_mail\_id` ),

UNIQUE INDEX `e\_add\_UNIQUE` (`e\_add` ))

1. CREATE TABLE `mydb`.`Elec\_equip` (

`el\_id` VARCHAR(20) NOT NULL,

`el\_name` VARCHAR(45) NOT NULL,

`cost` DECIMAL(20) NOT NULL,

PRIMARY KEY (`el\_id`),

UNIQUE INDEX `el\_id\_UNIQUE` (`el\_id` ))

1. CREATE TABLE `mydb`.`photographers` (

`p\_id` VARCHAR(20) NOT NULL,

`p\_fname` VARCHAR(20) NOT NULL,

`p\_lname` VARCHAR(20) NULL,

`p\_add` VARCHAR(40) NOT NULL,

`p\_mail\_id` VARCHAR(20) NOT NULL,

`p\_charges` DECIMAL(20) NULL,

PRIMARY KEY (`p\_id`),

UNIQUE INDEX `p\_id\_UNIQUE` (`p\_id` ),

UNIQUE INDEX `p\_mail\_id\_UNIQUE` (`p\_mail\_id` ))

1. CREATE TABLE `mydb`.`informals` (

`I\_id` VARCHAR(20) NOT NULL,

`I\_name` VARCHAR(25) NOT NULL,

`I\_type` VARCHAR(45) NOT NULL,

`I\_add` VARCHAR(45) NOT NULL,

`I\_mail\_id` VARCHAR(20) NOT NULL,

PRIMARY KEY (`I\_id`),

UNIQUE INDEX `I\_id\_UNIQUE` (`I\_id` ),

UNIQUE INDEX `I\_mail\_id\_UNIQUE` (`I\_mail\_id` ),

UNIQUE INDEX `I\_add\_UNIQUE` (`I\_add` ),

UNIQUE INDEX `I\_type\_UNIQUE` (`I\_type` ))

1. CREATE TABLE `mydb`.`t\_company1` (

`t\_id` VARCHAR(20) NOT NULL,

`t\_add` VARCHAR(45) NOT NULL,

`t\_name` VARCHAR(25) NOT NULL,

PRIMARY KEY (`t\_id`))

1. CREATE TABLE `mydb`.`t\_company2` (

`t\_phn` BIGINT(11) NOT NULL,

`t\_id2` VARCHAR(20) NOT NULL,

UNIQUE INDEX `t\_phn\_UNIQUE` (`t\_phn` ),

PRIMARY KEY (`t\_id2`),

CONSTRAINT `t\_id2`

FOREIGN KEY (`t\_id2`)

REFERENCES `mydb`.`t\_company1` (`t\_id`))

1. CREATE TABLE `mydb`.`venueaddress1` (

`pincode` VARCHAR(8) NOT NULL,

`line1` VARCHAR(45) NULL,

`line2` VARCHAR(45) NULL,

`v\_id2` VARCHAR(20) NOT NULL,

PRIMARY KEY (`pincode`, `v\_id2`),

UNIQUE INDEX `pincode\_UNIQUE` (`pincode` ),

INDEX `v\_id2\_idx` (`v\_id2` ),

CONSTRAINT `v\_id2`

FOREIGN KEY (`v\_id2`)

REFERENCES `mydb`.`venue1` (`v\_id`)

1. CREATE TABLE `mydb`.`venueaddress2` (

`city` VARCHAR(15) NOT NULL,

`state` VARCHAR(15) NOT NULL,

`pincode1` VARCHAR(8) NOT NULL,

PRIMARY KEY (`pincode1`),

CONSTRAINT `pincode1`

FOREIGN KEY (`pincode1`)

REFERENCES `mydb`.`venueaddress1` (`pincode`))

1. CREATE TABLE `mydb`.`employees2` (

`emp\_phn` VARCHAR(11) NOT NULL,

`emp\_id3` VARCHAR(20) NOT NULL,

UNIQUE INDEX `emp\_phn\_UNIQUE` (`emp\_phn` ),

PRIMARY KEY (`emp\_id3`),

CONSTRAINT `emp\_id3`

FOREIGN KEY (`emp\_id3`)

