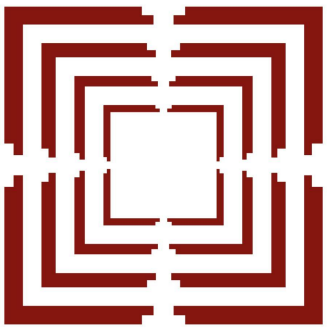


Ahmedabad
University

CSE250 : Database Management System
Winter Semester 2021

Guided by Professor Shefali Naik

Project Title:
Event Management System

Name	Enrolment Number
Malav Doshi	AU1940017
Samkit Kundalia	AU1940021
Parth Shah	AU1940065

Index

Description

We have developed an Event Management System keeping in mind the end users and stakeholders like - guests, registered participants, organising team, vendors and sponsors.

On the homepage of the system, the logged in users can check any upcoming events with all details for them. They may also register (RSVP) for it. Registering for the event executes a trigger which sends them the confirmation for the same.

The organisers (admin) level users can add events and other details. They can also view the finances of the event - including sponsorship packages, vendors, inventory and registration fees.

Each organising team is divided into departments and teams. It is to smoothen the process and execution of events by sharing responsibilities. Each department has a relation to the vendor to coordinate and communicate with them for event preparations. Each material/item brought from the vendor is stored in the inventory which provides detailed records - and prevents the scope for any future unwanted expenses.

There are sponsorship packages for sponsors to choose from for events. Each sponsor can sponsor any number of events based on those packages provided and seek for deliverables.

We have also created a facility for handling finances related to the event. Each transaction is recorded for accountability - including registration fees, vendor bills, sponsorship payments and other miscellaneous incomes/expenses. The registration fees are logged into the billing system

We have also created a mailing system to remind guests and ask for feedback - post event - which is then forwarded to the team for review.

System Requirements

1) Any Web Browser
2) Python (or Python3)
3) MySQL

Initialisation

1) Open terminal/git bash in your system
2) !git clone https://github.com/parthmshah1302/DBMS-Event-Management-System
3) !cd DBMS-Event-Management-System
4) !pip install virtualenv
5) !python -m virtualenv env
6) !.\env\Scripts\activate
7) !pip install -r requirements.txt
8) Run the DBMS-EVS.sql, DBMS-Insert.sql and DBMS-ProcFuncTrig.sql from /SQLFiles in that particular order.
9) Create a db.yaml file in your project's root directory.
10) Make sure your db.yaml file looks like this:

    mysql_host: 'localhost'
    mysql_user: 'root'
    mysql_password: <ROOT-PASSWORD>
    mysql_db: 'dbmsEventManagement'

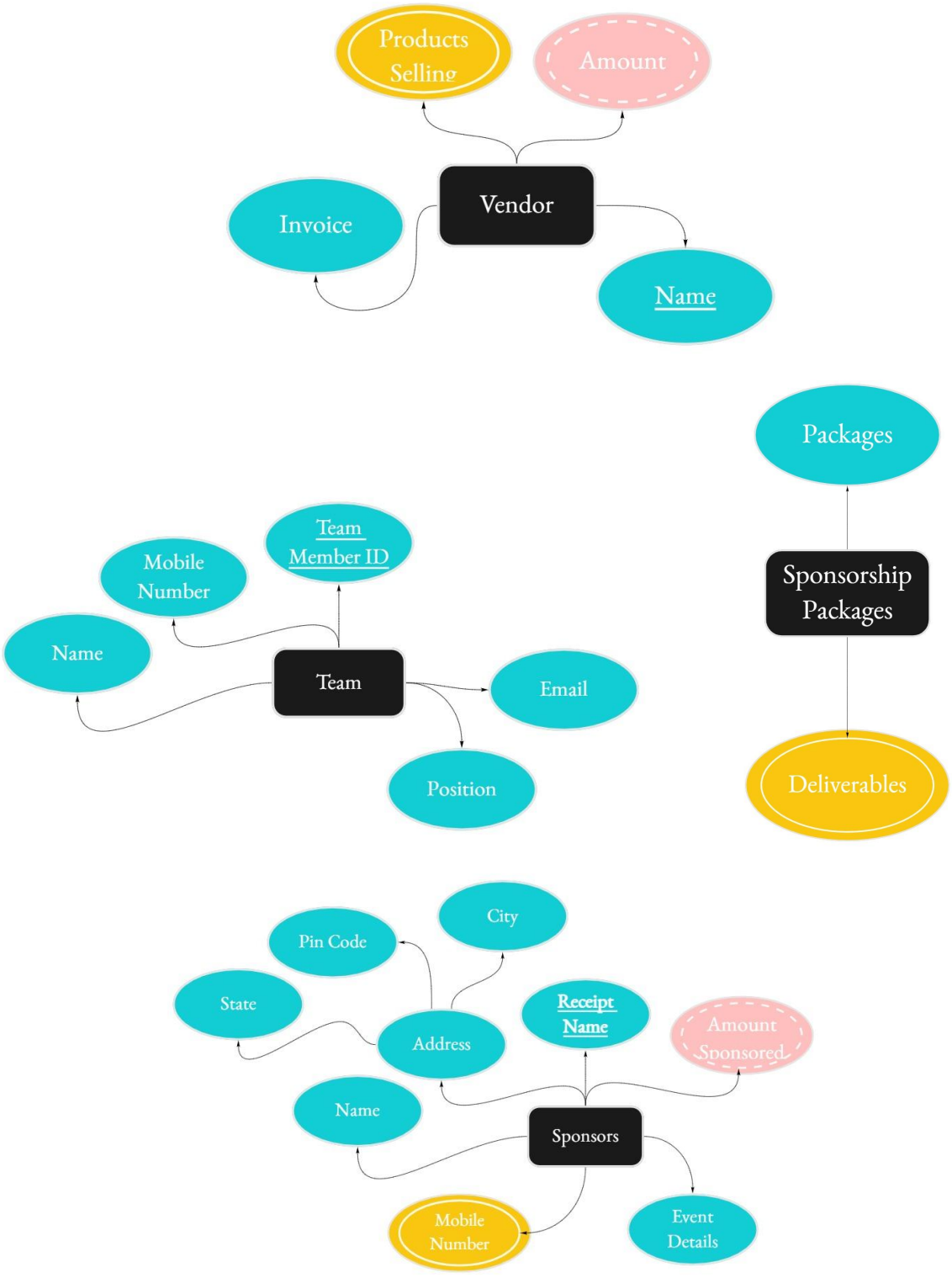
11) !python app.py

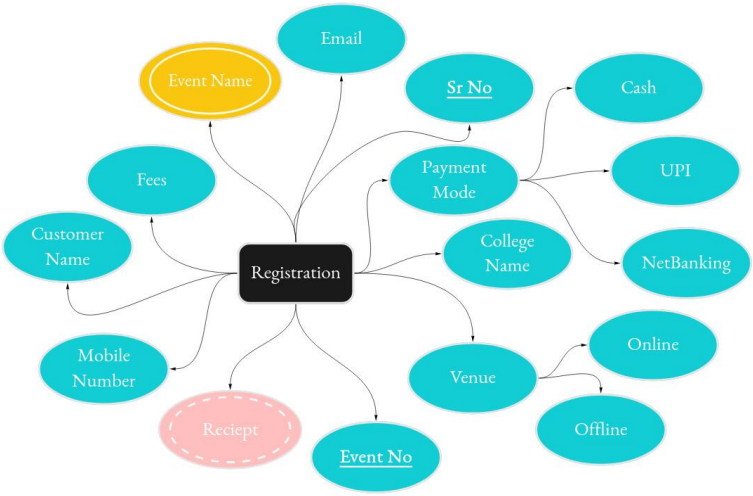
Running

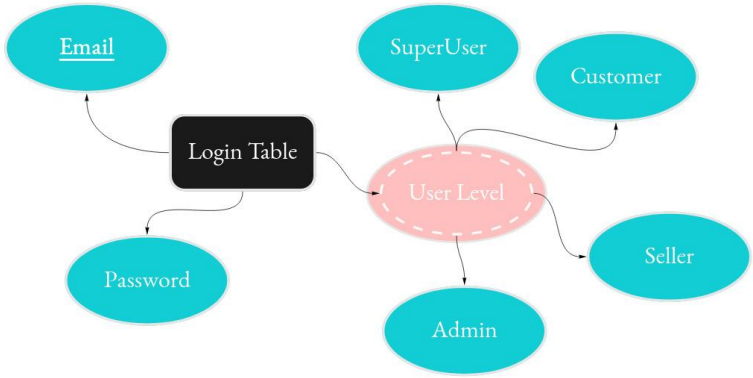
1) !cd DBMS-Event-Management-System
2) !.\env\Scripts\activate
3) !python app.py

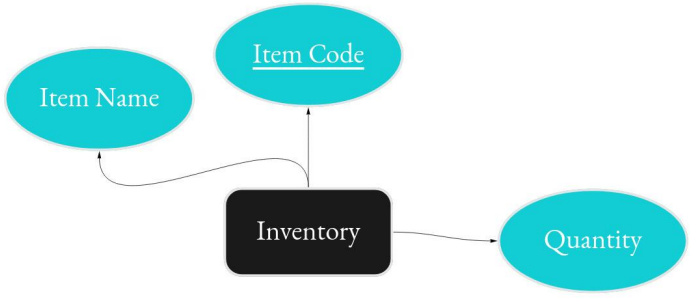
Now your Event Management System is running on http://127.0.0.1:5000

