

# **Project Report on**

## **INDIAN CULTURAL FESTIVAL**

By

Shanmukh Varma Rudraraju

Subject: Database Management Systems (CIS 556)



UNIVERSITY OF  
MICHIGAN

# Cultural Festival Database Management System

## Abstract:

Every year, Mumbai hosts the cultural event known as Kala Ghoda. This seven-day multi-cultural festival, which takes place in the Fort neighborhood, is delightful. You can purchase one-of-a-kind and handcrafted items from all around India at numerous stalls. Additionally, there are other ongoing events in a variety of genres, including dance, music, film, and heritage walks. Such a large festival requires sponsorship, advertising, and other support.

Therefore, there is a lot of information and data that requires to be kept. Traditional file systems are unable to handle the data because it is so large, interconnected, and challenging to manage. Tasks like insertion, deletion and updation of data occupy a lot of time in file systems but a database management system makes it far easier.

The database has been built to supersede any issues with the file systems that may be present. It has been designed to make the fewest errors possible and to issue warnings when incomplete or inaccurate data is entered. As a result, it is a user-friendly system that can result in an efficient, secure, and quick management.

In this project, I have created a database with the main tables for Entry, Events, Performers, Sponsors, Managers, Heritage Walk and Advertisements. For instance, the Performers table has a column for event id to link the performers with the category of event they're performing on, and the Advertisements table has a column for sponsor id linking it to the sponsor whose advertisement it is. Some of these are interdependent (one's primary key is another's foreign key).

## Introduction:

The organizer must put a lot of thought into preparing a big event like Kala Ghoda. It is necessary to invite guests, contact sponsors, issue advertisements, and keep track of all the events that must take place. As a result, a lot of crucial information about many different topics needs to be preserved. It must be categorized in a way that makes it simple for the administrators and organizers to access. As a result, it is required to group all of the data into groups with related attributes. This is done by splitting the data into tables and these tables are available for easy access.

Major tables pertaining to Entry, Events, Performers, Sponsors, Managers, Heritage Walk, and Advertisements are included in this system. The form allows the front-end user to sign up for an event, and their information will be saved in the database so that the event managers and organizing team can access it and use it to make the appropriate arrangements. All events,

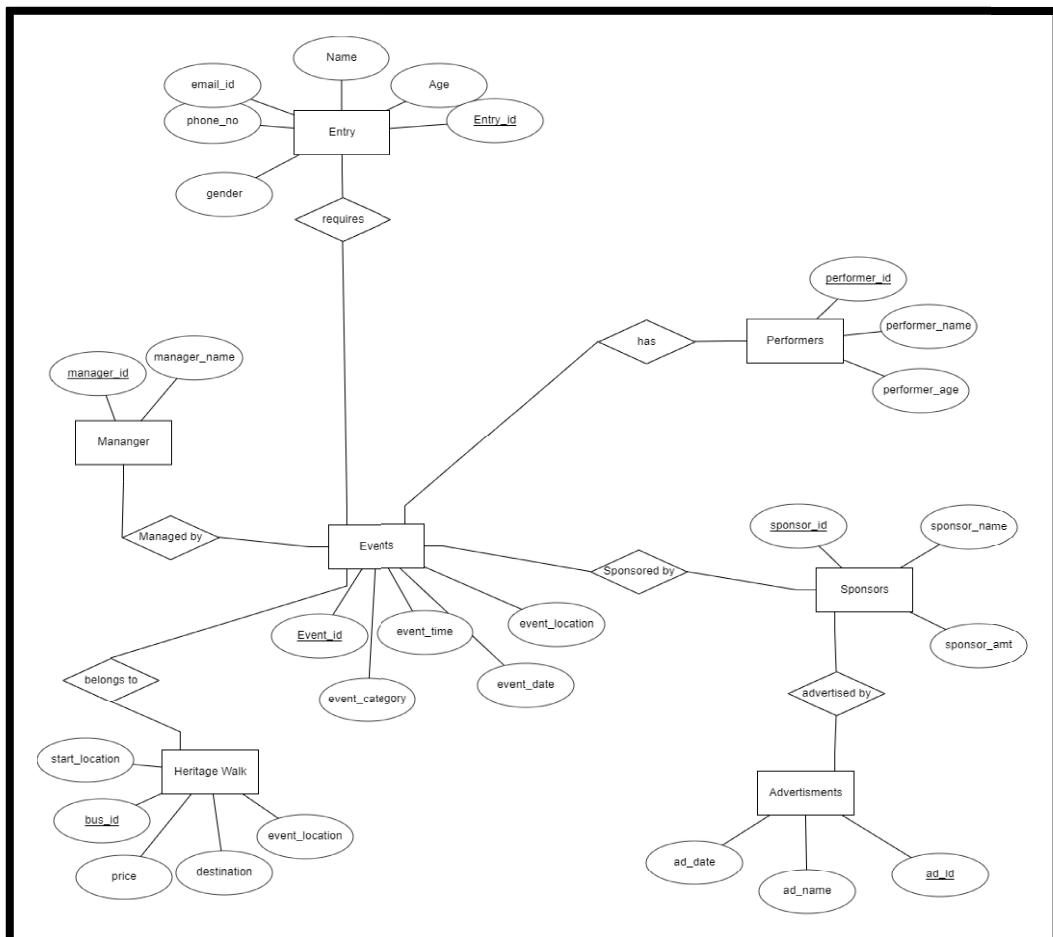
categories, curators, and sponsors have also had their information published. The festival's mission statement and its numerous components are both described in length.

