DBMS - MINI PROJECT

CONCERT MANAGEMENT SYSTEM

Submitted By:

Name: Amogh N Rao

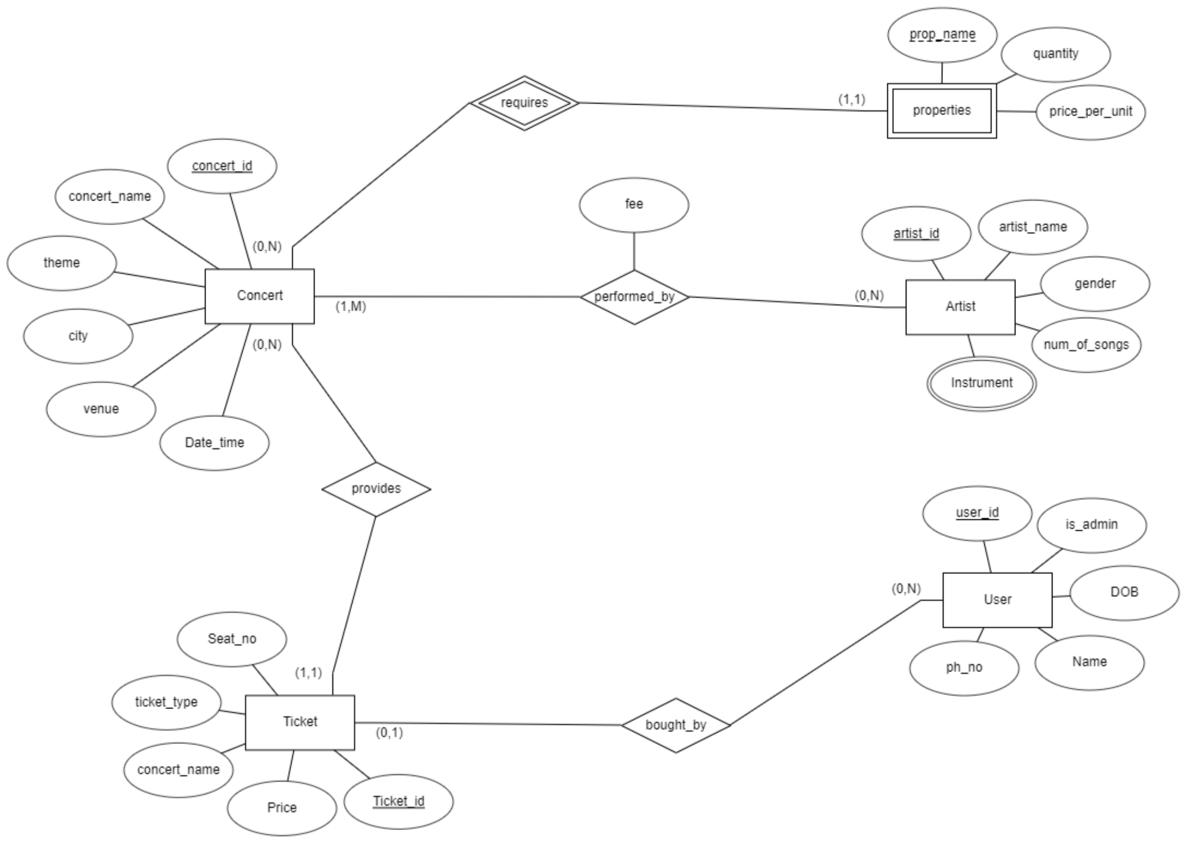
SRN: PES1UG20CS625

V Semester Section K

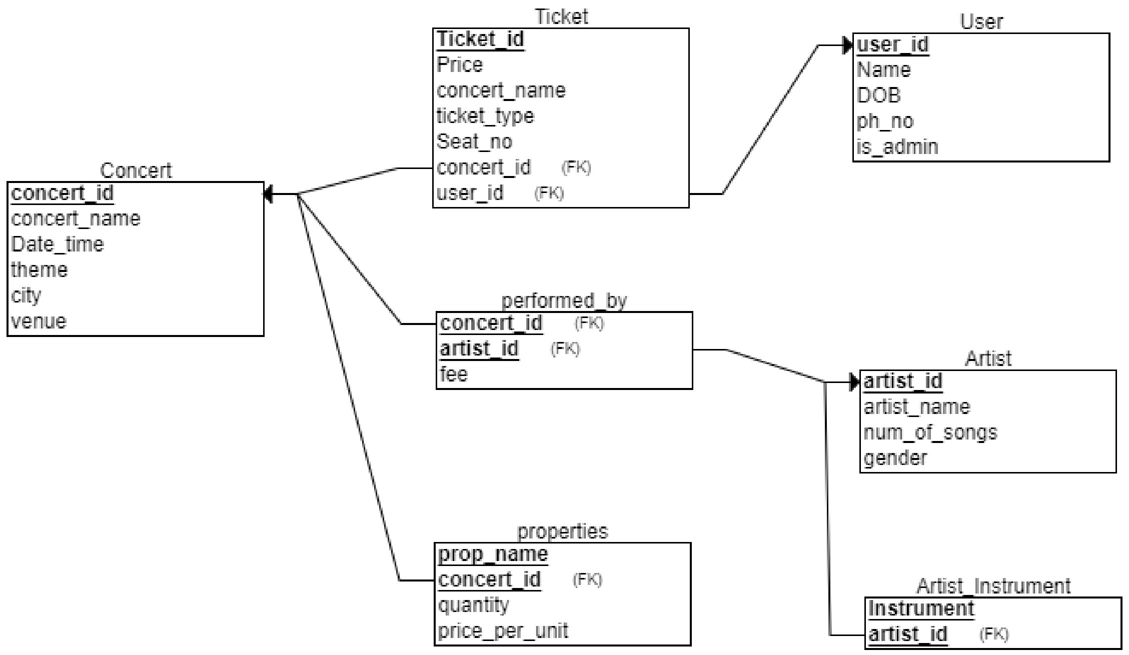
ABSTRACT

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');
```

<u>User</u>:

```
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);
```

<u>Properties</u>:

```
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

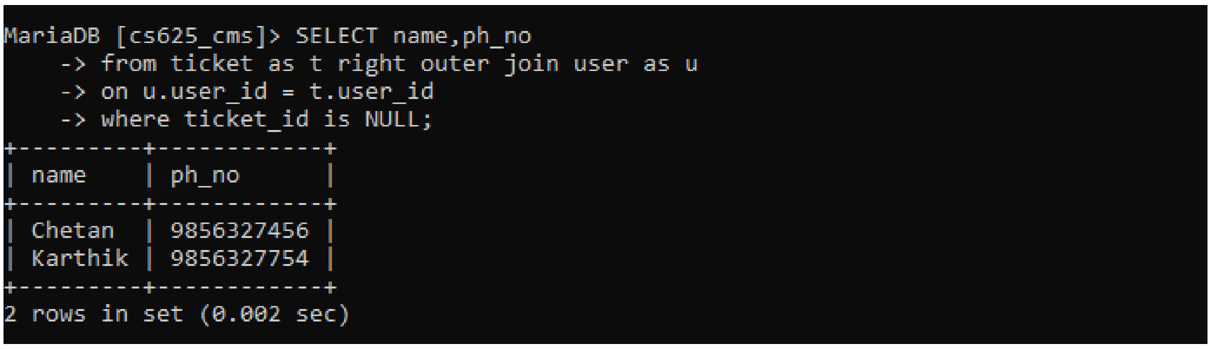
Join queries

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:



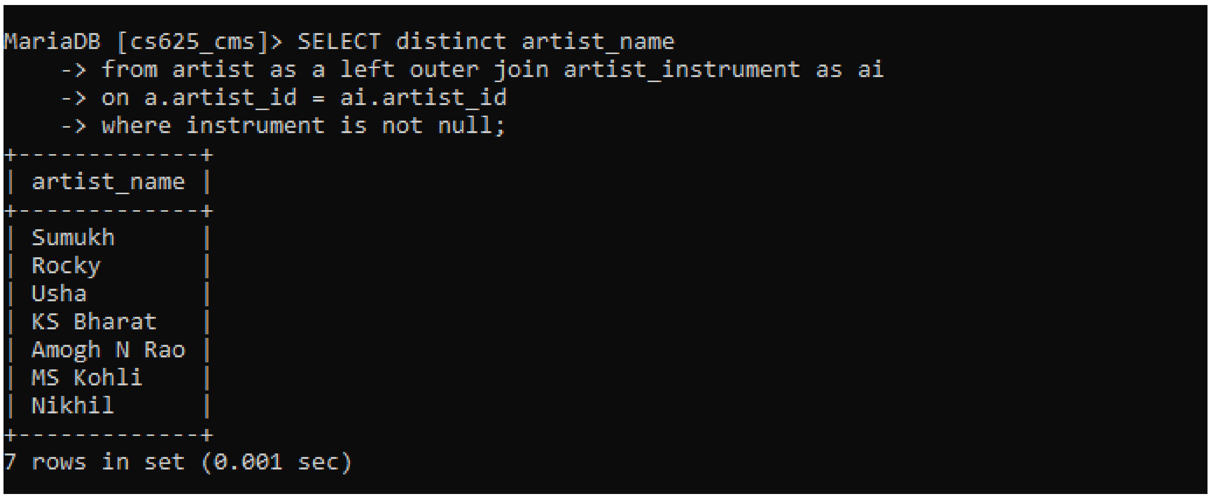
```
MariaDB [cs625_cms]> SELECT name,ph_no
    -> from ticket as t right outer join user as u
    -> on u.user_id = t.user_id
    -> where ticket_id is NULL;
+---------+------------+
| name    | ph_no      |
+---------+------------+
| Chetan  | 9856327456 |
| Karthik | 9856327754 |
+---------+------------+
2 rows in set (0.002 sec)
```

2. 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:



```
MariaDB [cs625_cms]> 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;
+-------------+
| artist_name |
+-------------+
| Sumukh      |
| Rocky       |
| Usha        |
| KS Bharat   |
| Amogh N Rao |
| MS Kohli    |
| Nikhil      |
+-------------+
7 rows in set (0.001 sec)
```

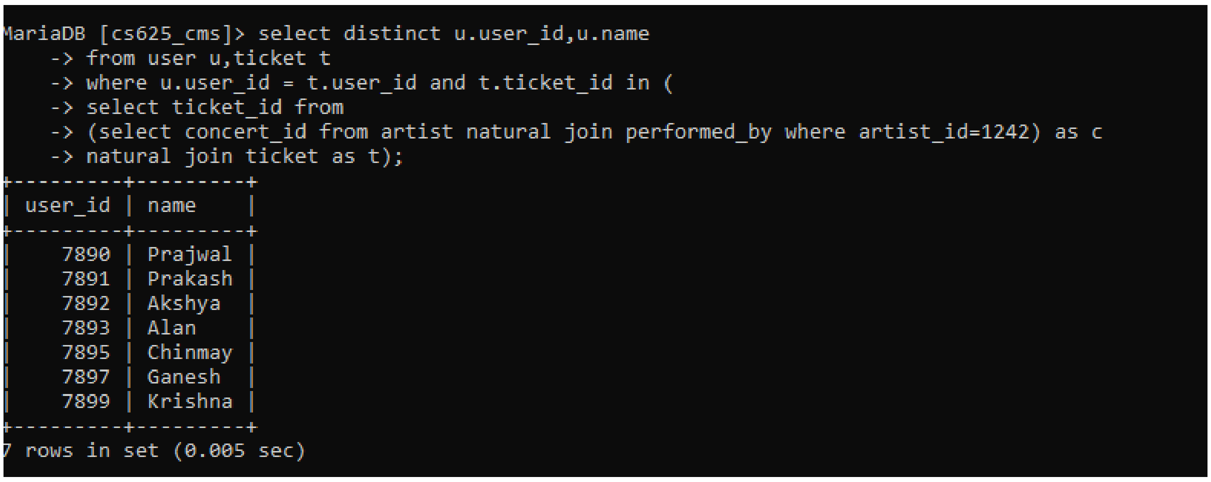
<u>Nested queries</u>

3. 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:



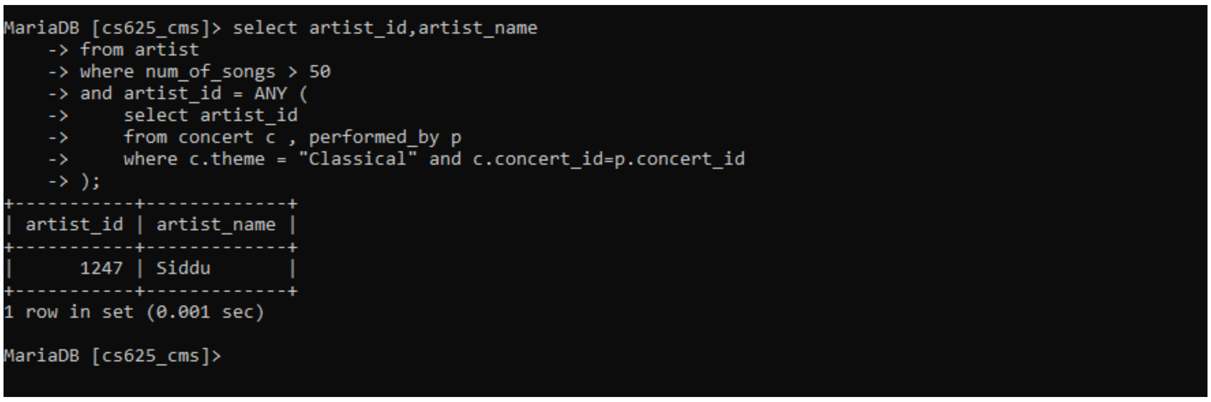
```
MariaDB [cs625_cms]> select distinct u.user_id,u.name
    -> from user u,ticket t
    -> where u.user_id = t.user_id and t.ticket_id in (
    -> select ticket_id from
    -> (select concert_id from artist natural join performed_by where artist_id=1242) as c
    -> natural join ticket as t);
+---------+---------+
| user_id | name    |
+---------+---------+
|    7890 | Prajwal |
|    7891 | Prakash |
|    7892 | Akshya  |
|    7893 | Alan    |
|    7895 | Chinmay |
|    7897 | Ganesh  |
|    7899 | Krishna |
+---------+---------+
7 rows in set (0.005 sec)
```

4. 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:



```
MariaDB [cs625_cms]> 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
    -> );
+-----------+-------------+
| artist_id | artist_name |
+-----------+-------------+
|      1247 | Siddu       |
+-----------+-------------+
1 row in set (0.001 sec)

MariaDB [cs625_cms]>
```

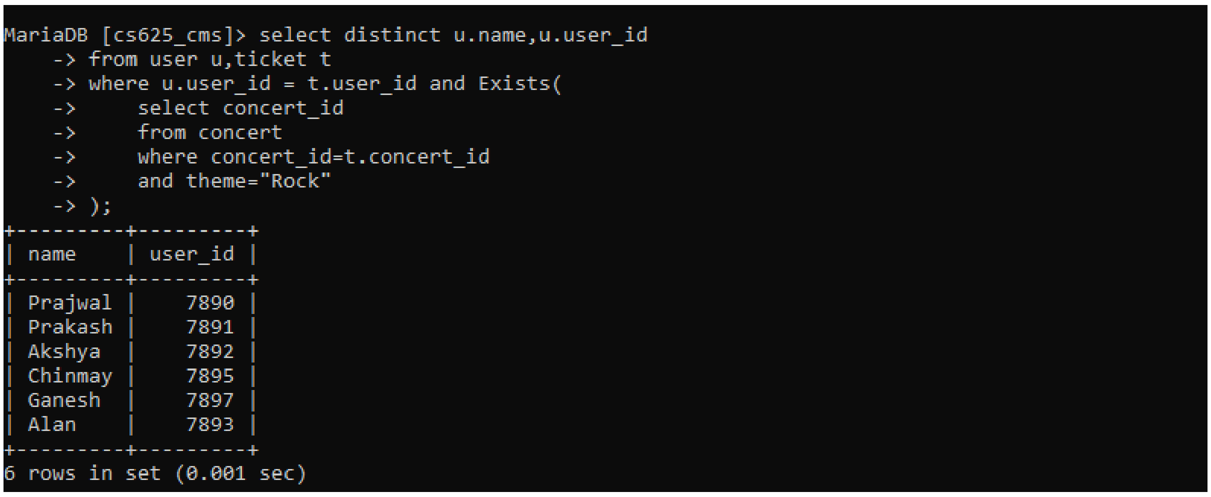
Co-related queries

5. 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:

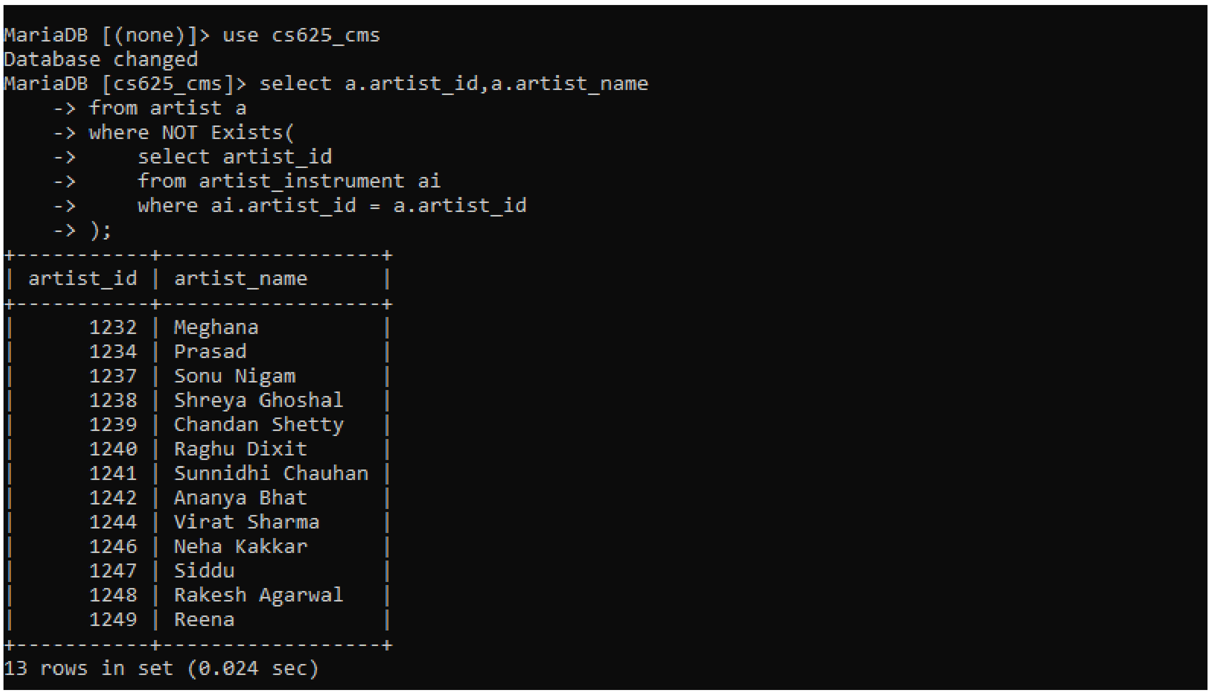


```
MariaDB [cs625_cms]> select distinct u.name,u.user_id
    -> from user u,ticket t
    -> where u.user_id = t.user_id and Exists(
    ->     select concert_id
    ->     from concert
    ->     where concert_id=t.concert_id
    ->     and theme="Rock"
    -> );
+---------+---------+
| name    | user_id |
+---------+---------+
| Prajwal |    7890 |
| Prakash |    7891 |
| Akshya  |    7892 |
| Chinmay |    7895 |
| Ganesh  |    7897 |
| Alan    |    7893 |
+---------+---------+
6 rows in set (0.001 sec)
```

6. 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:



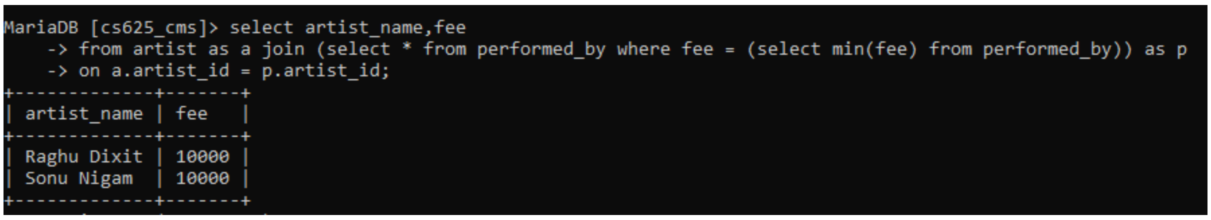
```
MariaDB [(none)]> use cs625_cms
Database changed
MariaDB [cs625_cms]> select a.artist_id,a.artist_name
    -> from artist a
    -> where NOT Exists(
    ->     select artist_id
    ->     from artist_instrument ai
    ->     where ai.artist_id = a.artist_id
    -> );
+-----------+------------------+
| artist_id | artist_name      |
+-----------+------------------+
|      1232 | Meghana          |
|      1234 | Prasad           |
|      1237 | Sonu Nigam       |
|      1238 | Shreya Ghoshal   |
|      1239 | Chandan Shetty   |
|      1240 | Raghu Dixit      |
|      1241 | Sunnidhi Chauhan |
|      1242 | Ananya Bhat      |
|      1244 | Virat Sharma     |
|      1246 | Neha Kakkar      |
|      1247 | Siddu            |
|      1248 | Rakesh Agarwal   |
|      1249 | Reena            |
+-----------+------------------+
13 rows in set (0.024 sec)
```

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:



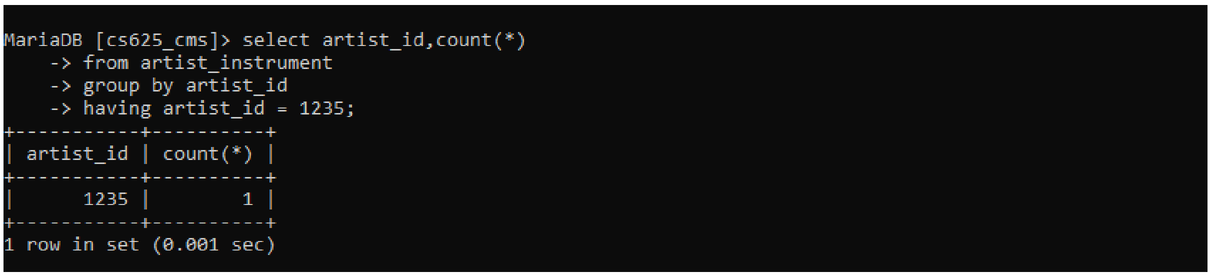
```
MariaDB [cs625_cms]> select artist_name,fee
    -> from artist as a join (select * from performed_by where fee = (select min(fee) from performed_by)) as p
    -> on a.artist_id = p.artist_id;
+-------------+-------+
| artist_name | fee   |
+-------------+-------+
| Raghu Dixit | 10000 |
| Sonu Nigam  | 10000 |
+-------------+-------+
```

2. 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:



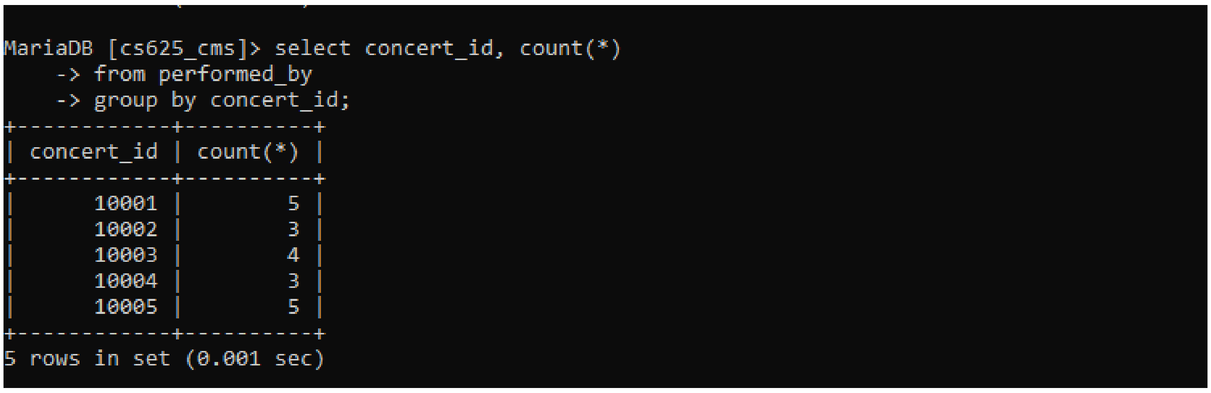
```
MariaDB [cs625_cms]> select artist_id,count(*)
    -> from artist_instrument
    -> group by artist_id
    -> having artist_id = 1235;
+-----------+----------+
| artist_id | count(*) |
+-----------+----------+
|      1235 |        1 |
+-----------+----------+
1 row in set (0.001 sec)
```

3. 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:



```
MariaDB [cs625_cms]> select concert_id, count(*)
    -> from performed_by
    -> group by concert_id;
+------------+----------+
| concert_id | count(*) |
+------------+----------+
|      10001 |        5 |
|      10002 |        3 |
|      10003 |        4 |
|      10004 |        3 |
|      10005 |        5 |
+------------+----------+
5 rows in set (0.001 sec)
```

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-%'

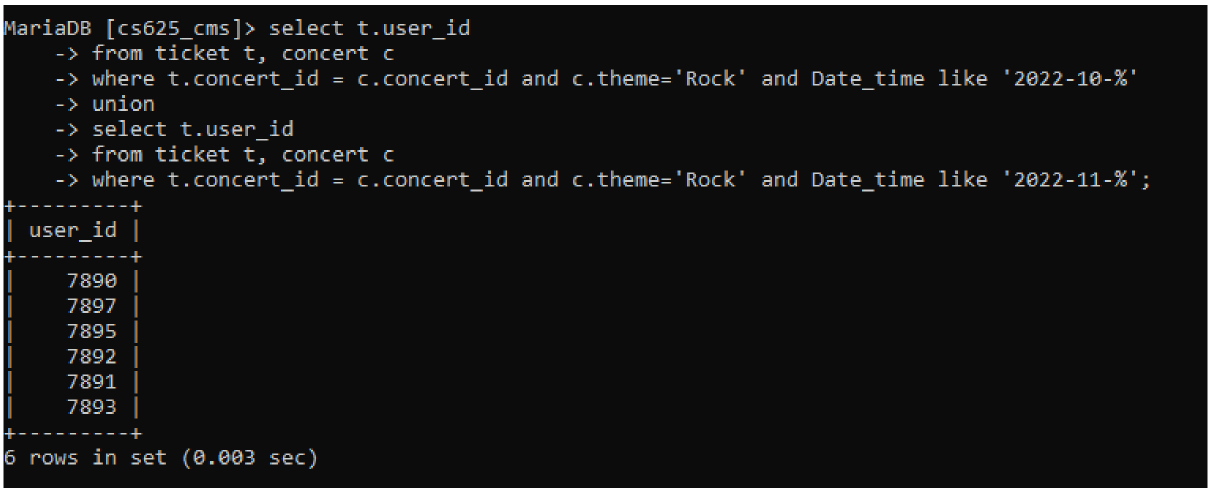
   union

   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:



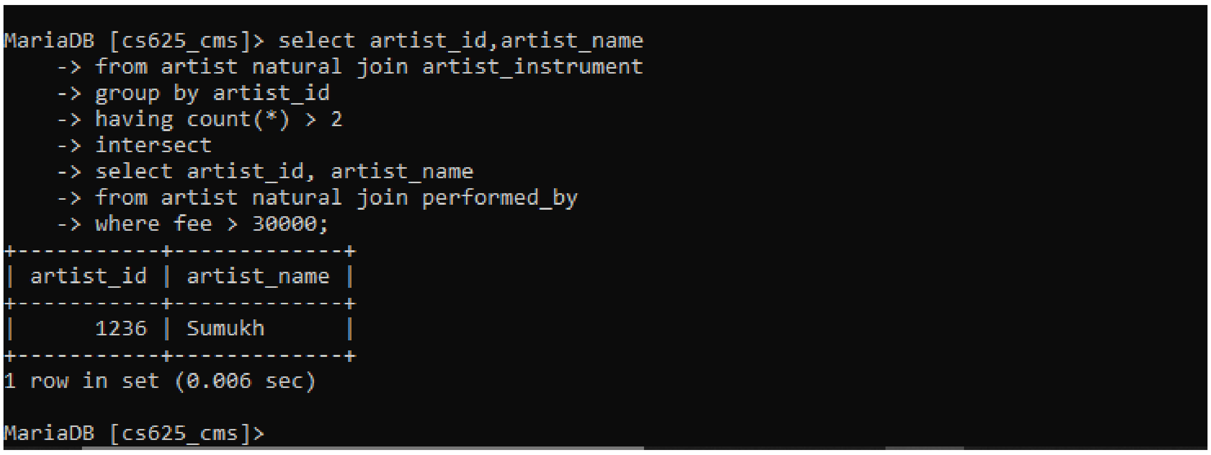
```
MariaDB [cs625_cms]> select t.user_id
    -> from ticket t, concert c
    -> where t.concert_id = c.concert_id and c.theme='Rock' and Date_time like '2022-10-%'
    -> union
    -> select t.user_id
    -> from ticket t, concert c
    -> where t.concert_id = c.concert_id and c.theme='Rock' and Date_time like '2022-11-%';
+---------+
| user_id |
+---------+
|    7890 |
|    7897 |
|    7895 |
|    7892 |
|    7891 |
|    7893 |
+---------+
6 rows in set (0.003 sec)
```

2. 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:



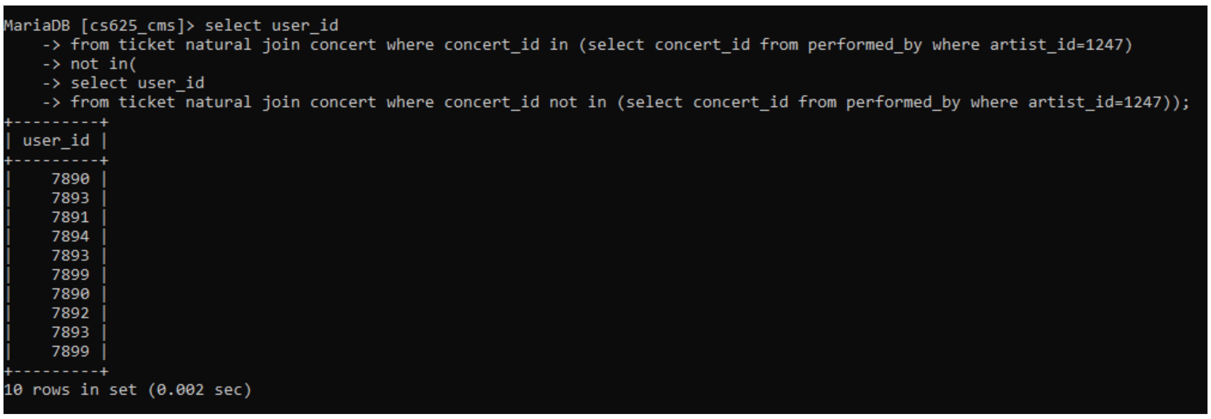
```
MariaDB [cs625_cms]> select artist_id,artist_name
    -> from artist natural join artist_instrument
    -> group by artist_id
    -> having count(*) > 2
    -> intersect
    -> select artist_id, artist_name
    -> from artist natural join performed_by
    -> where fee > 30000;
+-----------+-------------+
| artist_id | artist_name |
+-----------+-------------+
|      1236 | Sumukh      |
+-----------+-------------+
1 row in set (0.006 sec)

MariaDB [cs625_cms]>
```

3. 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:



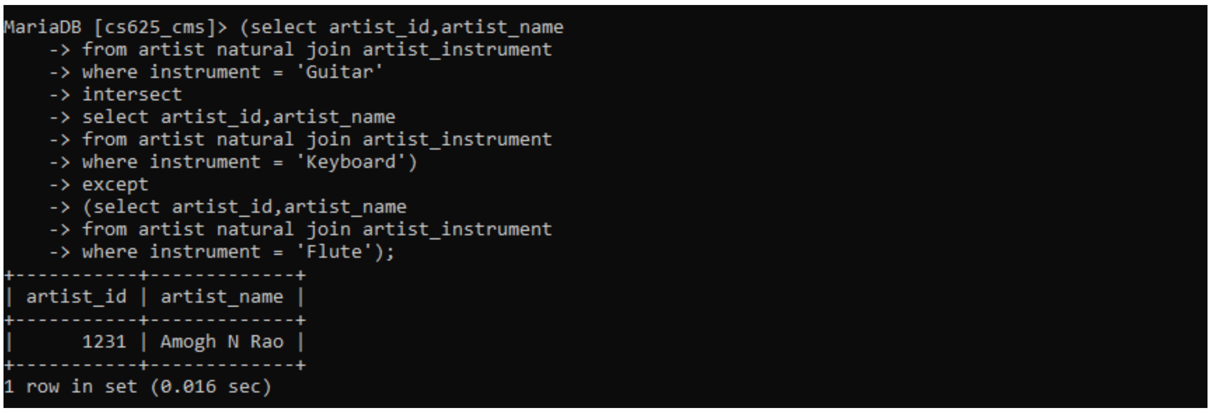
```
MariaDB [cs625_cms]> select user_id
    -> from ticket natural join concert where concert_id in (select concert_id from performed_by where artist_id=1247)
    -> not in(
    -> select user_id
    -> from ticket natural join concert where concert_id not in (select concert_id from performed_by where artist_id=1247));
+---------+
| user_id |
+---------+
|    7890 |
|    7893 |
|    7891 |
|    7894 |
|    7893 |
|    7899 |
|    7890 |
|    7892 |
|    7893 |
|    7899 |
+---------+
10 rows in set (0.002 sec)
```

4. 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:



```
MariaDB [cs625_cms]> (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');
+-----------+-------------+
| artist_id | artist_name |
+-----------+-------------+
|      1231 | Amogh N Rao |
+-----------+-------------+
1 row in set (0.016 sec)
```

<u>View</u>

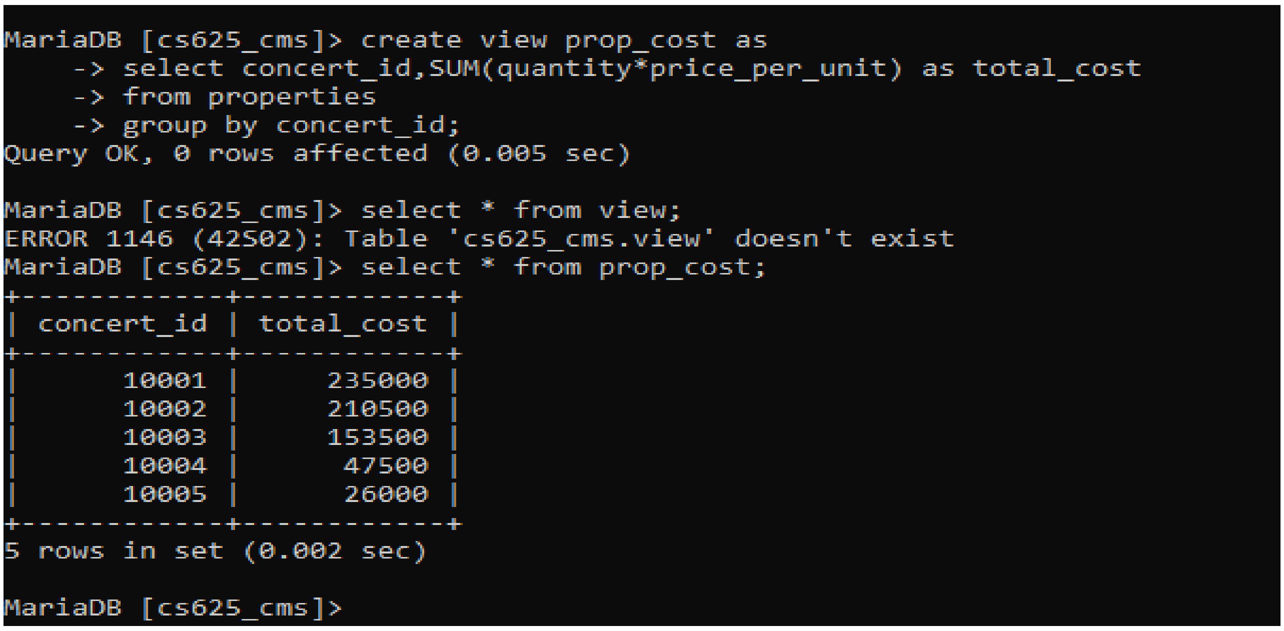
Demonstrate creation and querying one view

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:



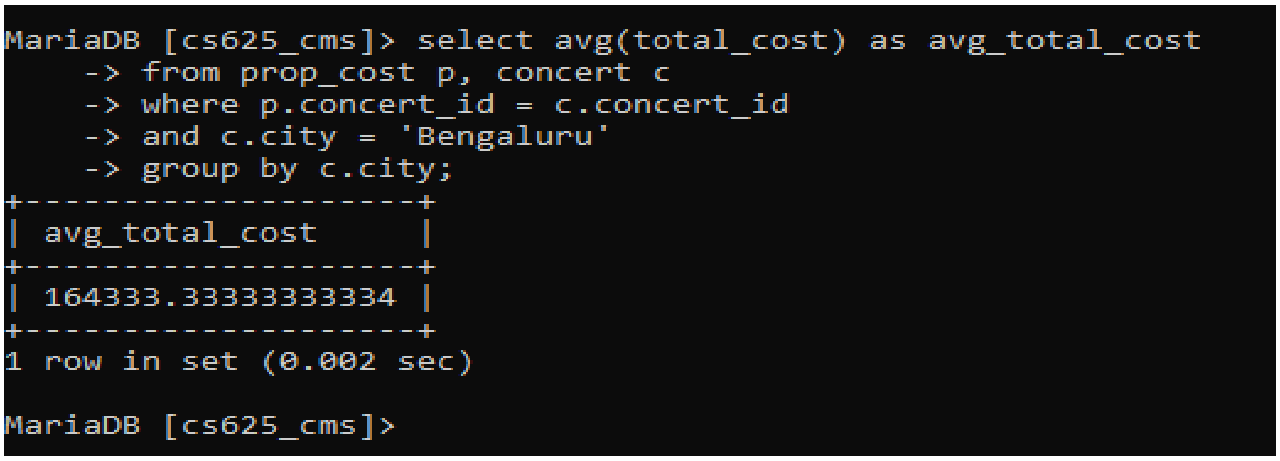
```
MariaDB [cs625_cms]> create view prop_cost as
    -> select concert_id,SUM(quantity*price_per_unit) as total_cost
    -> from properties
    -> group by concert_id;
Query OK, 0 rows affected (0.005 sec)

MariaDB [cs625_cms]> select * from view;
ERROR 1146 (42S02): Table 'cs625_cms.view' doesn't exist
MariaDB [cs625_cms]> select * from prop_cost;
+------------+------------+
| concert_id | total_cost |
+------------+------------+
|      10001 |     235000 |
|      10002 |     210500 |
|      10003 |     153500 |
|      10004 |      47500 |
|      10005 |      26000 |
+------------+------------+
5 rows in set (0.002 sec)

MariaDB [cs625_cms]>
```

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:



```
MariaDB [cs625_cms]> 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;
+--------------------+
| avg_total_cost     |
+--------------------+
| 164333.33333333334 |
+--------------------+
1 row in set (0.002 sec)

MariaDB [cs625_cms]>
```