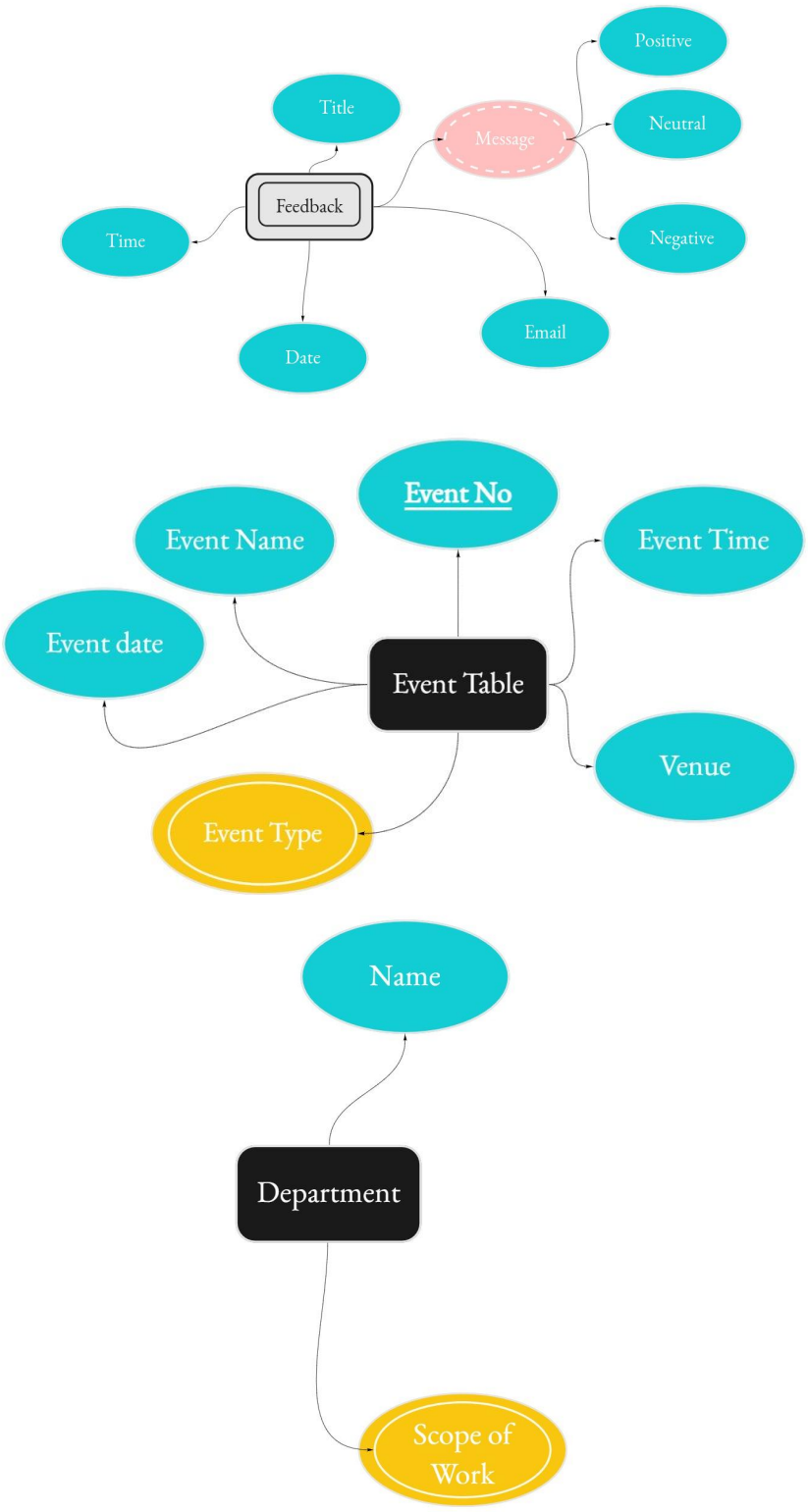
Entity-Relationship Diagram(🔗)

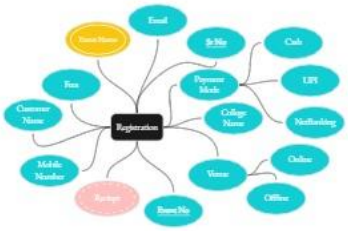


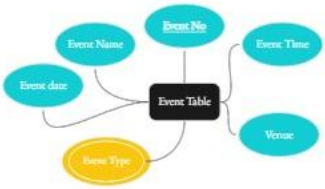

Email
Event Name
Sr No
Cash
Fees
Payment Mode
UPI
Customer Name
Registration
College Name
NetBanking
Mobile Number
Online
Venue
Reciept
Offline
Event No


Email
SuperUser
Customer
Login Table
User Level
Password
Seller
Admin


Item Code
Item Name
Inventory
Quantity

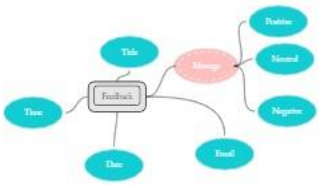

Positive
Title
Message
Neutral
Feedback
Time
Negative
Email
Date
Event No
Event Name
Event Time
Event date
Event Table
Venue
Event Type
Name
Department
Scope of Work


Email
Cash
UPI
Net Banking
Online
Offline
Registration
Payment Mode
College Name
Venue
Mobile Number
Fees
Customer Name

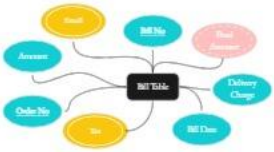

Event No
Event Name
Event Time
Event date
Event Table
Venue
Event Type

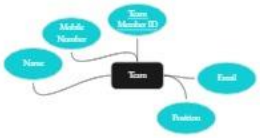

Email
SuperUser
Customer
Login Table
User Level
Seller
Password
Admin


Products Sellers
Amount
Vendor
Invoice
Name

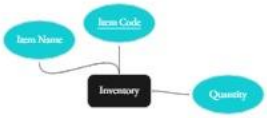

Positive
Neutral
Negative
Title
Message
Feedback
Time
Email
Date


City
Pin Code
State
Receipt Name
Address
Name
Sponsors
Event Details


Email
Bill No
Amount
Delivery Charge
Online No
Bill Table
Bill Date

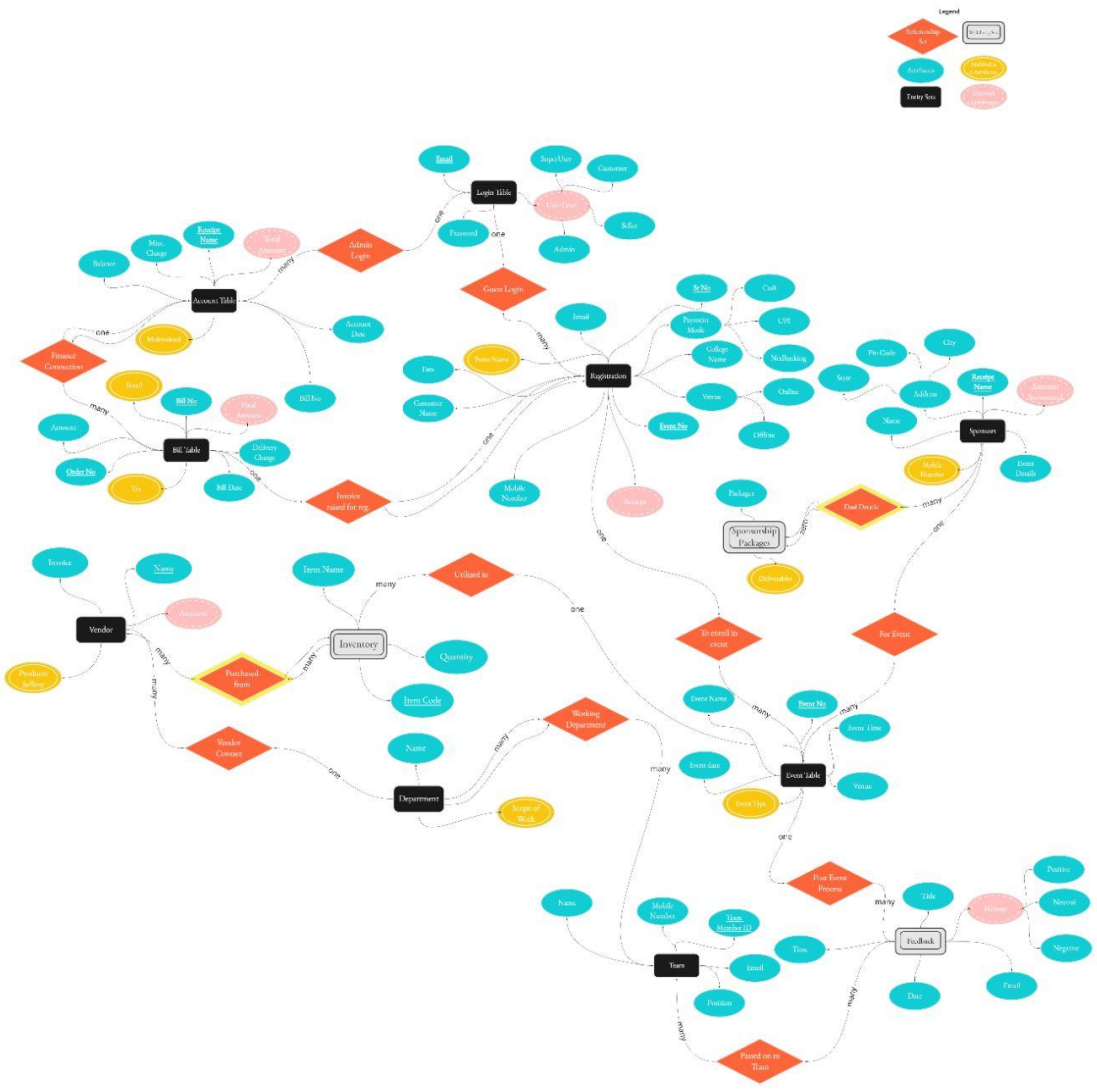

Mobile Number
Name
Email
Team


Misc. Charge
Balance
Account Table
Account Date
Bill No


Item Name
Item Code
Inventory
Quantity


Name
Department
Scope of Work


Packages
Sponsorship Packages
Deliverables



Please visit https://miro.com/app/board/o9J_lJQ9p5s=/ for full quality ER Diagram