Additionally, a registration form is offered so that users can register for an event and retain their personal information. There are also contact information, a map, and information on the festivities from the previous year.

## Problem Statement:

For the Kala Ghoda event, which deals with a lot of information and details and requires an effective administration system, to build a database they can use this system to help them with a variety of issues, like keeping tabs on the number of registrations, providing details about their venue, events, the many attractions they offer, and much more. The management of the festival as a whole is made simpler with the use of a database system, which also makes it simpler to display, handle, update, insert, and remove data items.

## 2. ER Diagram:



### 3. Reduction of ER model to Relational Model:

Advertisements(ad\_id,sponsor\_id,ad\_name,ad\_date)

Entry (Entry\_ID, Name, Age, Gender, Phone\_Number, Email\_ID, Event\_ID)

Events(event\_id, event\_category, event\_location, event\_date, event\_time)

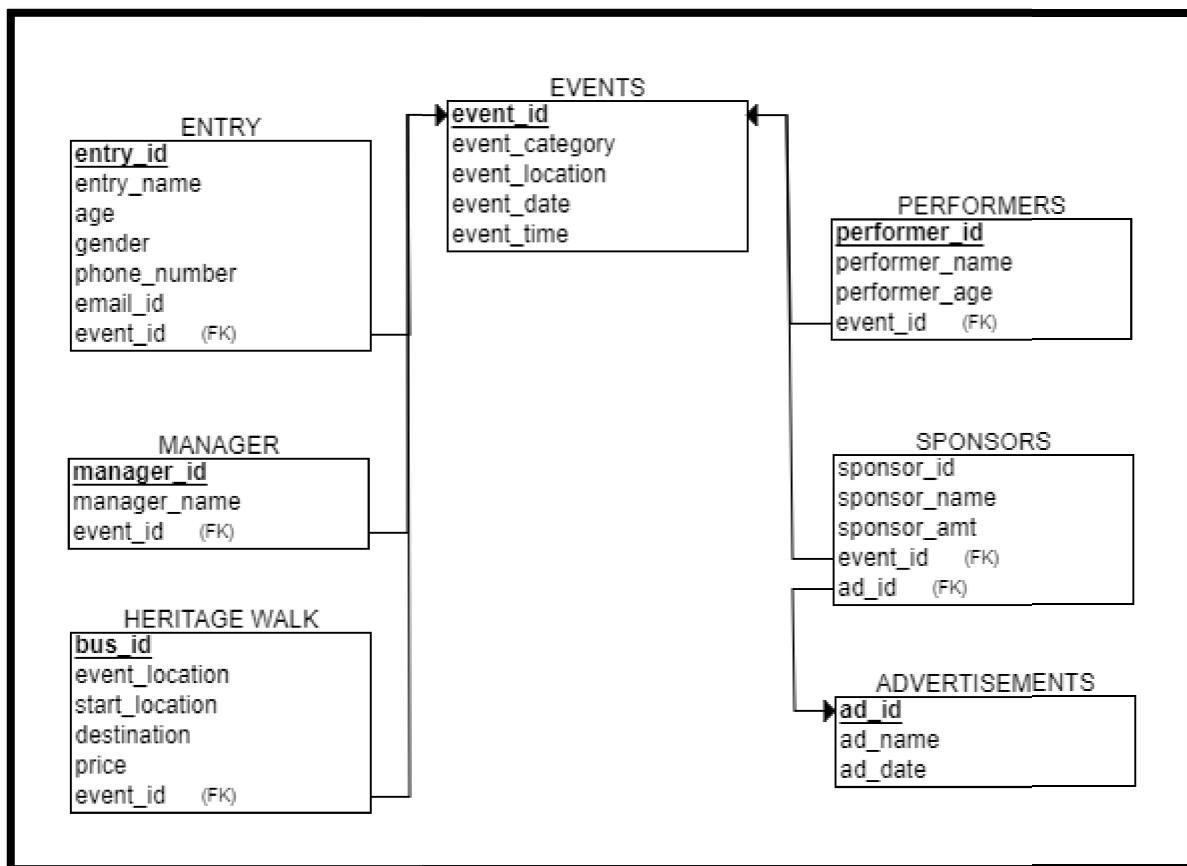
Heritage Walk( event\_location ,start location , destination, price, bus\_id)

Manager (manager\_id, event\_id, manager\_name)

Performers (performer\_id,event\_id, performer\_name, performer\_age)

Sponsors(sponsor\_id, Sponsor\_name, sponsor\_amt, event\_id)

### ER Schema Diagram:



**Constraints:** The constraints with respect to the tables in the database are

1) Advertisements

Not Null- All the attributes are not null  
Primary Key - ad\_id  
