**DBMS - MINI PROJECT**

***CONCERT MANAGEMENT SYSTEM***

Submitted By:

**Name**: Amogh N Rao

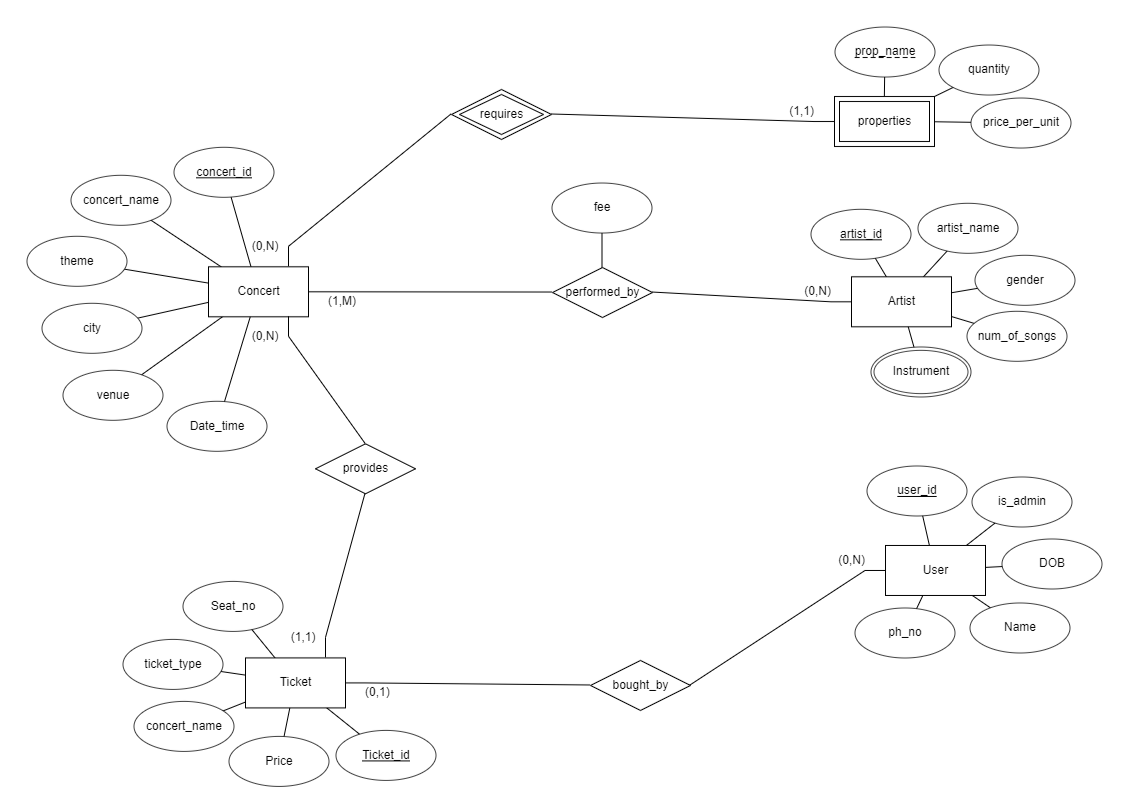
**SRN**: PES1UG20CS625

V Semester Section K

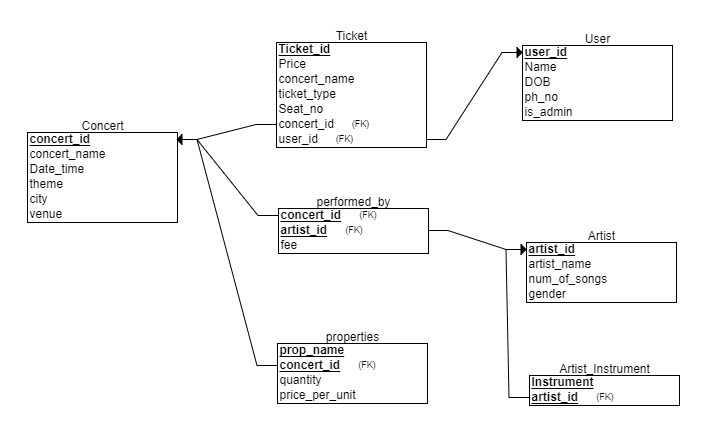
**ABSTRACT**

Concert management system provides an efficient way to store data about concerts. Entities recognised in this project are: Concert, Artist, User, Ticket and Properties. Properties is a weak entity and is identified by concert entity. Each concert is performed by a number artists. Concerts distribute tickets which can be bought by users. A Web Application can be built with this database which allows admins to add information regarding concerts and user to search and buy tickets.