Table Design

- List of all the tables in the database

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > show tables;
+------------------------------+
| Tables_in_dbmseventmanagement |
+------------------------------+
| account_table                |
| bill                         |
| department                   |
| event_table                  |
| feedback                     |
| inventory                    |
| login                        |
| registration                 |
| sponsors                     |
| sponsorship_package          |
| team                         |
| vendors                      |
+------------------------------+
12 rows in set (0.0052 sec)
```

- Describing the *account_table*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc account_table;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| balance      | double      | YES  |     | NULL    |       |
| misc_charges | double      | YES  |     | NULL    |       |
| receipt_name | varchar(50) | YES  |     | NULL    |       |
| account_date | date        | YES  |     | NULL    |       |
| bill_no      | int(11)     | YES  | MUL | NULL    |       |
| tot_amt      | double      | YES  |     | NULL    |       |
| paid_amt     | double      | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

- Describing the *bill*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc bill;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| bill_no    | int(11)     | NO   | PRI | NULL    | auto_increment |
| order_no   | int(11)     | NO   | PRI | NULL    |                |
| amount     | double      | YES  |     | NULL    |                |
| tax        | double      | YES  |     | NULL    |                |
| del_charge | double      | YES  |     | NULL    |                |
| final_amt  | double      | YES  |     | NULL    |                |
| bill_date  | date        | YES  |     | NULL    |                |
| email      | varchar(50) | YES  | MUL | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

- Describing the *department*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc department;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| department_name | varchar(50) | NO   | PRI | NULL    |       |
| vendor_relation | varchar(50) | YES  | MUL | NULL    |       |
| work_scope      | varchar(50) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

- Describing the *event_table*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc event_table;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| event_no   | int(11)     | NO   | PRI | NULL    | auto_increment |
| event_name | varchar(50) | YES  | MUL | NULL    |                |
| event_date | date        | YES  |     | NULL    |                |
| venue      | varchar(50) | YES  |     | NULL    |                |
| event_time | timestamp   | YES  |     | NULL    |                |
| event_type | varchar(50) | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

- Describing the *feedback*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc feedback;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| email   | varchar(50) | YES  | MUL | NULL    |       |
| title   | varchar(50) | YES  |     | NULL    |       |
| message | varchar(50) | YES  |     | NULL    |       |
| Date    | date        | YES  |     | NULL    |       |
| time    | time        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

- Describing the *inventory*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc inventory;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| item_code | varchar(50) | NO   | PRI | NULL    |       |
| item_name | varchar(50) | NO   | PRI | NULL    |       |
| quantity  | int(11)     | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

- Describing the *login*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc login;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| email | varchar(50) | NO   | PRI | NULL    |       |
| pass  | varchar(50) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

- Describing the *registration*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc registration;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| fees             | double      | YES  |     | NULL    |       |
| customer_name    | varchar(50) | YES  |     | NULL    |       |
| mob_name         | int(11)     | YES  |     | NULL    |       |
| email            | varchar(50) | YES  | MUL | NULL    |       |
| payment_mode     | varchar(50) | YES  |     | NULL    |       |
| sr_no            | varchar(50) | NO   | PRI | NULL    |       |
| college_name     | varchar(50) | YES  |     | NULL    |       |
| register_receipt | varchar(50) | YES  |     | NULL    |       |
| event_name       | varchar(50) | YES  |     | NULL    |       |
| event_no         | int(11)     | NO   | PRI | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
```

- Describing the *sponsors*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc sponsors;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| sponsors_name | varchar(50) | NO   | PRI | NULL    |       |
| address       | varchar(50) | YES  |     | NULL    |       |
| amount        | double      | YES  |     | NULL    |       |
| mob_num       | varchar(10) | NO   | PRI | NULL    |       |
| sponsor_type  | varchar(50) | YES  | MUL | NULL    |       |
| event_no      | int(11)     | YES  | MUL | NULL    |       |
| event_name    | varchar(50) | YES  | MUL | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

- Describing the *sponsorship_package*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc sponsorship_package;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| sponsor_type | varchar(50) | NO   | PRI | NULL    |       |
| deliverables | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

- Describing the *team*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc team;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| member_name     | varchar(50) | YES  |     | NULL    |       |
| mob_num         | int(11)     | YES  |     | NULL    |       |
| department_name | varchar(50) | YES  | MUL | NULL    |       |
| email           | varchar(50) | YES  |     | NULL    |       |
| team_member_id  | int(11)     | NO   | PRI | NULL    |       |
| position        | varchar(50) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

- Describing the *vendors*

```
MySQL  localhost:3306 ssl  dbmseventmanagement  SQL > desc vendors;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| products_taken | varchar(50) | YES  |     | NULL    |       |
| amount         | double      | YES  |     | NULL    |       |
| invoice_no     | varchar(50) | NO   | PRI | NULL    |       |
| vendor_name    | varchar(50) | YES  | MUL | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```

Queries to create tables(🔗)

```sql
create database dbmsEventManagement;

use dbmsEventManagement;

-- drop table feedback;
-- drop table inventory;
-- drop table team;
-- drop table department;
-- drop table vendors;
-- drop table sponsors;
-- drop table sponsorship_package;
-- drop table registration;
-- drop table account_table;
-- drop table bill;
-- drop table event_table;
-- drop table login;

create table login(email varchar(50) primary key, pass varchar(50));
create table event_table(
  event_no int auto_increment unique,
  primary key(event_no),
  event_name varchar(50),
  event_date date,
  venue varchar(50),
  event_time time,
  event_type varchar(50),
  index (event_name)
);


create table bill(
  bill_no int auto_increment unique,
  order_no int,
  amount double,
  tax double,
  del_charge double,
  final_amt double,
  bill_date date,
  email varchar(50),
  foreign key(email) references login(email),
  primary key(bill_no, order_no)
);


create table account_table(
  balance double,
  misc_charges double,
  receipt_name varchar(50),
  account_date date,
  bill_no int,
  foreign key(bill_no) references bill(bill_no),
  tot_amt double,
  paid_amt double
);
```

```sql
create table registration(
  fees double,
  customer_name varchar(50),
  mob_name varchar(50),
  email varchar(50),
  foreign key(email) references login(email),
  payment_mode varchar(50),
  sr_no varchar(50),
  college_name varchar(50),
  register_receipt varchar(50),
  event_name varchar(50),
  event_no int,
  foreign key(event_no) references event_table(event_no),
  primary key(event_no, sr_no)
);
create table sponsorship_package(
  sponsor_type varchar(50),
  deliverables varchar(50),
  primary key(sponsor_type)
);
create table sponsors(
  sponsors_name varchar(50),
  address varchar(50),
  amount double,
  mob_num int,
  sponsor_type varchar(50),
  foreign key(sponsor_type) references sponsorship_package(sponsor_type),
  event_no int,
  foreign key(event_no) references event_table(event_no),
  event_name varchar(50),
  foreign key(event_name) references event_table(event_name),
  primary key(sponsors_name, mob_num)
);
create table vendors(
  products_taken varchar(50),
  amount double,
  invoice_no varchar(50),
  vendor_name varchar(50),
  index (vendor_name),
  primary key(invoice_no)
);
create table department(
  department_name varchar(50),
  vendor_relation varchar(50),
  foreign key(vendor_relation) references vendors(vendor_name) on update cascade on delete cascade,
  work_scope varchar(50),
  primary key(department_name)
);
create table team (
  member_name varchar(50),
  mob_num int,
  department_name varchar(50),
  foreign key(department_name) references department (department_name) on update cascade on delete cascade,
  email varchar(50),
  team_member_id int,
  position varchar(50),
  primary key(team_member_id)
);
alter table team modify mob_num varchar(10);
create table inventory(
  item_code varchar (50),
  item_name varchar(50),
  quantity int,
  primary key(item_name, item_code)
);
create table feedback (
  email varchar(50),
  foreign key(email) references login(email) on update cascade on delete cascade,
  title varchar(50),
  message varchar(50),
  Date date,
  time time
);
alter table feedback modify message varchar(5000);
alter table feedback add Sentiment varchar(50);
alter table account_table add primary key(receipt_name);
alter table sponsors drop primary key;
alter table sponsors add primary key(sponsors_name,event_no);
```

Queries to insert data(🔗)

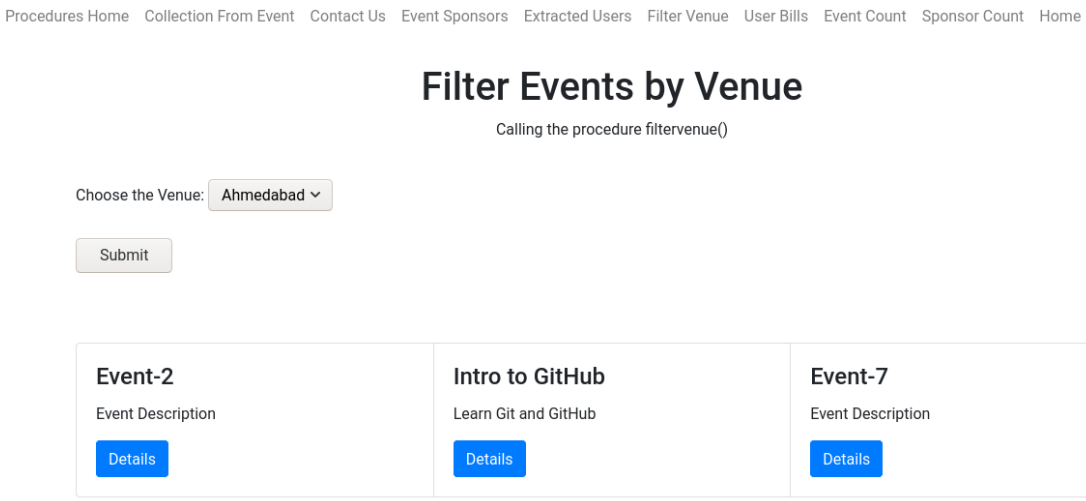
```sql
use dbmsEventManagement;

