Project Report on

INDIAN CULTURAL FESTIVAL

By

Shanmukh Varma Rudraraju

Subject: Database Management Systems (CIS 556)

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,

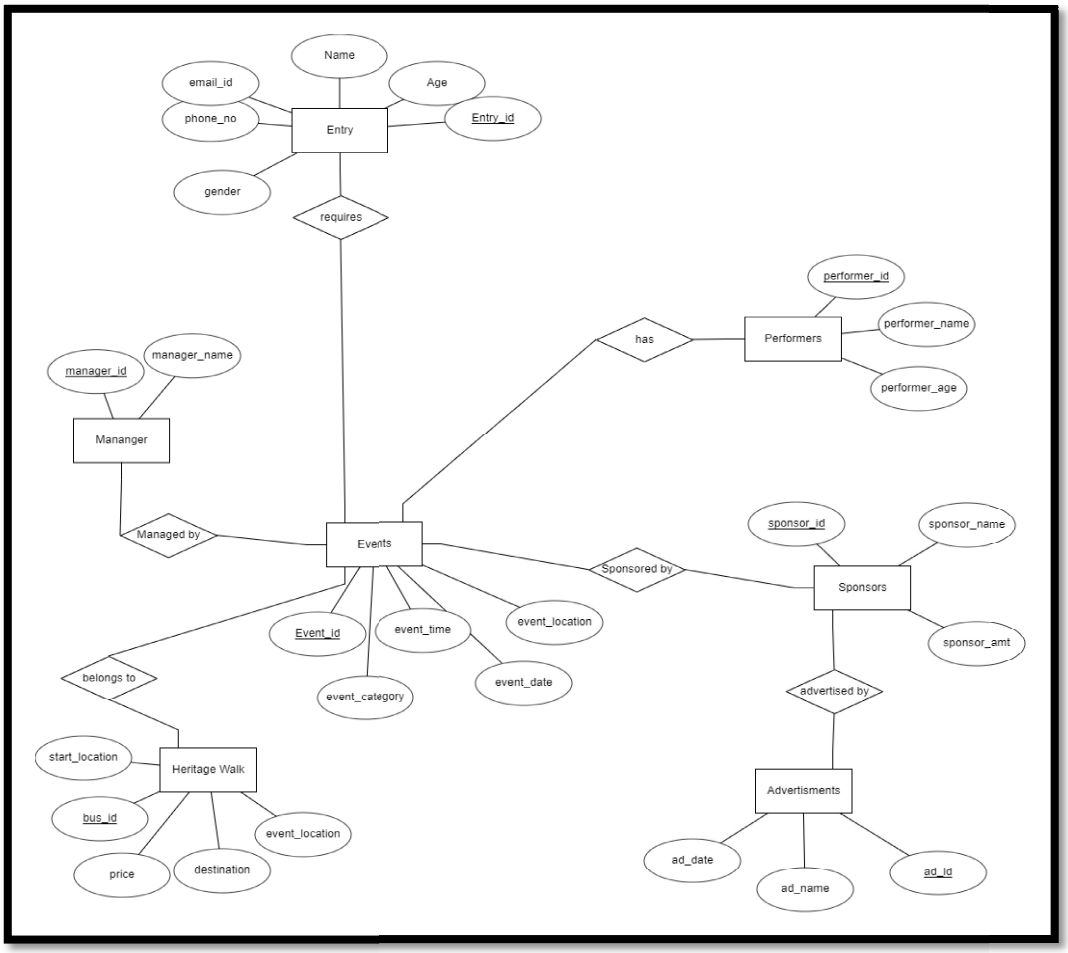
categories, curators, and sponsors have also had their information published. The festival's mission statement and its numerous components are both described in length.