**ER Diagram**



**Relational Schema**



**DDL statements - Building the database**

CREATE TABLE Concert

(

concert\_id INT NOT NULL,

concert\_name VARCHAR(50) NOT NULL,

Date\_time Timestamp NOT NULL,

theme VARCHAR(50) NOT NULL,

city VARCHAR(50) NOT NULL,

venue VARCHAR(50) NOT NULL,

PRIMARY KEY (concert\_id)

);

CREATE TABLE Artist

(

artist\_id INT NOT NULL,

artist\_name VARCHAR(20) NOT NULL,

num\_of\_songs INT NOT NULL,

gender enum('Male','Female','Other') NOT NULL,

PRIMARY KEY (artist\_id)

);

CREATE TABLE User

(

user\_id INT NOT NULL,

Name VARCHAR(20) NOT NULL,

DOB DATE NOT NULL,

ph\_no VARCHAR(10) NOT NULL,

is\_admin enum('yes','no') NOT NULL,

PRIMARY KEY (user\_id)

);

CREATE TABLE Ticket

(

Ticket\_id INT NOT NULL,

Seat\_no INT NOT NULL,

Price Float NOT NULL,

concert\_name VARCHAR(20) NOT NULL,

ticket\_type enum('gold','platinum','vip') NOT NULL,

concert\_id INT NOT NULL,

user\_id INT,

PRIMARY KEY (Ticket\_id,concert\_id),

FOREIGN KEY (concert\_id) REFERENCES Concert(concert\_id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES User(user\_id) ON DELETE CASCADE

);

CREATE TABLE properties

(

concert\_id INT NOT NULL,

prop\_name VARCHAR(50) NOT NULL,

quantity INT NOT NULL,

price\_per\_unit FLOAT NOT NULL,

PRIMARY KEY (concert\_id,prop\_name),

FOREIGN KEY (concert\_id) REFERENCES Concert(concert\_id) ON DELETE CASCADE

);

CREATE TABLE performed\_by

(

concert\_id INT NOT NULL,

artist\_id INT NOT NULL,

fee FLOAT NOT NULL,

PRIMARY KEY (concert\_id, artist\_id),

FOREIGN KEY (concert\_id) REFERENCES Concert(concert\_id) ON DELETE CASCADE,

FOREIGN KEY (artist\_id) REFERENCES Artist(artist\_id) ON DELETE CASCADE

);

CREATE TABLE Artist\_Instrument

(

Instrument VARCHAR(50) NOT NULL,

artist\_id INT NOT NULL,

PRIMARY KEY (Instrument, artist\_id),

FOREIGN KEY (artist\_id) REFERENCES Artist(artist\_id) ON DELETE CASCADE

);

**Tool Used**

* Database – MySql
* Backend – Python (mysql.connector)
* Frontend – Python (Streamlit)

**Populating the Database**

**Concert**:

insert into concert values(10001,'Fan-made Music Nights','2022-10-21 17:00:00','Rock','Bengaluru','GT Grounds');

insert into concert values(10002,'Bass From Base','2022-11-21 17:00:00','Rock','Bengaluru','GT Grounds');

insert into concert values(10003,'Blast From Past','2022-10-21 19:00:00','Retro','Mysore','Palace Grounds');

insert into concert values(10004,'Musical Fest','2022-10-28 09:00:00','Classical','Bengaluru','Kanteerava Hall');

insert into concert values(10005,'Quest To Music','2022-11-02 18:00:00','Rock','Mangaluru','Hard Rock Cafe');

**Artist**:

INSERT INTO Artist VALUES(1231,'Amogh N Rao',40,'Male');

INSERT INTO Artist VALUES(1232,'Meghana',32,'Female');

INSERT INTO Artist VALUES(1233,'Usha',56,'Female');

INSERT INTO Artist VALUES(1234,'Prasad',46,'Male');

INSERT INTO Artist VALUES(1235,'Nikhil',31,'Male');

INSERT INTO Artist VALUES(1236,'Sumukh',3,'Male');