 insert into login values ('m@g.com','123'),('b@g.com','123');
 insert into login values ('parthmshah1302@gmail.com','123'),('malavdoshi312@gmail.com','123');

insert into event_table (event_no, event_name, event_date, venue, event_time, event_type) values (1, 'Kamba', '2020-07-10', 'Shebunino', '15:22:10', 'Stronghold');
insert into event_table (event_no, event_name, event_date, venue, event_time, event_type) values (2, 'Yodoo', '2021-01-24', 'Abilay', '3:17:11', 'Aerified');
insert into event_table (event_no, event_name, event_date, venue, event_time, event_type) values (3, 'Zoomcast', '2020-07-15', 'Isnos', '6:57:29', 'Temp');
insert into event_table (event_no, event_name, event_date, venue, event_time, event_type) values (4, 'Quaxo', '2020-08-07', 'Paratunka', '14:55:36', 'Latlux');
insert into event_table (event_no, event_name, event_date, venue, event_time, event_type) values (5, 'Midel', '2021-02-25', 'Morro do Chapéu', '22:25:36', 'Zaam-Dox');

insert into registration values
(500,'Parth' ,'91932419f','parthmshah1302@gmail.com','Cash','A103','AU','g','Zoomcast',3),
(500,'Pnot' ,'91542919','m@g.com','Cash','A104','AU','g','Quaxo',4),
(500,'Malav' ,'91919439','malavdoshi312@gmail.com','Cheque','A106','AU','g','Zoomcast',3),
(500,'Parth' ,'91932419','parthmshah1302@gmail.com','Cash','A105','AU','g','Yoodo',2),
(500,'Pusrshotam' ,'919191919','b@g.com','Cash','A106','AU','g','Kamba',1),
(500,'Malav' ,'91919439','malavdoshi312@gmail.com','Cheque','A102','AU','g','Yoodo',2);

insert into sponsorship_package values ('GOLD' ,'sample');
insert into sponsorship_package values ('PLATINIUM' ,'sample');
insert into sponsorship_package values ('SILVER' ,'sample');
insert into sponsors values ('Manikchand','Muh mai',10000,'9999999','GOLD',1,'Kamba'),
                        ('RMD','Yeh bhi Muh mai',10000,'9999999','SILVER',2,'Yodoo'),
                        ('Old monk','Liver mai',10000,'9999999','PLATINIUM',4,'Quaxo'),
                        ('Parth Industires','Bhrigu lake',10000,'9999999','GOLD',1,'Kamba');
```

Stored Procedures and Functions

1. Following procedure provides a filter of events according to the venue. We also call a function to show the number of upcoming events.

```sql
drop procedure if exists filtervenue ;
delimiter $$
create procedure filtervenue()
    begin
        declare c_end int default 0;
        declare r_cityname varchar(50);
        declare count_event int;
        declare c_event cursor for select venue from event_table;
        declare continue handler for not found set c_end=1;

        open c_event;
        getvenuename: LOOP
        fetch c_event into r_cityname;
       --  select event_count;
            if c_end=1 then
                leave getvenuename;
            end if;
            select event_count(r_cityname) into count_event ;
            select r_cityname as "VENUE:", count_event as "Total Events";
            select event_name as "Event",date_format(event_date,"%M %d %Y") as "Date",event_time as "Time" from event_table
            where venue=r_cityname;
        end loop;
        close c_event;
    end$$
delimiter ;
```

Filtervenue procedure on the frontend



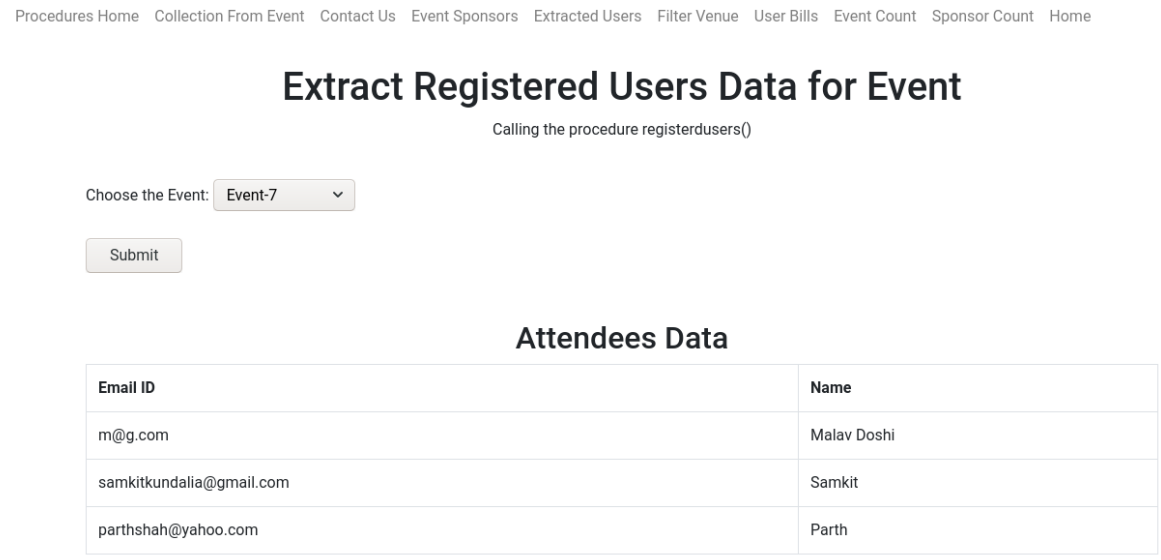
Code in backend used to call the procedure

```
@app.route('/filtervenue', methods=['GET','POST'])
def filtven():
    connection = mysql.connector.connect(host='localhost',database='dbmsEventManagement',user='admin',password='password')
    cursor = connection.cursor()
    cursor.callproc('filtervenue')
    for result in cursor.stored_results():
        print(result.fetchall())
    return("procedures/filtervenue.html",result)
```

2. Following procedure provides us the list of attendees of a particular event.

```
drop procedure if exists registeredusers;
delimiter $$
create procedure registeredusers()
    begin
        declare c_end int default 0;
        declare r_eventreg varchar(50);
        declare count_var int;
        declare c_registeredpeps cursor for select distinct event_name from registration order by event_name;
        declare continue handler for not found set c_end=1;
        open c_registeredpeps;
        getmailinglist: loop
            fetch c_registeredpeps into r_eventreg ;
                if c_end=1 then
                    leave getmailinglist;
                end if;
               select distinct r_eventreg as "Event Name";
                    select email,customer_name from registration where registration.event_name=r_eventreg ;
                    -- select count(email), fees from registration where registration.event_name = r_eventreg into count_var;
            end loop;
        close c_registeredpeps;
    end$$
delimiter ;
call registeredusers();
```

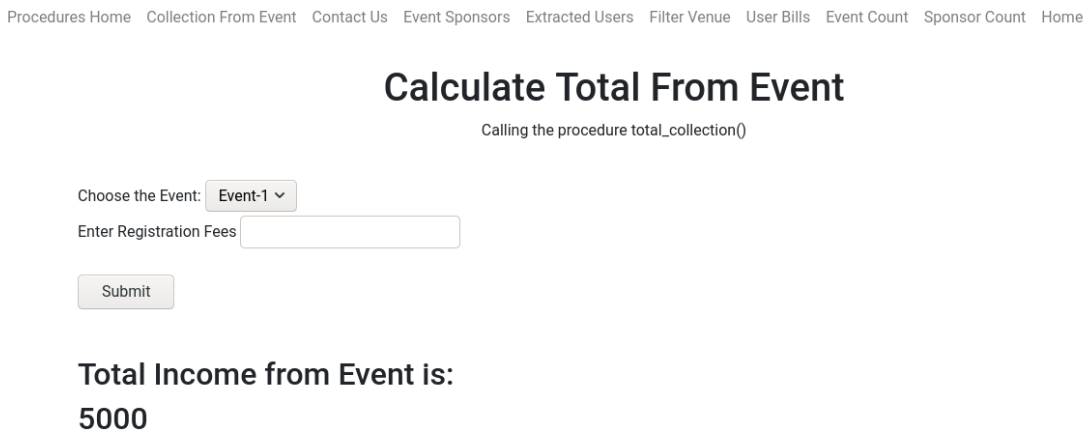
Output on the front-end when we call the procedure registeredUsers()



3. Following procedure provides us with the total collection of the event where we pass the event name and the fees in the entry fees in a particular event as parameters.