REFERENCES `mydb`.`employees1` (`emp\_id`))

1. CREATE TABLE IF NOT EXISTS `mydb`.`event` (

`eve\_id` VARCHAR(20) NOT NULL,

`eve\_date` DATE NOT NULL,

`eve\_time` TIME NOT NULL,

`eve\_feedback` VARCHAR(59) NULL,

`eve\_type` VARCHAR(25) NOT NULL,

`eve\_budget` BIGINT(20) NOT NULL,

`eve\_trans\_charges` DECIMAL(10) NULL,

`emp\_id5` VARCHAR(20) NOT NULL,

`p\_id1` VARCHAR(20) NOT NULL,

`d\_type2` VARCHAR(20) NOT NULL,

`m\_type1` VARCHAR(20) NOT NULL,

`f\_type1` VARCHAR(20) NOT NULL,

`v\_id3` VARCHAR(20) NOT NULL,

`c\_id2` VARCHAR(20) NOT NULL,

PRIMARY KEY (`eve\_id`, `emp\_id5`, `p\_id1`, `d\_type2`, `m\_type1`, `f\_type1`, `v\_id3`, `c\_id2`),

UNIQUE INDEX `eve\_id\_UNIQUE` (`eve\_id` ),

INDEX `emp\_id5\_idx` (`emp\_id5` ),

INDEX `p\_id1\_idx` (`p\_id1` ),

INDEX `v\_id3\_idx` (`v\_id3` ),

INDEX `c\_id2\_idx` (`c\_id2` ),

INDEX `f\_type1\_idx` (`f\_type1` ),

INDEX `d\_type2\_idx` (`d\_type2` ),

INDEX `m\_type1\_idx` (`m\_type1` ),

CONSTRAINT `emp\_id5`

FOREIGN KEY (`emp\_id5`)

REFERENCES `mydb`.`employees1` (`emp\_id`)

CONSTRAINT `p\_id1`

FOREIGN KEY (`p\_id1`)

REFERENCES `mydb`.`photographers` (`p\_id`)

CONSTRAINT `v\_id3`

FOREIGN KEY (`v\_id3`)

REFERENCES `mydb`.`venue1` (`v\_id`)

CONSTRAINT `c\_id2`

FOREIGN KEY (`c\_id2`)

REFERENCES `mydb`.`customer1` (`c\_id`)

CONSTRAINT `f\_type1`

FOREIGN KEY (`f\_type1`)

REFERENCES `mydb`.`food` (`f\_type`)

CONSTRAINT `d\_type2`

FOREIGN KEY (`d\_type2`)

REFERENCES `mydb`.`decoration` (`d\_type`)

CONSTRAINT `m\_type1`

FOREIGN KEY (`m\_type1`)

REFERENCES `mydb`.`makeup` (`m\_type`))

1. CREATE TABLE `mydb`.`Apointed\_to` (

`emp\_id1` VARCHAR(20) NOT NULL,

`eve\_id5` VARCHAR(20) NOT NULL,

PRIMARY KEY (`emp\_id1`, `eve\_id5`),

INDEX `eve\_id\_idx` (`eve\_id5` ),

CONSTRAINT `emp\_id1`

FOREIGN KEY (`emp\_id1`)

REFERENCES `mydb`.`employees1` (`emp\_id`)

CONSTRAINT `eve\_id5`

FOREIGN KEY (`eve\_id5`)

REFERENCES `mydb`.`event` (`eve\_id`))

31 . CREATE TABLE `mydb`.`conducted` (

`I\_id1` VARCHAR(20) NOT NULL,

`eve\_id3` VARCHAR(20) NOT NULL,

PRIMARY KEY (`I\_id1`, `eve\_id3`),

CONSTRAINT `I\_id1`

FOREIGN KEY (`I\_id1`)

REFERENCES `mydb`.`informals` (`I\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `eve\_id3`

FOREIGN KEY (`eve\_id3`)