INSERT INTO Artist VALUES(1237,'Sonu Nigam',101,'Male');

INSERT INTO Artist VALUES(1238,'Shreya Ghoshal',78,'Female');

INSERT INTO Artist VALUES(1239,'Chandan Shetty',24,'Male');

INSERT INTO Artist VALUES(1240,'Raghu Dixit',66,'Male');

INSERT INTO Artist VALUES(1241,'Sunnidhi Chauhan',89,'Female');

INSERT INTO Artist VALUES(1242,'Ananya Bhat',35,'Female');

INSERT INTO Artist VALUES(1243,'MS Kohli',12,'Male');

INSERT INTO Artist VALUES(1244,'Virat Sharma',8,'Male');

INSERT INTO Artist VALUES(1245,'KS Bharat',15,'Male');

INSERT INTO Artist VALUES(1246,'Neha Kakkar',65,'Female');

INSERT INTO Artist VALUES(1247,'Siddu',63,'Male');

INSERT INTO Artist VALUES(1248,'Rakesh Agarwal',45,'Male');

INSERT INTO Artist VALUES(1249,'Reena',23,'Female');

INSERT INTO Artist VALUES(1250,'Rocky',99,'Male');

**User**:

insert into User values(7890,'Prajwal','1995-10-12','9856327418','no');

insert into User values(7891,'Prakash','1994-10-12','9856327236','yes');

insert into User values(7892,'Akshya','1996-10-12','9856327766','no');

insert into User values(7893,'Alan','2002-10-12','9856327746','no');

insert into User values(7894,'Ajith','2001-10-12','9856327964','yes');

insert into User values(7895,'Chinmay','1989-10-12','9856327123','no');

insert into User values(7896,'Chetan','1978-10-12','9856327456','no');

insert into User values(7897,'Ganesh','1999-10-12','9856327968','no');

insert into User values(7898,'Karthik','2005-10-12','9856327754','no');

insert into User values(7899,'Krishna','1987-10-12','9856327365','yes');

**Ticket:**

insert into ticket values (100011,1,1500,'Fan-made Music Nights','vip',10001,7890);

insert into ticket values (100019,9,1000,'Fan-made Music Nights','platinum',10001,7891);

insert into ticket values (100018,10,1000,'Fan-made Music Nights','platinum',10001,7892);

insert into ticket values (100015,11,500,'Fan-made Music Nights','gold',10001,7895);

insert into ticket values (100016,13,500,'Fan-made Music Nights','gold',10001,7895);

insert into ticket values (100017,14,500,'Fan-made Music Nights','gold',10001,7895);

insert into ticket values (100012,4,1500,'Fan-made Music Nights','vip',10001,7897);

insert into ticket values (100021,5,500,'Bass From Base','gold',10002,7890);

insert into ticket values (100024,8,500,'Bass From Base','gold',10002,7891);

insert into ticket values (100022,10,1000,'Bass From Base','platinum',10002,7893);

insert into ticket values (100022,1,1500,'Bass From Base','vip',10002,7894);

insert into ticket values (100022,4,1500,'Bass From Base','vip',10002,7899);

insert into ticket values (100032,5,1500,'Blast From Past','vip',10003,7894);

insert into ticket values (100034,10,500,'Blast From Past','gold',10003,7893);

insert into ticket values (100036,11,1000,'Blast From Past','platinum',10003,7899);

insert into ticket values (100041,32,1500,'Musical Fest','vip',10004,7890);

insert into ticket values (100042,1,500,'Musical Fest','gold',10004,7892);

insert into ticket values (100043,12,500,'Musical Fest','gold',10004,7893);

insert into ticket values (100044,4,1000,'Musical Fest','platinum',10004,7899);

**Properties**:

insert into properties value (10001,'mike',10,1500);

insert into properties value (10002,'mike',7,1500);

insert into properties value (10003,'mike',9,1500);

insert into properties value (10004,'mike',5,1500);

insert into properties value (10005,'mike',4,1500);

insert into properties value (10001,'speaker',12,10000);

insert into properties value (10002,'speaker',15,10000);

insert into properties value (10003,'speaker',11,10000);

insert into properties value (10001,'screen',10,10000);

insert into properties value (10002,'screen',5,10000);

insert into properties value (10003,'screen',3,10000);

insert into properties value (10004,'screen',4,10000);

insert into properties value (10005,'screen',2,10000);

**performed\_by**:

insert into performed\_by values(10001,1231,40000);

insert into performed\_by values(10001,1235,35000);

insert into performed\_by values(10001,1240,10000);

insert into performed\_by values(10001,1242,50000);

insert into performed\_by values(10001,1245,45000);

insert into performed\_by values(10002,1250,100000);

insert into performed\_by values(10002,1249,25000);

insert into performed\_by values(10002,1246,55000);

insert into performed\_by values(10003,1236,65000);

insert into performed\_by values(10003,1237,10000);

insert into performed\_by values(10003,1238,65000);

insert into performed\_by values(10003,1247,60000);

insert into performed\_by values(10004,1242,68000);

insert into performed\_by values(10004,1247,42000);

insert into performed\_by values(10004,1234,15000);

insert into performed\_by values(10005,1244,25000);

insert into performed\_by values(10005,1239,35000);

insert into performed\_by values(10005,1233,45000);

insert into performed\_by values(10005,1241,55000);

**Artist\_instrument**:

insert into Artist\_Instrument values('Guitar',1231);

insert into Artist\_Instrument values('Keyboard',1231);

insert into Artist\_Instrument values('Flute',1233);

insert into Artist\_Instrument values('Keyboard',1235);

insert into Artist\_Instrument values('Drums',1236);

insert into Artist\_Instrument values('Guitar',1236);

insert into Artist\_Instrument values('Keyboard',1236);

insert into Artist\_Instrument values('Flute',1236);

insert into Artist\_Instrument values('Drums',1250);

insert into Artist\_Instrument values('Guitar',1243);

insert into Artist\_Instrument values('Flute',1245);

**Queries**

**Join queries**

1. Retreive the names and phone numbers of users who have not bought any tickets

**SQL**:

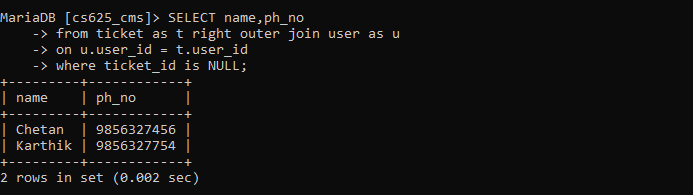
SELECT name,ph\_no

from ticket as t right outer join user as u

on u.user\_id = t.user\_id

where ticket\_id is NULL;

**Screenshot**:



1. List the artist names who play atleast one instrument

**SQL**:

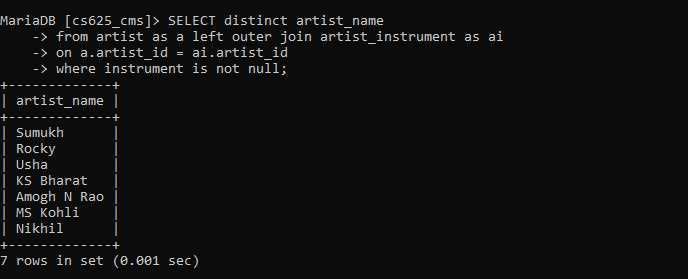
SELECT distinct artist\_name

from artist as a left outer join artist\_instrument as ai

on a.artist\_id = ai.artist\_id

where instrument is not null;

**Screenshot**:



**Nested queries**

1. List the user names and user\_id of users who have attended atleast 1 concert in which artist with artist\_id=1242 performed.

**SQL**:

select distinct u.user\_id,u.name

from user u,ticket t

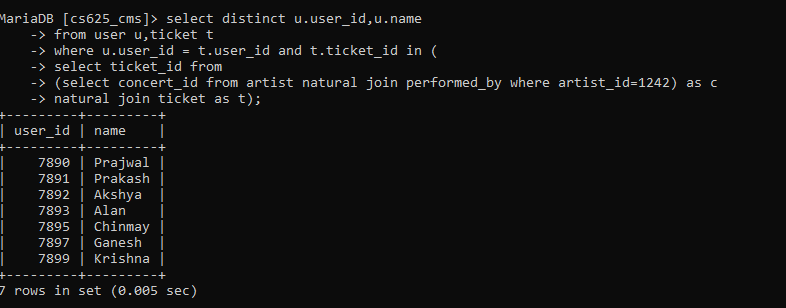
where u.user\_id = t.user\_id and t.ticket\_id in (

select ticket\_id from

(select concert\_id from artist natural join performed\_by where artist\_id=1242) as c

natural join ticket as t);

**Screenshot**:



1. Artist who have sung more than 50 songs and performed in any Classical concerts

**SQL**:

select artist\_id,artist\_name

from artist

where num\_of\_songs > 50

and artist\_id = ANY (

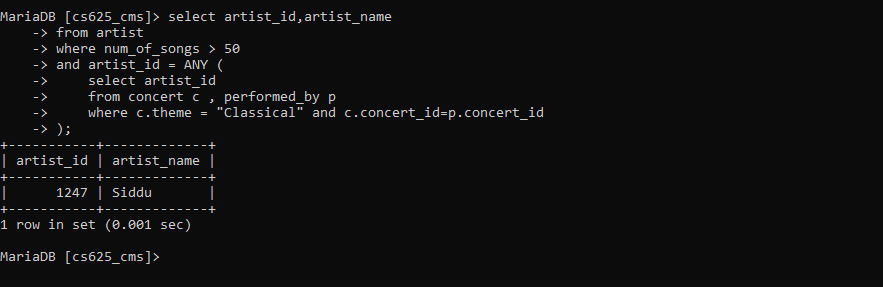
select artist\_id

from concert c , performed\_by p

where c.theme = "Classical" and c.concert\_id=p.concert\_id

);

**Screenshot**:



**Co-related queries**

1. Lists the users who have attended Rock concerts

**SQL**:

select distinct u.name,u.user\_id

from user u,ticket t

where u.user\_id = t.user\_id and Exists(

select concert\_id

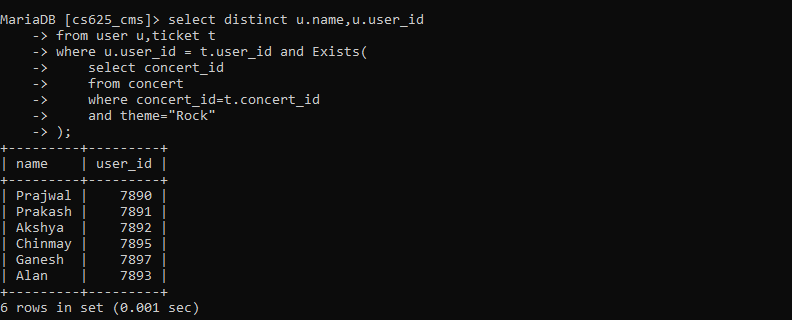
from concert

where concert\_id=t.concert\_id

and theme="Rock"

);

**Screenshot**:



1. Artist who does not play any instrument

**SQL**:

select a.artist\_id,a.artist\_name

from artist a

where NOT Exists(

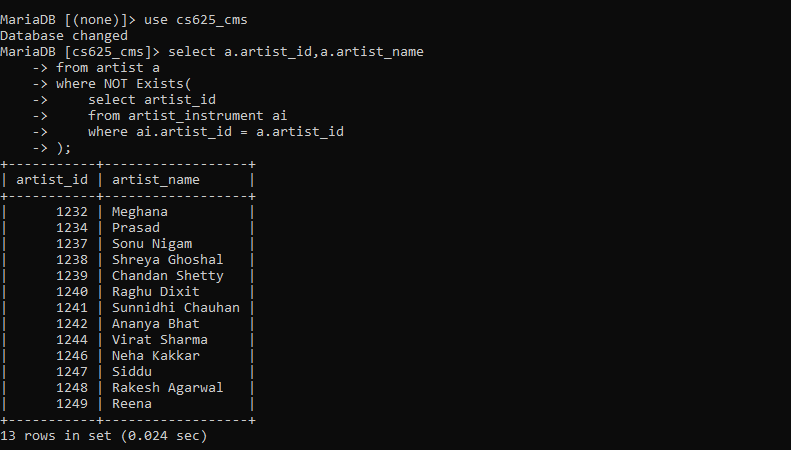
select artist\_id

from artist\_instrument ai

where ai.artist\_id = a.artist\_id

);

**Screenshot**:



**Aggregate Functions**

1. Find the artist names who taken the minimum fee to perform in a concert

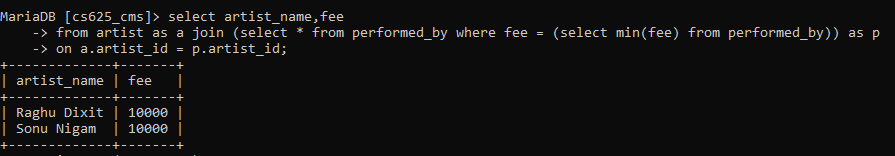
**SQL**:

select artist\_name,fee

from artist as a join (select \* from performed\_by where fee = (select min(fee) from performed\_by)) as p

on a.artist\_id = p.artist\_id;

**Screenshot:**



1. Retrieve the number of instruments played by artist with artist\_id=1235

**SQL**:

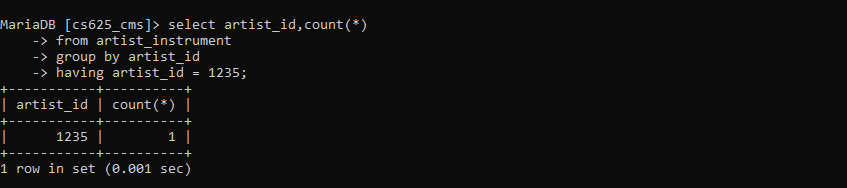
select artist\_id,count(\*)

from artist\_instrument

group by artist\_id

having artist\_id = 1235;

**Screenshot**:



1. List the concert\_id of all the concerts and the number of artists who performed in that concert

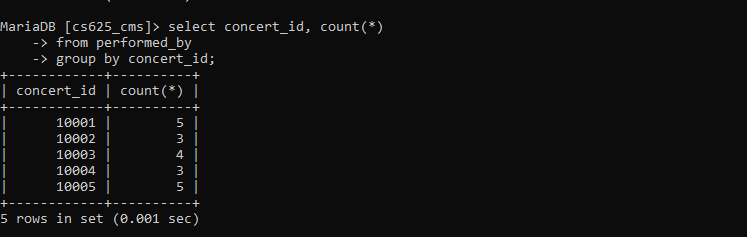
**SQL**:

select concert\_id, count(\*)

from performed\_by

group by concert\_id;

**Screenshot**:



**Set Operations**

1. Find user ids who have attended rock concert during the month of Oct 2022 or Nov 2022

**SQL**:

select t.user\_id

from ticket t, concert c

where t.concert\_id = c.concert\_id and c.theme='Rock' and Date\_time like '2022-10-%'

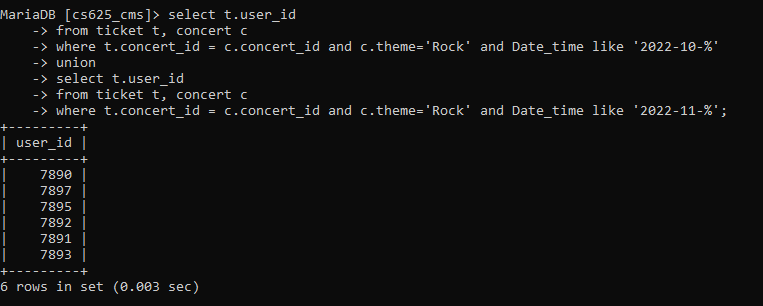
union

select t.user\_id

from ticket t, concert c

where t.concert\_id = c.concert\_id and c.theme='Rock' and Date\_time like '2022-11-%';

**Screenshot**:



1. Artists who charged fee greater than 30000 and plays more than 2 instruments

**SQL:**

select artist\_id,artist\_name

from artist natural join artist\_instrument

group by artist\_id

having count(\*) > 2

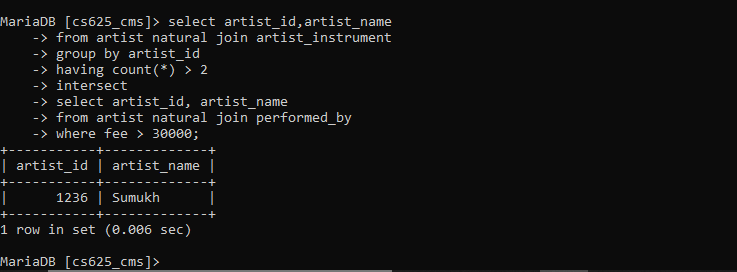
intersect

select artist\_id, artist\_name

from artist natural join performed\_by

where fee > 30000;

**Screenshot**:



1. Users who have attended concerts in which artist\_id=1247 performed and not attended any other concerts

**SQL**:

select user\_id

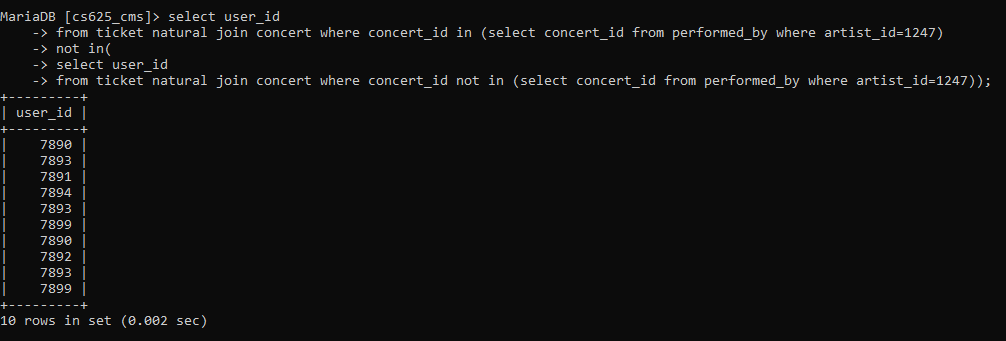
from ticket natural join concert where concert\_id in (select concert\_id from performed\_by where artist\_id=1247)

not in(

select user\_id

from ticket natural join concert where concert\_id not in (select concert\_id from performed\_by where artist\_id=1247));

**Screenshot**:



1. Artits who play Guitar and keyboard but not flute

**SQL**:

(select artist\_id,artist\_name

from artist natural join artist\_instrument

where instrument = 'Guitar'

intersect

select artist\_id,artist\_name

from artist natural join artist\_instrument

where instrument = 'Keyboard')

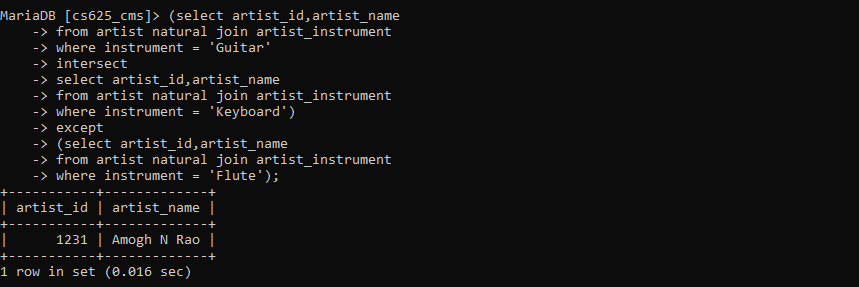
except

(select artist\_id,artist\_name

from artist natural join artist\_instrument

where instrument = 'Flute');

**Screenshot**:



**View**

**Demonstrate creation and querying one view**

Prop\_cost is a view which stores the total cost of all properties a concert requires.

**SQL**:

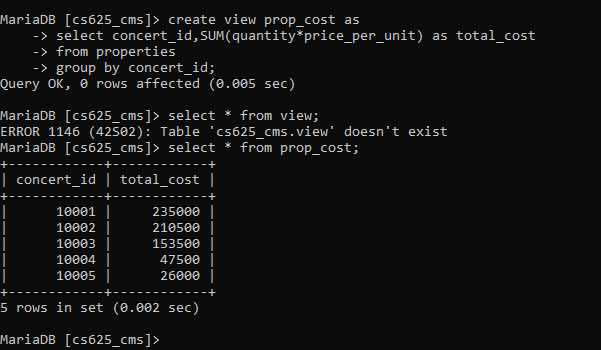
create view prop\_cost as

select concert\_id,SUM(quantity\*price\_per\_unit) as total\_cost

from properties

group by concert\_id;

**Screenshot**:



**Query**: Find the average of total cost of properties for concerts that took place in 'Bengaluru'

**SQL**:

select avg(total\_cost) as avg\_total\_cost

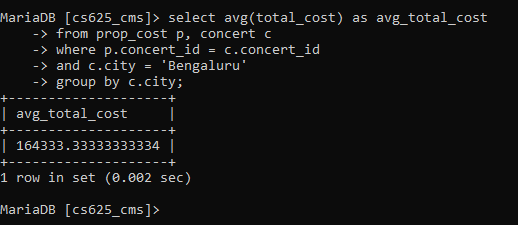
from prop\_cost p, concert c

where p.concert\_id = c.concert\_id

and c.city = 'Bengaluru'

group by c.city;

**Output**:



**Functions**

The below function takes 2 parameters: artist\_id, date\_time and returns the count of the concerts that the artist(artist\_id)nis performing in that particular date(date\_time).

DELIMITER $$

CREATE FUNCTION is\_performing(artist\_id INT,date\_time TimeStamp)

RETURNS INT

BEGIN

DECLARE performing INT;

SET performing = (SELECT COUNT(\*)

FROM concert c,performed\_by p

where c.concert\_id=p.concert\_id and

date(c.Date\_time)=date(date\_time) and

p.artist\_id=artist\_id);

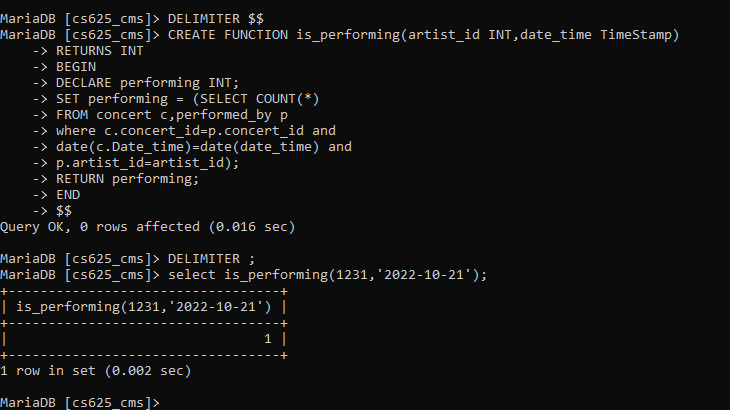
RETURN performing;

END

$$

DELIMITER ;

**Output**:



**Triggers**

The below trigger makes use of is\_performing function. While inserting to the table performed\_by, if the artist is performing in a different concert on the same date as the new concert, it blocks the insert operation.

DELIMITER $$

CREATE TRIGGER insert\_before\_performed\_by

BEFORE INSERT

ON performed\_by FOR EACH ROW

BEGIN

DECLARE concert\_date date;

DECLARE performing INT;

DECLARE err\_msg VARCHAR(100);

SET err\_msg = 'Artist not available....performing in a different concert on the same day:(';

SET concert\_date = (SELECT Date\_time FROM concert c where c.concert\_id=new.concert\_id);

SET performing = (SELECT is\_performing(new.artist\_id,date(concert\_date)));

IF performing > 0 THEN

SIGNAL SQLSTATE'45000'

SET MESSAGE\_TEXT = err\_msg;

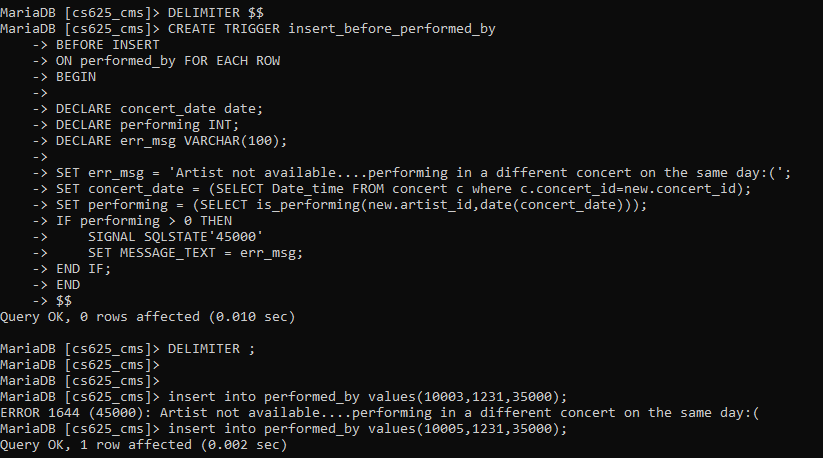
END IF;

END

$$

DELIMITER ;

**Screenshot**:



**Developing a Frontend**

The frontend should support

1. Addition, Modification and Deletion of records from any chosen table

2. There should be a window to accept and run any SQL statement and display the result