```
drop procedure if exists total_collection;
delimiter $$
create procedure total_collection(p_fees int,p_event_name varchar(20))
begin
    declare c_end int default 0;
    declare r_eventname varchar(20);
    declare r_count int;
    declare ans int;
    select count(r.customer_name) from registration r left join event_table e on e.event_no=r.event_no where e.event_name=p_event_name into r_count;
    set ans=r_count*p_fees;
    select r_eventname as "Event name";
    select ans as "Total Collection";
end $$
delimiter ;
call total_collection(500,'Zoomcast');
```

Output on the front-end when we call the procedure total_collection()



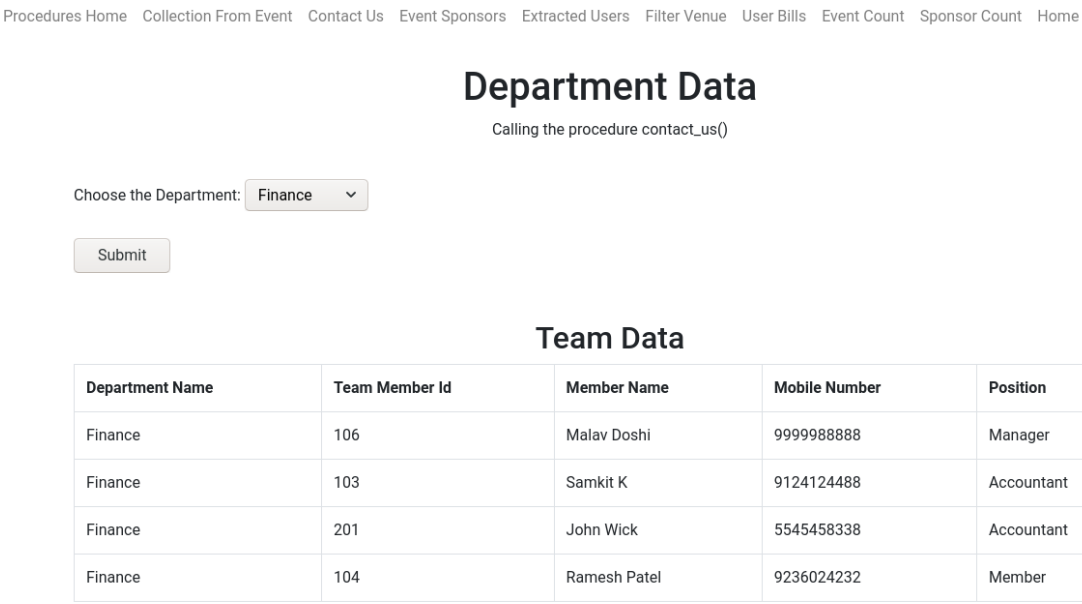
Code in backend used to call the procedure

```
@app.route('/calctotalfromevent', methods=['GET','POST'])
def calctotalfromevent():
    if request.method == 'POST':
            totalReq = request.form
            fees = totalReq['fees']
            event_name = totalReq['event_name']
            connection = mysql.connector.connect(host='localhost',database='dbmsEventManagement',user='admin',password='password')
            cursor = connection.cursor()
            cursor.callproc('totalcollection',(fees,event_name))
            for result in cursor.stored_results():
                print(result.fetchall())
    return render_template("procedures/calctotalfromevent.html",result)
```

4. Following procedure helps the user to get in touch with members of the event team in a particular department.

```
drop procedure if exists contact_us;
delimiter $$
create procedure contact_us (dept_name varchar(20))
begin
    select department_name,team_member_id,member_name, mob_num, position from team where department_name=dept_name;
end$$
delimiter ;
```

Output when we call the procedure contact_us() on the frontend



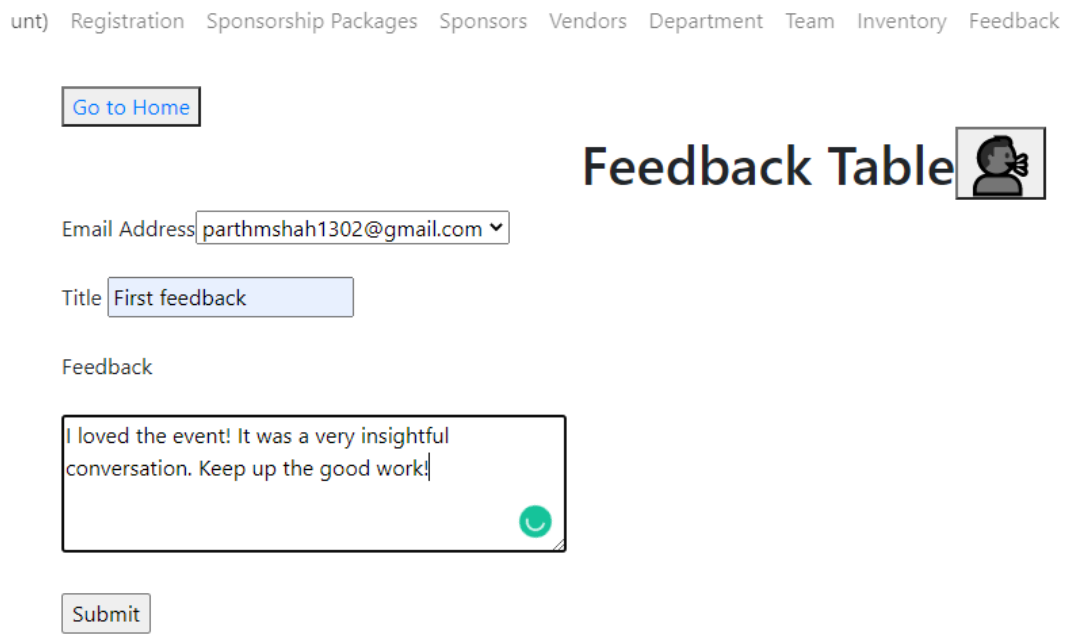
Code in backend used to call the procedure

```
@app.route('/contactus', methods=['GET','POST'])
def contactUs():
    if request.method == 'POST':
        contactForm = request.form
        dept_name = contactForm['dept_name']
        connection = mysql.connector.connect(host='localhost',database='dbmsEventManagement',user='admin',password='password')
        cursor = connection.cursor()
        cursor.callproc('contact_us')
        for result in cursor.stored_results():
            print(result.fetchall())
    return render_template("procedures/contactus.html",result)
```

5. Following procedure takes the feedback message as it's parameter. It makes an API call in real time through Python which parses in the message from MySQL and returns the sentiment of the message. This saves countless hours that is spent on reading and understanding the messages in detail.

```
drop procedure if exists event_feedback;
delimiter $$
create procedure event_feedback()
begin
    declare c_end int default 0;
    declare r_event_no varchar(50);
    declare r_email varchar(50);
    declare r_event_name varchar(50);
    declare c_eventfeedback cursor for select distinct event_no,email from registration order by event_no;
    declare continue handler for not found set c_end=1;
    open c_eventfeedback;
        getfeedback: loop
            fetch c_eventfeedback into r_event_no,r_email ;
                if c_end=1 then
                    leave getfeedback;
                end if;
            select event_name from event_table where event_no=r_event_no into r_event_name ;
            -- select r.email, f.title, f.message, f.Date, f.time, f.sentiment from registration r right join on feedback f on f.email=r.email where e.event_name=r_event_name;
            select distinct r_event_name as "Event Name",r.customer_name, f.title, f.message, f.Date, f.time, f.sentiment from  feedback f left join registration r on r.email=f.email where f.email=r_email and r_event_no=f.event_no;
        end loop;
    close c_eventfeedback;
end$$
delimiter ;
```

Input message, category 1:



Returned sentiment of joy:

t) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Email ID	Title	Message	Date	Time	Sentiment
parthmshah1302@gmail.com	First feedback	I loved the event! It was a very insightful conversation. Keep up the good work!	2021-04-20	2:54:17	joy

Input message, category 2:

unt) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Go to Home

Feedback Table 

Email Address malavdoshi312@gmail.com

Title Second feedback

Feedback

I was bored to death in this event. Please do not organise such events ever!

Submit

Returned sentiment of sadness:

Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Email ID	Title	Message	Date	Time	Sentiment
parthmshah1302@gmail.com	First feedback	I loved the event! It was a very insightful conversation. Keep up the good work!	2021-04-20	2:54:17	joy
malavdoshi312@gmail.com	Second feedback	I was bored to death in this event. Please do not organise such events ever!	2021-04-20	2:55:56	sadness

6. Following bill provides the user with the bill of registration of the event that one did.