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)

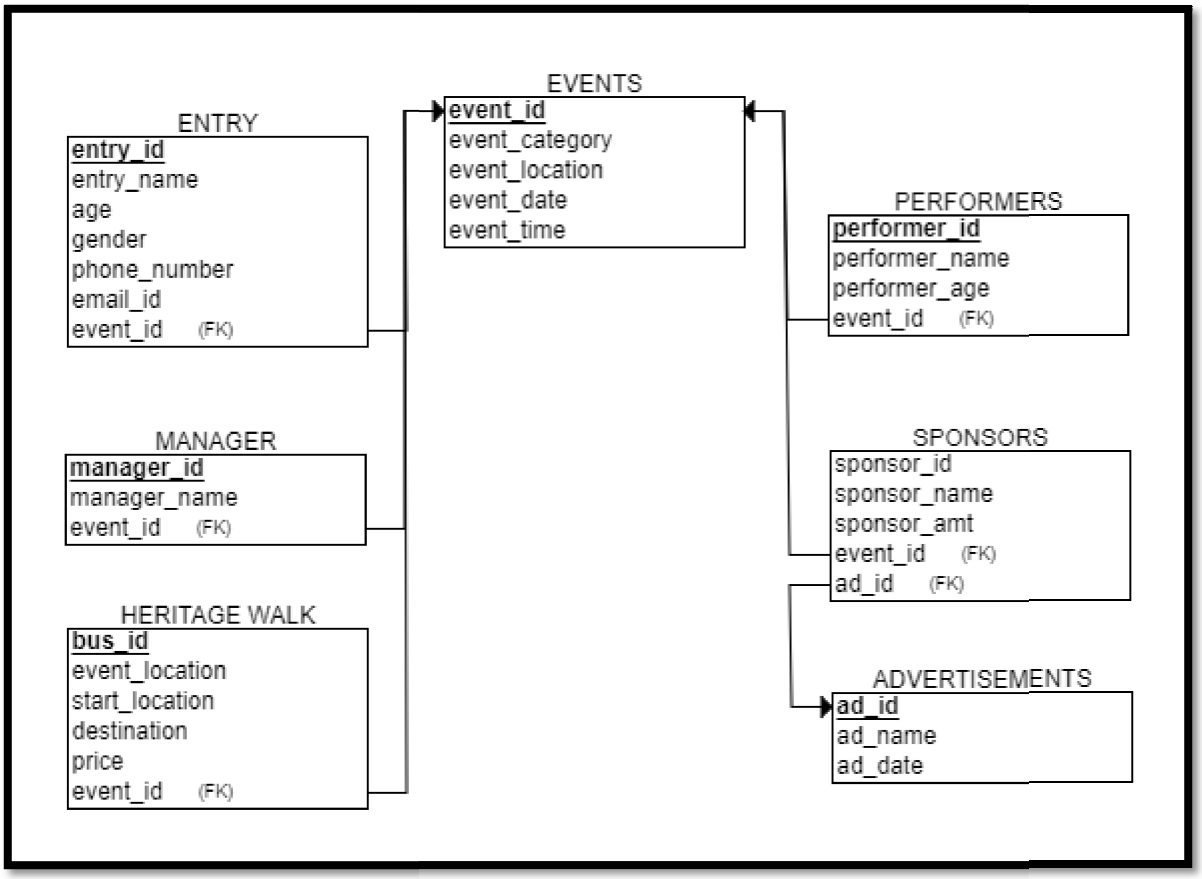
Heritage Walk(event_location ,start location , destination, price, bus_id)

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.