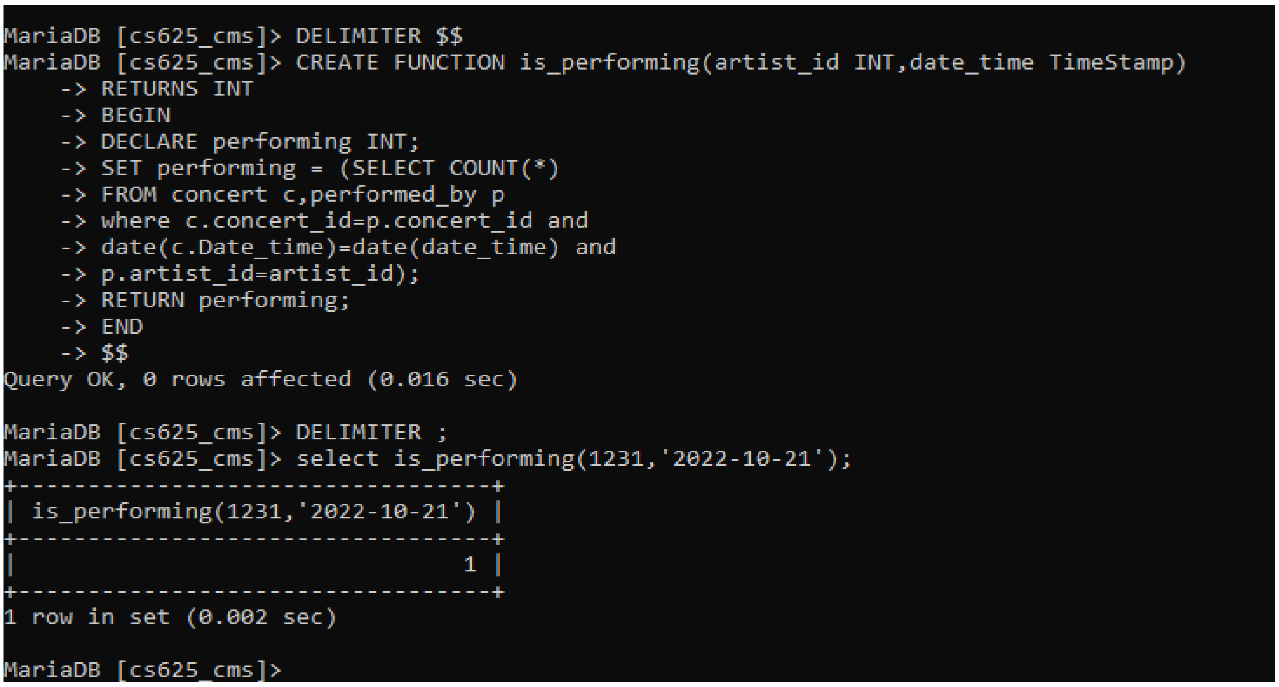
Functions

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:



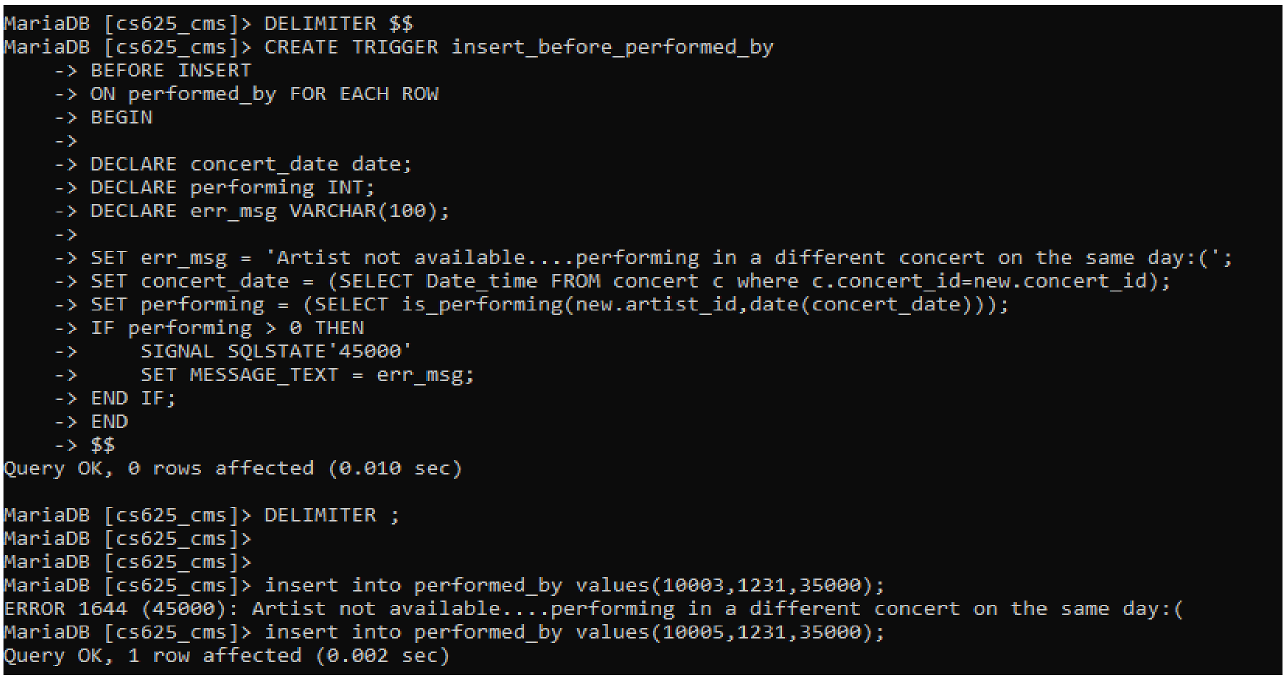
```
MariaDB [cs625_cms]> DELIMITER $$
MariaDB [cs625_cms]> CREATE FUNCTION is_performing(artist_id INT,date_time TimeStamp)
    -> RETURNS INT
    -> BEGIN
    -> DECLARE performing INT;
    -> SET performing = (SELECT COUNT(*)
    -> FROM concert c,performed_by p
    -> where c.concert_id=p.concert_id and
    -> date(c.Date_time)=date(date_time) and
    -> p.artist_id=artist_id);
    -> RETURN performing;
    -> END
    -> $$
Query OK, 0 rows affected (0.016 sec)

MariaDB [cs625_cms]> DELIMITER ;
MariaDB [cs625_cms]> select is_performing(1231,'2022-10-21');
+----------------------------------+
| is_performing(1231,'2022-10-21') |
+----------------------------------+
|                                1 |
+----------------------------------+
1 row in set (0.002 sec)

MariaDB [cs625_cms]>
```

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:



```
MariaDB [cs625_cms]> DELIMITER $$
MariaDB [cs625_cms]> CREATE TRIGGER insert_before_performed_by
    -> BEFORE INSERT
    -> ON performed_by FOR EACH ROW
    -> BEGIN
    ->
    -> DECLARE concert_date date;
    -> DECLARE performing INT;
    -> DECLARE err_msg VARCHAR(100);
    ->
    -> SET err_msg = 'Artist not available....performing in a different concert on the same day:(';
    -> SET concert_date = (SELECT Date_time FROM concert c where c.concert_id=new.concert_id);
    -> SET performing = (SELECT is_performing(new.artist_id,date(concert_date)));
    -> IF performing > 0 THEN
    ->     SIGNAL SQLSTATE'45000'
    ->     SET MESSAGE_TEXT = err_msg;
    -> END IF;
    -> END
    -> $$
Query OK, 0 rows affected (0.010 sec)

MariaDB [cs625_cms]> DELIMITER ;
MariaDB [cs625_cms]>
MariaDB [cs625_cms]>
MariaDB [cs625_cms]> insert into performed_by values(10003,1231,35000);
ERROR 1644 (45000): Artist not available....performing in a different concert on the same day:(
MariaDB [cs625_cms]> insert into performed_by values(10005,1231,35000);
Query OK, 1 row affected (0.002 sec)
```

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