```
drop procedure if exists customer_bill;
delimiter $$
create procedure customer_bill(p_email varchar(50),p_pass varchar(20))
begin
    declare c_end int default 0;
    declare r_bill varchar(50);
    declare r_email varchar(50);
    declare c_bill cursor for select distinct email from bill;
    declare continue handler for not found set c_end=1;
    open c_bill;
        getbill: loop
            fetch c_bill into r_bill ;
                if c_end=1 then
                    leave getbill;
                end if;
            select * from bill where email=r_bill and login.pass=p_pass;
        end loop;
    close c_bill;
end$$
delimiter ;
```

Output when we call the procedure customer_bill() on the frontend

Procedures Home Collection From Event Contact Us Event Sponsors Extracted Users Filter Venue User Bills Event Count Sponsor Count Home

List of Particular Customer Bills

Calling the procedure customer_bill()

Enter Email:

Enter Password:

Submit

List of Bills

Bill Number	Order Number	Amount	Tax	Delivery Charge	Final Amount	Bill Date
2	25	100	18	2	120	15-04-2021
4	27	50	18	2	70	15-04-2021

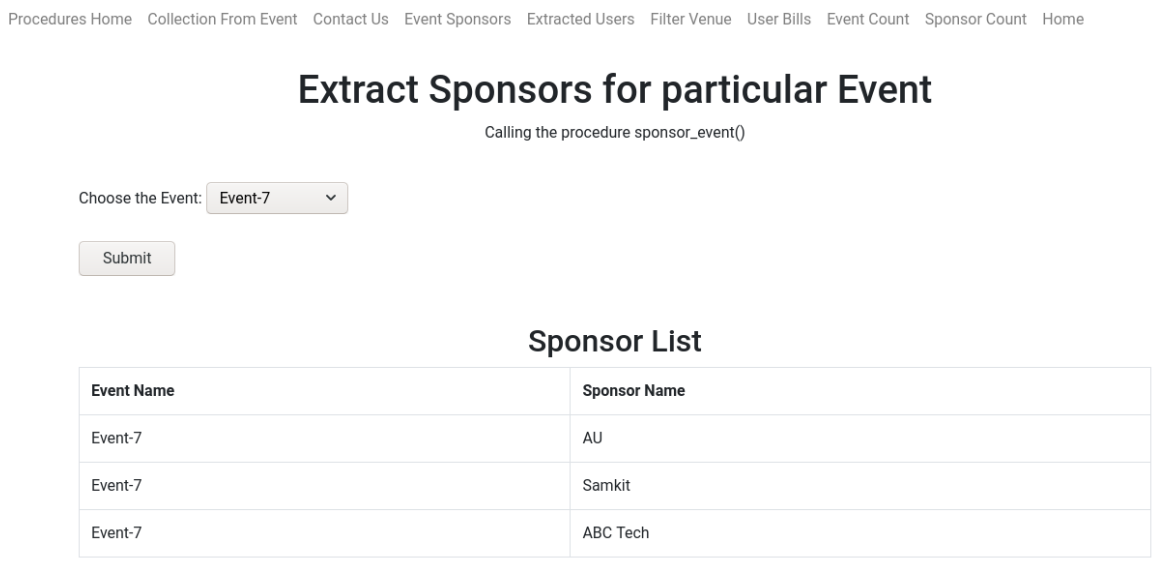
Code in backend used to call the procedure

```
@app.route('/userbill', methods=['GET','POST'])
def userbill():
    if request.method == 'POST':
            logindet = request.form
            email = logindet['email']
            password = logindet['password']
            connection = mysql.connector.connect(host='localhost',database='dbmsEventManagement',user='admin',password='password')
            cursor = connection.cursor()
            cursor.callproc('customer_bill',(email,password))
            for result in cursor.stored_results():
                print(result.fetchall())
    return render_template("procedures/userbill.html",result)
```

7. Following procedure displays all the sponsors of a particular event.

```
drop procedure if exists sponsor_event;
delimiter $$
create procedure sponsor_event()
    begin
        declare c_end int default 0;
        declare r_eventsp varchar(50);
        declare count_var int;
        declare c_sponsor cursor for select distinct event_no from event_table order by event_name;
        declare continue handler for not found set c_end=1;
        open c_sponsor;
        getsp: loop
            fetch c_sponsor into r_eventsp ;
                if c_end=1 then
                    leave getsp;
                end if;
                    select e.event_name from event_table e where e.event_no = r_eventsp;
                    select s.sponsors_name from event_table e left join sponsors s on e.event_no=s.event_no
                    where e.event_no=r_eventsp ;
            end loop;
        close c_sponsor;
    end$$
delimiter ;
call sponsor_event();
```

Output when we call sponsor_event() on the front-end



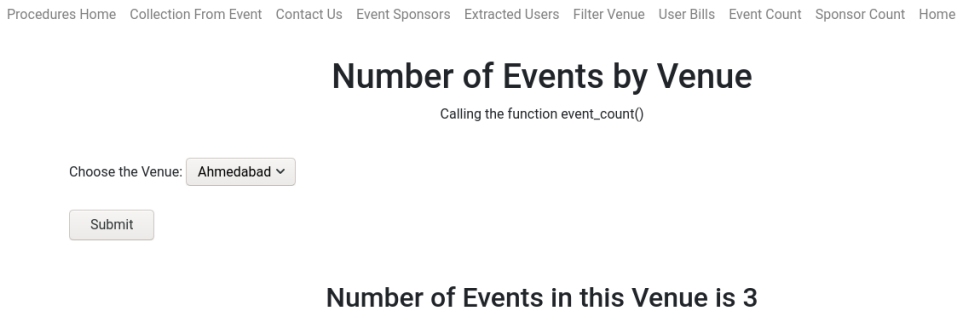
Code in backend used to call the procedure

```python
@app.route('/eventsponsors', methods=['GET','POST'])
def proceduresEventSponsors():
    connection = mysql.connector.connect(host='localhost',database='dbmsEventManagement',user='admin',password='password')
    cursor = connection.cursor()
    cursor.callproc('sponsor_event')
    for result in cursor.stored_results():
        print(result.fetchall())
    return render_template("procedures/eventsponsors.html",result)
```

8. Following function displays the number of events that are upcoming in a particular venue.

```sql
drop function if exists event_count ;
delimiter $$
 create function event_count(city_name varchar(20)) returns int deterministic
    begin
        declare totnum int default 0;
        select count(event_no) from event_table where event_table.venue=city_name into totnum;-- as "Upcoming Events";
       -- call filtervenue();
        return totnum;
    end$$
delimiter ;
```

Output when we call the function event_count



Code in backend used to call the procedure

```python
@app.route('/event_count', methods=['GET','POST'])
def Fevent_count():
    if request.method == 'POST':
            venueform = request.form
            city_name = venueform['event_name']
            cur = mysqlcon.connection.cursor()
            resultValue = cur.execute("SELECT event_count",(city_name))
            mysqlcon.connection.commit()
            cur.close()
    return render_template("procedures/event_count.html",resultValue)
```

9. Following function calculates the total amount associated with a particular bill.

```sql
drop function if exists totalamt;
delimiter $$
create function totalamt(b_no int) returns double deterministic
    begin
        declare c_end int default 0;
        declare r_totalamt varchar(50);
        declare total_amount double default 0.0;
        declare amt double default 0.0;
        declare t1 double default 0.0;
        declare del double default 0.0;
        declare c_bills cursor for select amount from bill where bill.bill_no=b_no;
        declare continue handler for not found set c_end=1;
        open c_bills;
        gettotamt: loop
            fetch c_bills into r_totalamt ;
                if c_end=1 then
                    leave gettotamt;
                end if;
            select amount from bill where bill.bill_no=b_no into amt  ;
            select tax from bill where bill.bill_no=b_no into t1 ;
            select del_charge from bill where bill.bill_no=b_no into del ;
            set total_amount=amt+(amt*t1/100)+del;
        end loop;
        return total_amount;
        end$$
    delimiter ;
```

ınt) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Go to Home

Bill Table

Order Number 123456

Amount 100

Tax Percentage 18

Delivery Charge 30

Date 08-04-2021

Email Address parthmshah1302@gmail.com Submit

Procedure called to populate the Final Amount field

ınt) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Bill Number	Order Number	Amount	Tax (%)	Delivery Charge	Final Amount	Bill Date	Email	Actions
4	104	23330.0	25.0	500.0	29662.5	February 12 2021	parthmshah1302@gmail.com	Delete
5	105	23000.0	8.0	2500.0	27340.0	June 19 2020	malavdoshi312@gmail.com	Delete
6	123456	100.0	18.0	30.0	148.0	April 08 2021	parthmshah1302@gmail.com	Delete

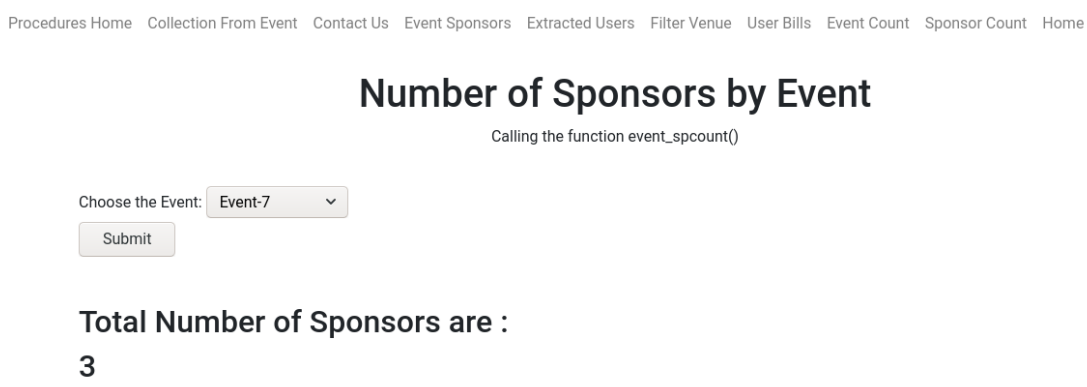
Python code that implements this to the Frontend

```python
@app.route('/bill', methods=['GET', 'POST'])
def bill():
    cur = mysqlcon.connection.cursor()
    cur.execute("SELECT email FROM login")
    emailTuple = cur.fetchall()
    mysqlcon.connection.commit()
    cur.close()
    if request.method == 'POST':
        billDetails = request.form
        order_no = billDetails['order_no']
        amount = billDetails['amount']
        tax = billDetails['tax']
        del_charge = billDetails['del_charge']
        bill_date = billDetails['bill_date']
        email = billDetails['email']

        cur = mysqlcon.connection.cursor()
        final_amt = cur.callproc('totalamt',[order_no])
        cur.execute("INSERT INTO bill( order_no, amount, tax, del_charge,final_amt,bill_date,email) VALUES(%s,%s,%s,%s,%s,%s,%s)",
                    (order_no, amount, tax, del_charge, final_amt, bill_date, email))
        mysqlcon.connection.commit()
        cur.close()

        return redirect('/billData')
    return render_template('bill.html', emailTuple=emailTuple)
```

10. Following function displays the total number of sponsors associated with a particular event.

```
drop function if exists event_spcount ;
delimiter $$
 create function event_spcount(r_event_name varchar(20)) returns int deterministic
    begin
        declare totnum int default 0;
        select count(sponsors_name) from sponsors s left join event_table e on e.event_no=s.event_no
        where e.event_name=r_event_name into totnum;
        return totnum;
    end$$
delimiter ;
```

Output when we call the function event_spcount() on the frontend



Code in the backend used to call the procedure

```
@app.route('/event_spcount', methods=['GET','POST'])
def Fevent_sponsorcount():
    if request.method == 'POST':
            sponsorform = request.form
            event_name = sponsorform['event_name']
            cur = mysqlcon.connection.cursor()
            resultValue = cur.execute("SELECT event_spcount",(event_name))
            mysqlcon.connection.commit()
            cur.close()
    return render_template("procedures/event_spcount.html",resultValue)
```

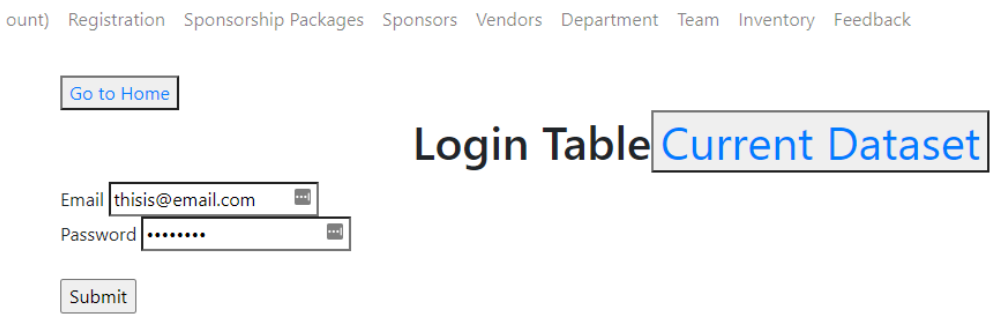
Triggers

1. Trigger to display error messages by checking whether the email exists in the database on inserting and updating .

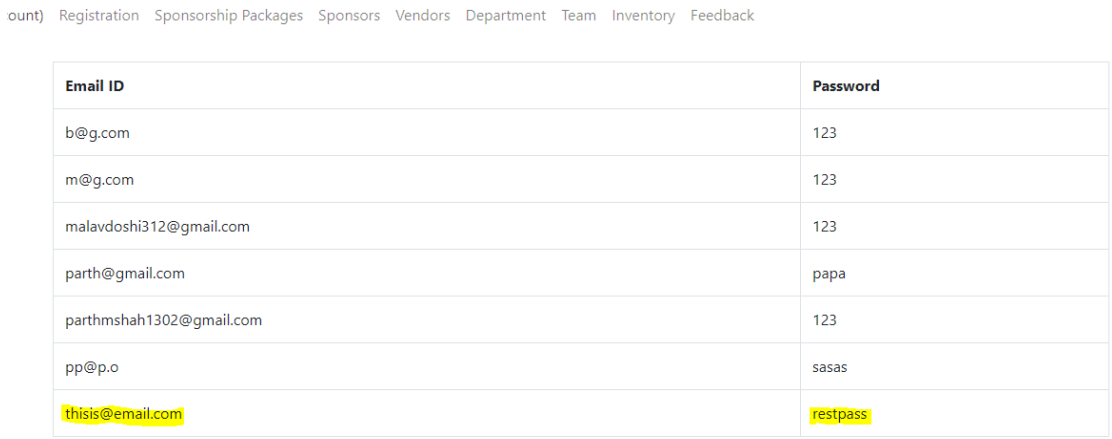
```
drop trigger if exists check_login;
delimiter $$
    create trigger check_login before insert on login for each row
        begin
        declare old_email varchar(50);
        select email from login where email=new.email into old_email;
        if old_email is not null then
            signal sqlstate '66666'
            set message_text="Email is invalid/duplicate. Please try again!";
        end if;
    end $$
delimiter ;
```

```
drop trigger if exists check_loginup;
delimiter $$
    create trigger check_loginup before update on login for each row
        begin
        declare old_email varchar(50);
        select email from login where email=new.email into old_email;
        if old_email is not null then
            signal sqlstate '66667'
            set message_text="Email is invalid/duplicate. Please try again!";
            end if;
    end $$
delimiter ;
```

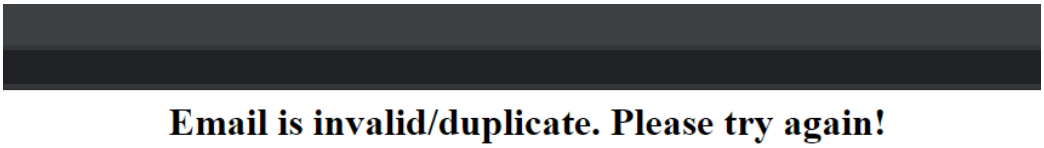
Entering a normal email address:



Data added successfully:



Trigger fired successfully:



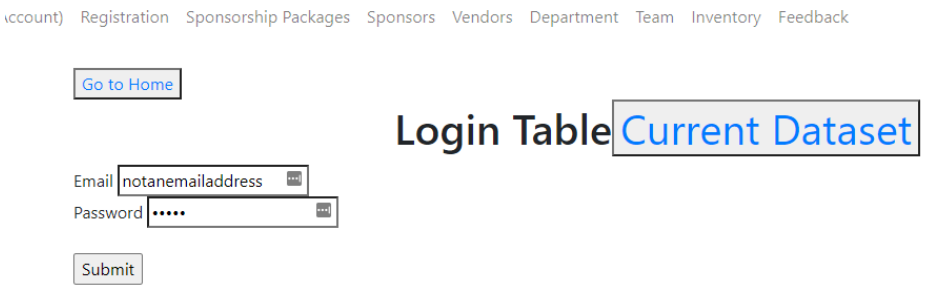
Connection with the Frontend

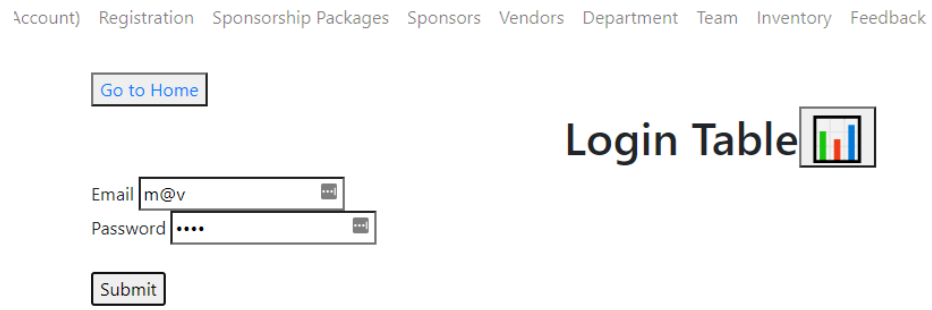
```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    try:
        if request.method == 'POST':
            userDetails = request.form
            email = userDetails['email']
            password = userDetails['password']

            cur = mysqlcon.connection.cursor()
            cur.execute("INSERT INTO login(email, pass) VALUES(%s, %s)",
                        (email, password))
            mysqlcon.connection.commit()
            cur.close()
            return redirect('/loginData')
        return render_template('login.html')
    except:
        return('<h1 style="text-align:center">Email is invalid/duplicate. Please try again!</h1?')