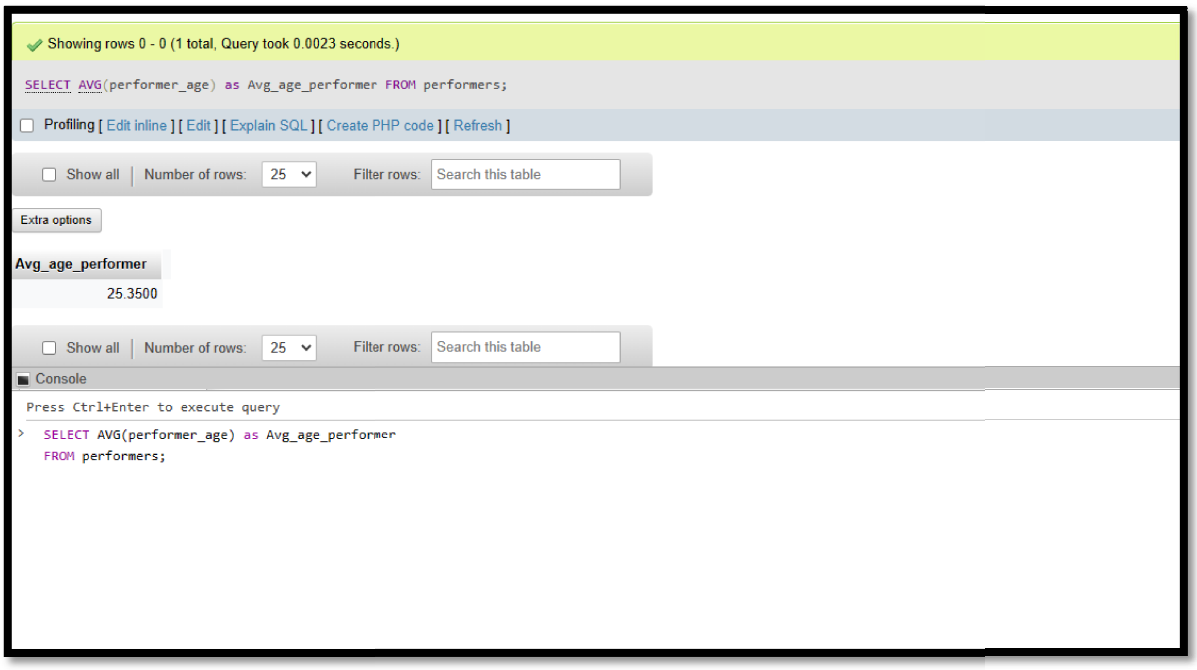
5. Database Queries & Results:

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

Query:

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

Result:

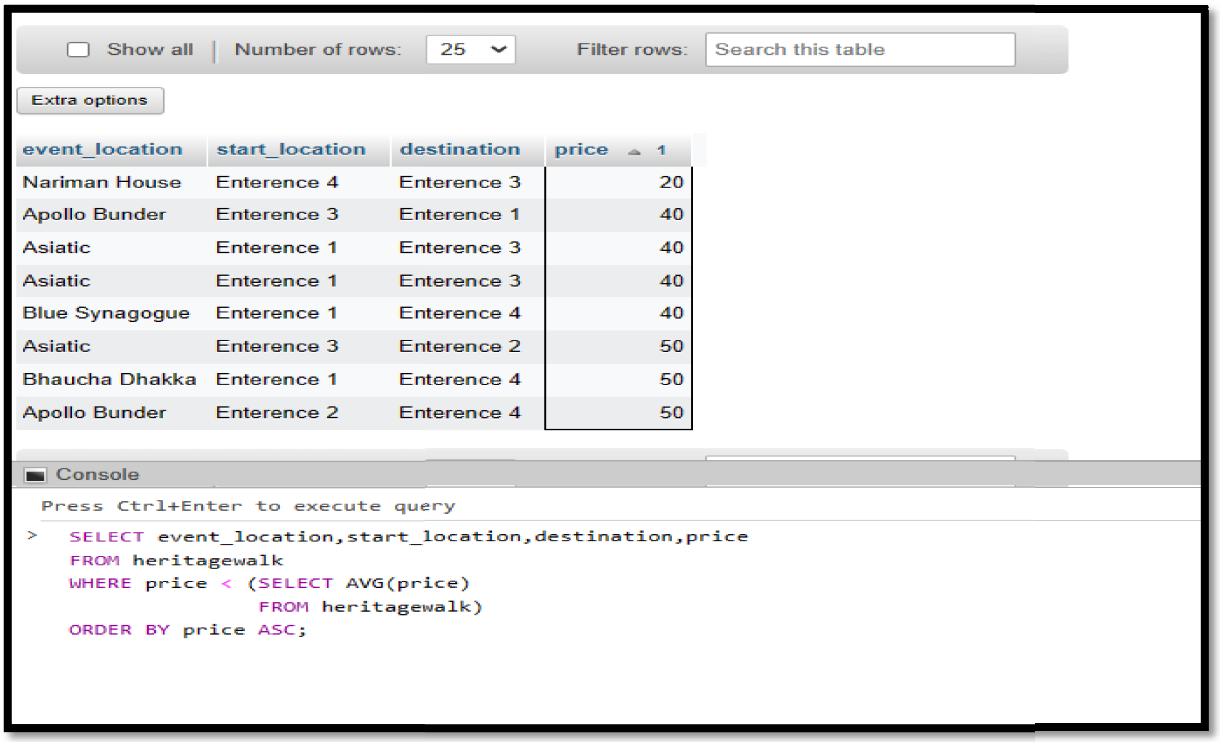


2. Find the event_location, start_location and destination of heritage walk consisting prices less than the average price of heritage walk?

Query:

```
SELECT event_location,start_location,destination,price

FROM heritagewalk

WHERE price < (SELECT AVG(price)

                FROM heritagewalk)

ORDER BY price ASC;
```

Result:



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	Ojasi
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";
```

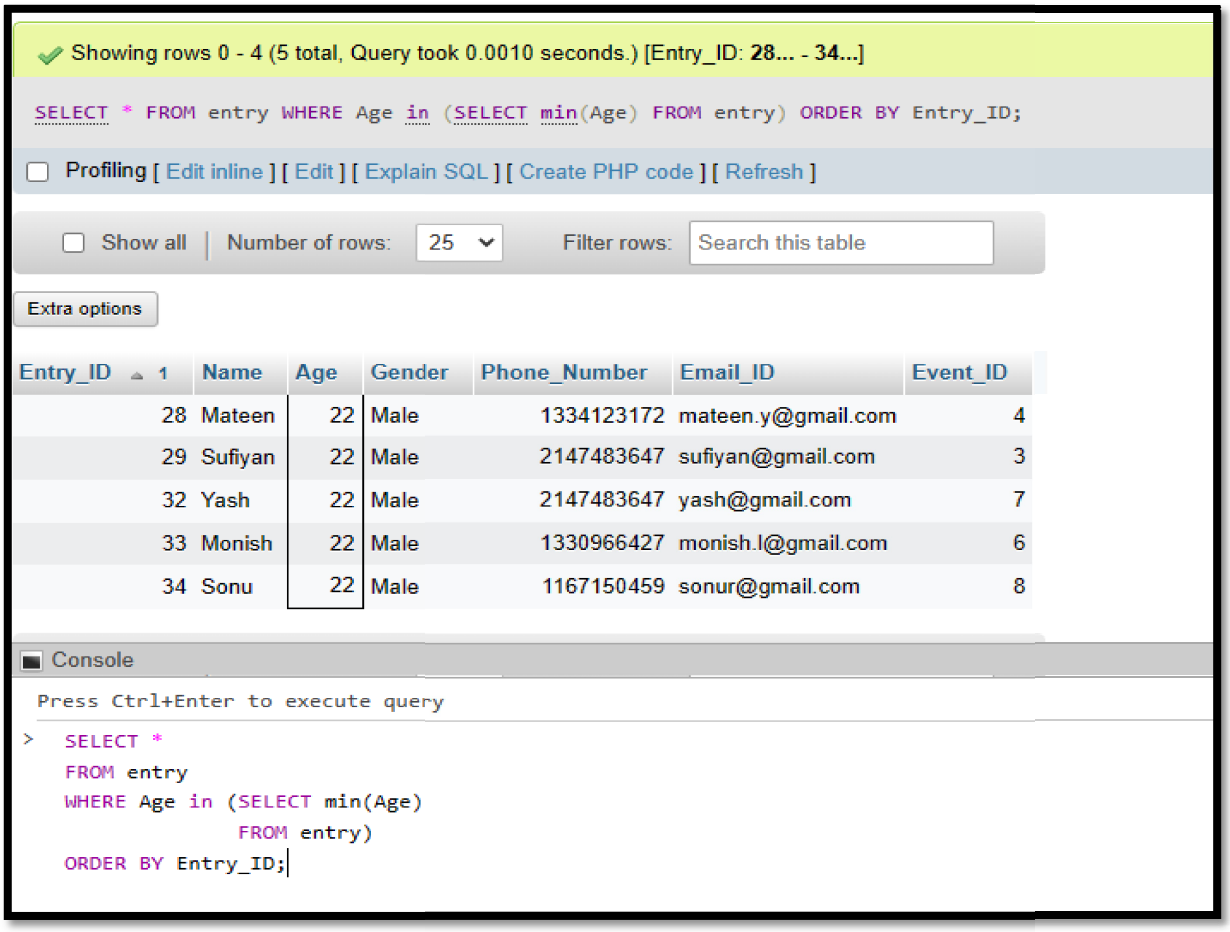
Result:

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:



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";
```