Foreign Key - sponsor\_id  
Auto Increments - ad\_id

2) Entry

Not Null- Entry\_ID, Name, Phone\_Number, Event\_ID  
Primary Key - Entry\_ID  
Foreign Key - Event\_ID  
Auto Increment- Entry\_ID

3) Events

Not Null - All the attributes are not null  
Primary Key - Event\_ID

4) Heritage Walk

Not Null- All the attributes are not null

5) Managers

Not Null- All attributes are not null  
Primary Key - manager\_id  
Foreign Key - event\_id

6) Performers

Not Null- All attributes are not null  
Primary Key - performer\_id  
Foreign key - event\_id  
Auto Increment - performer\_id

7) Sponsors

Not Null - All attributes are not null  
Primary Key - sponsor\_id  
Foreign Key - event\_id  
Auto Increment- sponsor\_id

## 4. Normalization of relational model:

Normalization is a database design technique that organizes tables in a manner that reduces redundancy and dependency of data. Using relationships, normalization breaks up huge tables into smaller ones. Normalization aims to remove redundant (useless) data and guarantee logical data storage. Additionally, it is employed to get rid of unwanted traits including Insertion, Update, and Deletion Anomalies.

The tables that are created in this management system are already normalized and cannot be normalized further as they follow the rules of the three major types of normal forms:

- 1) 1NF - A relation is in 1NF if it contains an atomic value.
- 2) 2NF - A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
- 3) 3NF - A relation will be in 3NF if it is in 2NF and no transition dependency exists.

**1NF:** The below table cells has a unique and atomic value.

Example:

Entry						
Entry_ID	Name	Age	Gender	Phone_Number	Email_ID	Event_ID

**2NF:** The below tables in which each column has a single column primary key.

Example:

Entry						
Entry_ID	Name	Age	Gender	Phone_Number	Email_ID	Event_ID

Advertisements		
ad_id	ad_name	ad_date

## Performers

performer_id	event_id	performer_name	performer_age
--------------	----------	----------------	---------------

## Manager

manager_id	manager_name	event_id
------------	--------------	----------

**3NF:** There are no transitive functional dependencies.

## 5. Database Queries & Results:

1. Write query to find Average age of the performer?

**Query:**

```
SELECT AVG(performer_age) as Avg_age_performer  
FROM performers;
```

**Result:**

The screenshot shows a database query results interface. At the top, a green bar indicates "Showing rows 0 - 0 (1 total, Query took 0.0023 seconds)". Below this is a toolbar with options like Profiling, Edit inline, Explain SQL, Create PHP code, Refresh, Show all, Number of rows (set to 25), and Filter rows. A search bar says "Search this table". The main area displays a single row with the heading "Avg\_age\_performer" and the value "25.3500". Below this is another toolbar with the same options as above. At the bottom, there is a "Console" section with a prompt "Press Ctrl+Enter to execute query" and a history entry: "> SELECT AVG(performer\_age) as Avg\_age\_performer FROM performers;".

2. Find the event\_location, start\_location and destination of heritage walk consisting prices less than the average price of heritage walk?

### Query:

```
SELECT event_location,start_location,destination,price  
FROM heritagewalk  
WHERE price < (SELECT AVG(price)  
                FROM heritagewalk)  
ORDER BY price ASC;
```

### Result:

The screenshot shows a database interface with a results table and a console window.

**Results Table:**

event_location	start_location	destination	price
Nariman House	Enterence 4	Enterence 3	20
Apollo Bunder	Enterence 3	Enterence 1	40
Asiatic	Enterence 1	Enterence 3	40
Asiatic	Enterence 1	Enterence 3	40
Blue Synagogue	Enterence 1	Enterence 4	40
Asiatic	Enterence 3	Enterence 2	50
Bhaucha Dhakka	Enterence 1	Enterence 4	50
Apollo Bunder	Enterence 2	Enterence 4	50

**Console:**

```
Press Ctrl+Enter to execute query  
> SELECT event_location,start_location,destination,price  
      FROM heritagewalk  
      WHERE price < (SELECT AVG(price)  
                      FROM heritagewalk)  
      ORDER BY price ASC;
```

3. Find the Manager Name and performer's who are associated with each other?

**Query:**

```
SELECT performer_name,manager_name  
FROM performers as p, manager as m  
WHERE p.event_id = m.event_id  
ORDER BY performer_id;
```

**Result:**

performer_name	manager_name
Rutvi	Rahul
Lavanya	Harsh
Kareena	Rina
Mihika	Sneha
Sharanya	Nishee
Nidhi	Pragya
Harsh	Meethi
Tanishk	Ojas
Tanvi	Kareena
Rahul	Dipanita
Suresh	Rahul
Deboparna	Ayush
Divya	Mihika
Anshul	Asma
Uday	Smriti
Sakshi	Ayushi
Abhigyan	Disha
Nandini	Pearl
Garima	Eshika
Ibhan	Dhwani

Console

```
Press Ctrl+Enter to execute query  
> SELECT performer_name,manager_name  
  FROM performers as p, manager as m  
 WHERE p.event_id = m.event_id  
 ORDER BY performer_id
```

4. Find the performer's who are associated with dance event?

**Query:**

```
SELECT performer_name  
FROM performers as p, events as e  
WHERE p.event_id = e.event_id AND e.event_category = "Dance";
```

**Result:**

The screenshot shows a MySQL query results interface. At the top, a green bar indicates "Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)". Below this is the SQL query: "SELECT performer\_name FROM performers as p, events as e WHERE p.event\_id = e.event\_id AND e.event\_category = "Dance";". There are buttons for Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh. Below the query are two tabs: "performer\_name" and "Console". The "performer\_name" tab displays the results: Mihika, Garima, and Nandini. Each result has a "Delete" link next to it. There are also "Show all", "Number of rows: 25", and "Filter rows: Search this table" buttons. The "Console" tab contains the query text and a note: "Press Ctrl+Enter to execute query".

```
Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)  
  
SELECT performer_name FROM performers as p, events as e WHERE p.event_id = e.event_id AND e.event_category = "Dance";  
  
Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]  
  
 Show all | Number of rows: 25 Filter rows: Search this table  
  
Extra options  
  
performer_name  
Mihika  
Garima  
Nandini  
  
 Show all | Number of rows: 25 Filter rows: Search this table  
  
Console  
Press Ctrl+Enter to execute query  
> SELECT performer_name  
    FROM performers as p, events as e  
    WHERE p.event_id = e.event_id AND e.event_category = "Dance";
```

5. Filter the entries that are younger among all entries?

**Query:**

```
SELECT *
FROM entry
WHERE Age in (SELECT min(Age)
               FROM entry)
ORDER BY Entry_ID;
```

**Result:**

The screenshot shows a MySQL query results interface. At the top, a green bar indicates "Showing rows 0 - 4 (5 total, Query took 0.0010 seconds.) [Entry\_ID: 28... - 34...]" and displays the executed SQL query. Below this is a toolbar with options like Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh. Further down are buttons for Show all, Number of rows (set to 25), and Filter rows, along with a search bar. A "Extra options" button is also present. The main area displays a table with columns: Entry\_ID, Name, Age, Gender, Phone\_Number, Email\_ID, and Event\_ID. The data is as follows:

Entry_ID	Name	Age	Gender	Phone_Number	Email_ID	Event_ID
28	Mateen	22	Male	1334123172	mateen.y@gmail.com	4
29	Sufiyan	22	Male	2147483647	sufiyan@gmail.com	3
32	Yash	22	Male	2147483647	yash@gmail.com	7
33	Monish	22	Male	1330966427	monish.l@gmail.com	6
34	Sonu	22	Male	1167150459	sonur@gmail.com	8

At the bottom, a "Console" section shows the query again with the instruction "Press Ctrl+Enter to execute query".

```
Press Ctrl+Enter to execute query
> SELECT *
  FROM entry
 WHERE Age in (SELECT min(Age)
                 FROM entry)
 ORDER BY Entry_ID;
```

6. Find the entries who attended “Rutvi’s” performance?

**Query:**

```
SELECT Name,Email_ID
```

```
from entry as e, performers as p
```

```
WHERE e.event_ID = p.event_id AND p.performer_name = "Rutvi";
```

**Result:**

The screenshot shows a MySQL query results interface. At the top, a green bar indicates "Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)". Below this is the SQL query:

```
SELECT Name,Email_ID from entry as e, performers as p WHERE e.event_ID = p.event_id AND p.performer_name = "Rutvi";
```

Below the query are several navigation and search controls:

- Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
- Show all | Number of rows: 25 | Filter rows: Search this table
- Extra options

The main result area displays a single row of data:

Name	Email_ID
Sufiyan	sufiyan@gmail.com

Below the result table are more search and filter controls:

```
□ Show all | Number of rows: 25 | Filter rows: Search this table
```

At the bottom, there are "Query results operations" and a "Console" section:

Press Ctrl+Enter to execute query

```
> SELECT Name,Email_ID  
from entry as e, performers as p  
WHERE e.event_ID = p.event_id and p.performer_name = "Rutvi";
```

7. Find the heritagewalk which are near to the events and display the related information of the events?

### Query:

```
SELECT event_id,event_category,event_date,event_time  
FROM events as e, heritagewalk as h  
WHERE e.event_location = h.event_location;
```

### Result:

The screenshot shows a MySQL query results interface. At the top, a green bar indicates "Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)". Below this is the SQL query: "SELECT event\_id, event\_category, event\_date, event\_time FROM events as e, heritagewalk as h WHERE e.event\_location = h.event\_location;". There are buttons for Profiling, Edit inline, Explain SQL, Create PHP code, and Refresh. Below the query is a table with four rows of data. The table has columns: event\_id, event\_category, event\_date, and event\_time. The data is as follows:

event_id	event_category	event_date	event_time
2	Music	2020-04-05 00:00:00	from 4:00pm onwards
2	Music	2020-04-05 00:00:00	from 4:00pm onwards
2	Music	2020-04-05 00:00:00	from 4:00pm onwards
2	Music	2020-04-05 00:00:00	from 4:00pm onwards

At the bottom, there is a "Console" section with the query text again and a note: "Press Ctrl+Enter to execute query".

```
SELECT event_id, event_category, event_date, event_time  
FROM events as e, heritagewalk as h  
WHERE e.event_location = h.event_location;
```

8. Find the sponsor\_id and ad\_date associated with Twitter?

**Query:**

```
SELECT sponsor_id,ad_date  
from advertisements  
WHERE ad_name = "Twitter";
```

**Result:**

The screenshot shows a MySQL query results interface. At the top, a green bar indicates "Showing rows 0 - 4 (5 total, Query took 0.0054 seconds.)". Below this, the SQL query is displayed: "SELECT sponsor\_id,ad\_date from advertisements WHERE ad\_name = "Twitter";". There are buttons for Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh. Below the query, there are controls for Show all (unchecked), Number of rows (set to 25), Filter rows, and a search bar. A "Extra options" button is also present. The main area displays a table with two columns: sponsor\_id and ad\_date. The data is as follows:

sponsor_id	ad_date
3	25th March
5	28th March
6	8th March
11	12th March
10	21st March

At the bottom, a "Console" section shows the executed query: "SELECT sponsor\_id,ad\_date from advertisements WHERE ad\_name = "Twitter";". It also includes a note: "Press Ctrl+Enter to execute query".

9. Find the Entries with age greater than average age among all entries ?

**Query:**

```
SELECT Entry_id,Name,Gender,Age
```

```
FROM entry
```

```
WHERE Age > (SELECT Avg(Age)
```

```
FROM entry)
```

```
ORDER BY age DESC;
```

**Result:**

The screenshot shows a MySQL query results interface. At the top, a green bar indicates "Showing rows 0 - 3 (4 total, Query took 0.0006 seconds.)". Below this is the SQL query:

```
SELECT Entry_id,Name,Gender,Age FROM entry WHERE Age > (SELECT Avg(Age) FROM entry) ORDER BY age DESC;
```

Below the query are several buttons: Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh. There are also filters for Show all (unchecked), Number of rows (set to 25), and Filter rows (Search this table). An "Extra options" button is also present.

The main area displays a table with the following data:

Entry_id	Name	Gender	Age
30	Ali	Male	35
31	Hemanth	Male	27
26	Sudheer	Male	25
35	Swetha	Female	25

At the bottom, there is a "Console" section with a text input field containing the same SQL query. A note says "Press Ctrl+Enter to execute query".

10. Find the performer's details who are managed by "Asma" ?

**Query:**

```
SELECT performer_id,performer_name,performer_age  
FROM performers as p, manager as m  
WHERE m.manager_name = "Asma" AND p.event_id = m.event_id;
```

**Result:**

The screenshot shows a MySQL query results interface. At the top, a green status bar indicates "Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)". Below it is a toolbar with options like Profiling, Edit inline, Explain SQL, Create PHP code, and Refresh. The main area displays a table with three columns: performer\_id, performer\_name, and performer\_age. A single row is shown: performer\_id 14, performer\_name Anshul, and performer\_age 32. There are also "Show all" and "Number of rows" dropdowns, a search bar, and an "Extra options" button. At the bottom, there is a console window with the same SQL query entered.

performer_id	performer_name	performer_age
14	Anshul	32

```
SELECT performer_id,performer_name,performer_age  
FROM performers as p, manager as m  
WHERE m.manager_name = "Asma" AND p.event_id = m.event_id;
```