@app.route('/loginData')
def loginData():
    cur = mysqlcon.connection.cursor()
    resultValue = cur.execute("SELECT * FROM login")
    if resultValue > 0:
        userDetails = cur.fetchall()
        return render_template('loginData.html', userDetails=userDetails)
    else:
        return "Data does not exist, go to home page"
```

2. Trigger to check whether or not is the email valid on inserting and updating

```
drop trigger if exists validemail_u;
delimiter $$
    create trigger validemail_u before update on login for each row
    begin
        if new.email not like '%@%' then
            signal sqlstate value '93102'
                set message_text = 'The email you updated is invalid';
        elseif char_length(new.email) <5 then
            signal sqlstate value '92001'
            set message_text = 'The email you entered is invalid length';
        end if;
    end$$
delimiter ;
```

Adding an abnormal email to fire the trigger:





Trigger fired successfully:

Email is invalid/duplicate. Please try again!

Connection with the Frontend

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    try:
        if request.method == 'POST':
            userDetails = request.form
            email = userDetails['email']
            password = userDetails['password']

            cur = mysqlcon.connection.cursor()
            cur.execute("INSERT INTO login(email, pass) VALUES(%s, %s)",
                        (email, password))
            mysqlcon.connection.commit()
            cur.close()
            return redirect('/loginData')
        return render_template('login.html')
    except:
        return('<h1 style="text-align:center">Email is invalid/duplicate. Please try again!</h1?')

@app.route('/loginData')
def loginData():
    cur = mysqlcon.connection.cursor()
    resultValue = cur.execute("SELECT * FROM login")
    if resultValue > 0:
        userDetails = cur.fetchall()
        return render_template('loginData.html', userDetails=userDetails)
    else:
        return "Data does not exist, go to home page"
```

3. Trigger to delete records from child table registration , feedback and sponsors when deleted in the parent table event_table.

```
drop trigger if exists event_delete;
delimiter $$
    create trigger event_delete before delete on event_table for each row
        begin
            delete from registration where event_no=old.event_no;
            delete from sponsors where event_no=old.event_no;
        end $$
    delimiter ;
delete from event_table where event_name='Quaxo';
```

Following is the database before deleting from Event Table

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Event Number	Event Name	Event Date	Event Location	Event Time	Action
1	Kamba	July 10 2020	Shebunino	03:22 PM	Delete
2	Yodoo	January 24 2021	Abilay	03:17 AM	Delete
3	Zoomcast	July 15 2020	Isnos	06:57 AM	Delete
5	Midel	February 25 2021	Morro do Chapéu	10:25 PM	Delete

Following is the database before deleting from Sponsors.

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

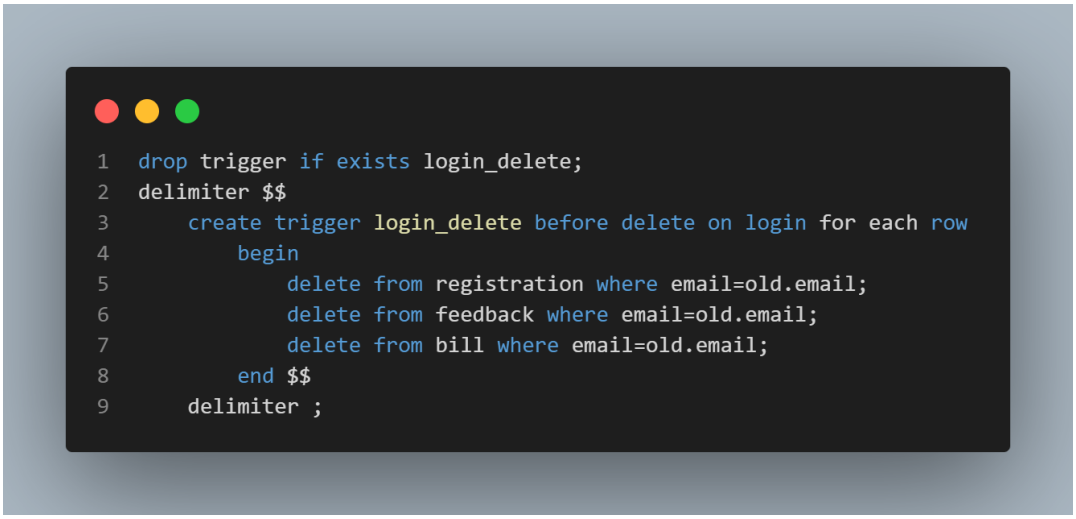
Sponsor's Name	Sponsor's Address	Amount (in ₹)	Mobile Number	Sponsor Type	Event No	Event Name	Actions
Amazon	Seattle	20000.0	9099121212	PLATINIUM	2	Midel	Delete
Apple	Cupertino,California	5000.0	9129129129	SILVER	3	Yodoo	Delete
Infosys	Pune, India	13000.0	9870434267	GOLD	2	Midel	Delete
Reliance	Navi Mumbai	10000.0	9870423567	PLATINIUM	1	Kamba	Delete

Following is the database before deleting from Registration.

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Fees	Customer Name	Mobile number	Email Address	Payment Mode	Serial Number	Serial Number	Registration Reciept	Event Name	Event Number
500.0	Pusrshotam	919191919	b@g.com	Cash	A106	AU	g	Kamba	1
500.0	Malav	91919439	malavdoshi312@gmail.com	Cheque	A102	AU	g	Yoodo	2
500.0	Parth	91932419	parthmshah1302@gmail.com	Cash	A105	AU	g	Yoodo	2
500.0	Parth	91932419f	parthmshah1302@gmail.com	Cash	A103	AU	g	Zoomcast	3
500.0	Malav	91919439	malavdoshi312@gmail.com	Cheque	A106	AU	g	Zoomcast	3

4. Trigger to delete records from child table bill, registration,feedback when deleted in the parent table login.



```
drop trigger if exists login_delete;
delimiter $$
    create trigger login_delete before delete on login for each row
        begin
            delete from registration where email=old.email;
            delete from feedback where email=old.email;
            delete from bill where email=old.email;
        end $$
    delimiter ;
```

Following is the data before any record is deleted from login

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Email ID	Password
b@g.com	123
m@g.com	123
m@v.c	passpass
malavdoshi312@gmail.com	123
parth@gmail.com	papa
parthmshah1302@gmail.com	123
pp@p.o	sasas
thisis@email.com	restpass

Following is the data from bill table

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Bill Number	Order Number	Amount	Tax (%)	Delivery Charge	Final Amount	Bill Date	Email	Actions
1	101	200.0	18.0	33.0	None	April 13 2021	malavdoshi312@gmail.com	Delete
2	102	6000.0	12.0	1200.0	7920.0	June 22 2019	m@g.com	Delete
3	103	12200.0	18.0	5000.0	19396.0	April 16 2021	m@v.c	Delete
4	104	23330.0	25.0	500.0	29662.5	February 12 2021	parthmshah1302@gmail.com	Delete
5	105	23000.0	8.0	2500.0	27340.0	June 19 2020	malavdoshi312@gmail.com	Delete

Following is registration table before delete

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Fees	Customer Name	Mobile number	Email Address	Payment Mode	Serial Number	Serial Number	Registration Reciept	Event Name	Event Number
500.0	Pusrshotam	919191919	b@g.com	Cash	A106	AU	g	Kamba	1
500.0	Malav	91919439	malavdoshi312@gmail.com	Cheque	A102	AU	g	Yoodo	2
500.0	Parth	91932419	parthmshah1302@gmail.com	Cash	A105	AU	g	Yoodo	2
500.0	Parth	91932419f	parthmshah1302@gmail.com	Cash	A103	AU	g	Zoomcast	3
500.0	Malav	91919439	malavdoshi312@gmail.com	Cheque	A106	AU	g	Zoomcast	3

Following is the feedback table before delete

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Email ID	Title	Message	Date	Time	Sentiment
parthmshah1302@gmail.com	First feedback	I loved the event! It was a very insightful conversation. Keep up the good work!	2021-04-20	2:54:17	joy
malavdoshi312@gmail.com	Second feedback	I was bored to death in this event. Please do not organise such events ever!	2021-04-20	2:55:56	sadness
b@g.com	Great	The event was great	2021-04-20	3:20:45	joy
b@g.com	Great	The event was great	2021-04-20	3:20:47	joy

5. Trigger to delete records from the child table team, vendors when deleted in the parent table department.

```
drop trigger if exists sponsor_package_delete;
delimiter $$
    create trigger sponsor_package_delete before delete on sponsorship_package for each row
        begin
            delete from sponsors where sponsor_type=old.sponsor_type;
        end $$
    delimiter ;
```

Following is the department table before deleting

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Department name	Vendor Details	Scope of Work	Actions
HR	Sachil Pillai	Management	Delete
PR	Bansari Kim	Public Relations	Delete
Tech	Sachil Pillai	Technical	Delete

Following is the table vendors before delete

Name	Products	Amount	Invoice	Actions
Sumit Kant	Decorative	10000.0	A101	Delete
Sachil Pillai	Tech	20000.0	A102	Delete
Bansari Kim	Jewels	200000.0	A103	Delete

Following is the table team before delete

Home Login Events Bills Finance(Account) Registration Sponsorship Packages Sponsors Vendors Department Team Inventory Feedback

Member Name	Department Name	Mobile number	Email Address	Position	Member ID
Samkit	9099123908	HR	samkit@google.com	Lead	1
Parth	9012390867	Tech	parth@hotmail.com	Manager	2
Malav	9123498700	HR	malav@google.com	Head	3
Kashvi	9099120987	PR	kashvi@yahoo.com	Member	4
Aneri	9102368457	PR	aneri@gmail.com	Lead	5

6. Trigger to update records from child table registration , feedback and sponsors when deleted in the parent table event_table.

```
drop trigger if exists event_update;
delimiter $$
    create trigger event_update after update on event_table for each row
        begin
            update registration set event_no=new.event_no where event_no=old.event_no;
            update sponsors set event_no=new.event_no where event_no=old.event_no;
            update feedback set event_no=new.event_no where event_no=old.event_no;
        end $$
delimiter ;
```

7. Trigger to delete records from child table registration , feedback and sponsors when deleted in the parent table event_table.

```
drop trigger if exists login_update;
delimiter $$
    create trigger login_update after update on login for each row
        begin
            update registration set email=new.email where email=old.email;
            update feedback set email=new.email where email=old.email;
            update bill set email=new.email where email=old.email;
        end $$
delimiter ;

```