REFERENCES `mydb`.`event` (`eve\_id`))

1. CREATE TABLE `mydb`.`supplied by` (

`e\_id2` VARCHAR(20) NOT NULL,

`el\_id2` VARCHAR(20) NOT NULL,

PRIMARY KEY (`e\_id2`, `el\_id2`),

INDEX `el\_id\_idx` (`el\_id2` ),

CONSTRAINT `e\_id2`

FOREIGN KEY (`e\_id2`)

REFERENCES `mydb`.`Ecompany` (`e\_id`)

CONSTRAINT `el\_id2`

FOREIGN KEY (`el\_id2`)

REFERENCES `mydb`.`Elec\_equip` (`el\_id`))

1. CREATE TABLE `mydb`.`provided` (

`t\_id1` VARCHAR(20) NOT NULL,

`v\_name1` VARCHAR(45) NOT NULL,

PRIMARY KEY (`t\_id1`, `v\_name1`),

CONSTRAINT `t\_id`

FOREIGN KEY (`t\_id1`)

REFERENCES `mydb`.`t\_company1` (`t\_id`)

CONSTRAINT `v\_name`

FOREIGN KEY (`v\_name1`)

REFERENCES `mydb`.`transportation` (`v\_name`))

1. CREATE TABLE `mydb`.`served by` (

`cat\_id1` VARCHAR(20) NOT NULL,

`f\_type1` VARCHAR(45) NOT NULL,

PRIMARY KEY (`cat\_id1`, `f\_type1`),

UNIQUE INDEX `cat\_id1\_UNIQUE` (`cat\_id1` ),

UNIQUE INDEX `f\_type1\_UNIQUE` (`f\_type1` ),

CONSTRAINT `cat\_id`

FOREIGN KEY (`cat\_id1`)

REFERENCES `mydb`.`caterers` (`cat\_id`)

CONSTRAINT `f\_type`

FOREIGN KEY (`f\_type1`)

REFERENCES `mydb`.`food` (`f\_type`))

1. CREATE TABLE `mydb`.`arranged for` (

`v\_name1` VARCHAR(40) NOT NULL,

`eve\_id2` VARCHAR(20) NOT NULL,

PRIMARY KEY (`v\_name1`, `eve\_id2`),

CONSTRAINT `v\_name1`

FOREIGN KEY (`v\_name1`)

REFERENCES `mydb`.`transportation` (`v\_name`)

CONSTRAINT `eve\_id2`

FOREIGN KEY (`eve\_id2`)

REFERENCES `mydb`.`event` (`eve\_id`))

1. CREATE TABLE `mydb`.`needs` (

`el\_id1` VARCHAR(20) NOT NULL,

`eve\_id4` VARCHAR(20) NOT NULL,

PRIMARY KEY (`el\_id1`, `eve\_id4`),

INDEX `eve\_id\_idx` (`eve\_id4` ),

CONSTRAINT `el\_id1`

FOREIGN KEY (`el\_id1`)

REFERENCES `mydb`.`Elec\_equip` (`el\_id`)

CONSTRAINT `eve\_id4`

FOREIGN KEY (`eve\_id4`)

REFERENCES `mydb`.`event` (`eve\_id`)

)

1. CREATE TABLE `mydb`.`Performs` (

`per\_id1` VARCHAR(20) NOT NULL,

`eve\_id1` VARCHAR(20) NOT NULL,

PRIMARY KEY (`per\_id1`, `eve\_id1`),

INDEX `eve\_id\_idx` (`eve\_id1` ),

CONSTRAINT `per\_id1`

FOREIGN KEY (`per\_id1`)

REFERENCES `mydb`.`performers` (`per\_id`)

CONSTRAINT `eve\_id1`

FOREIGN KEY (`eve\_id1`)

REFERENCES `mydb`.`event` (`eve\_id`)

)

***SOFTWARE USED***:

Back-end design: MYSQL work bench 6.3CE.

***OTHER DETAILS:***

TRIGGERS –These are used in calculating budget of particular event.