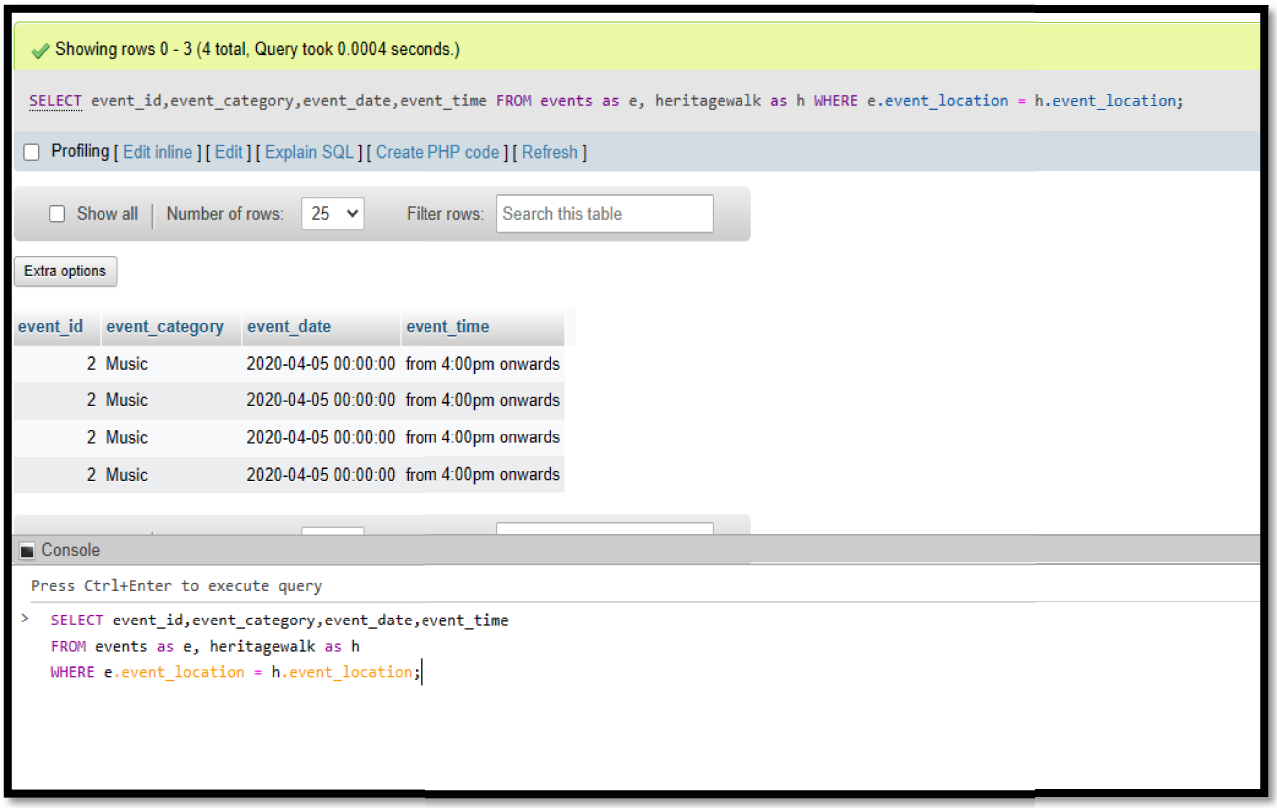
Result:

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:

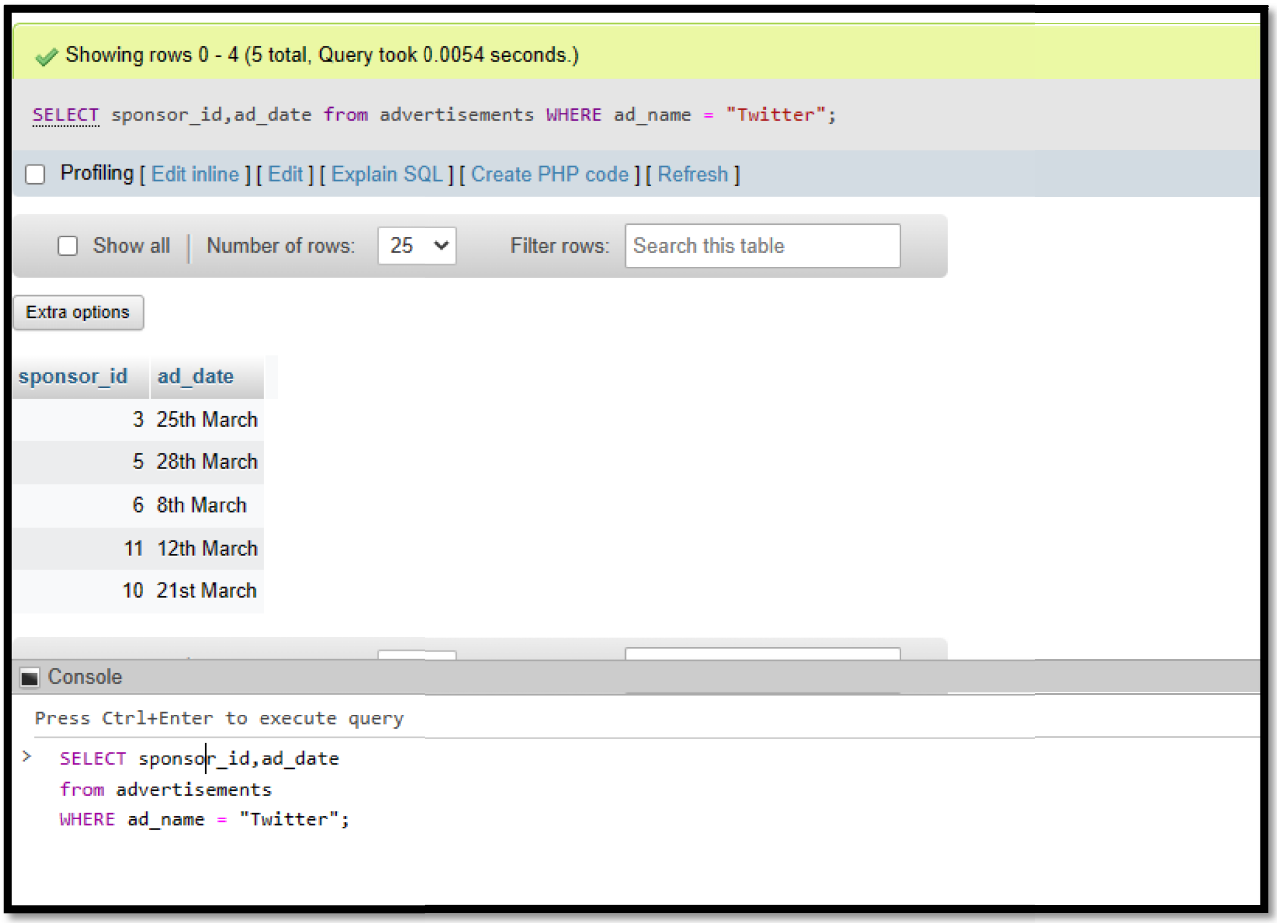


8. Find the sponser_id and ad_date associated with Twitter?

Query:

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

Result:

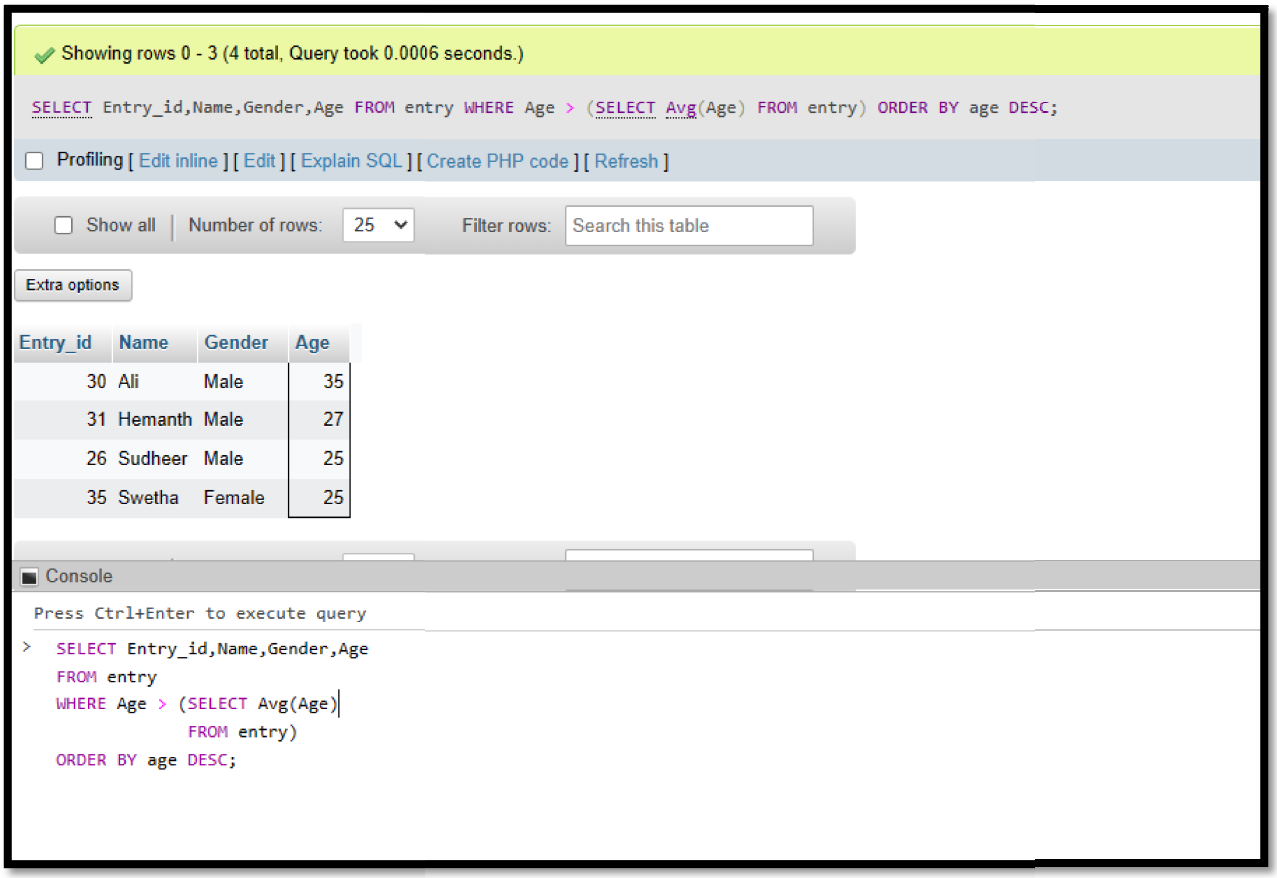


9. Find the Entries with age greater than average age among all entries ?

Query:

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

Result:



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: