DBMS PROJECT WORK

“MOVIE TICKET BOOKING SYSTEM”

Group-02:
Aanshi Patwari (AU1841004)
Dipika Pawar (AU1841052)
Miracle Rindani (AU1841017)
Nildeep Jadav (AU1641024)

Guided By: Prof. Shefali Naik

DESCRIPTION

Box offices are becoming a thing of the past now. Nobody is interested in standing in long queues just to buy tickets to watch a movie. Our **movie ticket booking system** makes the task of booking tickets to watch movies easier and allows you to enjoy the movie-watching experience while sitting in seats of your choice and having your refreshments delivered to you at your seats.

Our website is a **two-user** website: you can either continue further as **the admin** or continue further as **a viewer**.

When you enter the website as **the admin**, you can manage various cineplexes present in the same city. As an admin, you can add, and update movies in different cineplexes. You can also add, update and delete the choice and quantity of refreshments in all the cineplexes. The website also allows the admin to generate a daily box office collection report for all the movies currently being shown in all the cineplexes and suggest the admin to analyze if he/she should update movie timing, i.e., increase/decrease the number of shows in a particular/some/all cineplexes.

When a user enters the website as **a viewer**, you can book tickets for an ongoing movie and also pre-order your refreshments so that you don't need to wait in line to buy your refreshments at the cineplex. The website shows you a list of both, **upcoming** and **ongoing movies**. You can filter the movie list based on the genre or language. You can book tickets for an ongoing movie. The site shows you the details of the selected movie like **movie name**, **actors**, **director**, **genre**, **movie description**, **movie certification (U/UA/A)**, and **movie ratings**. If you wish to watch the movie, then you need to click on the **"Proceed**

to See Showtimes" button. Upon pressing it, the **Showtimes page** opens up which contains the list of all the cineplexes in which the movie is currently being shown, along with the movie showtimes for all the cineplexes. You can also view the showtimes for different dates. Upon selecting the desired showtime, you are directed to the **Seats page**. You select the desired number of seats from the unoccupied seats. The seats are segregated into three categories as **Platinum**, **Gold**, and **Silver**. The cost of a seat depends on the showtime and the category it belongs to.

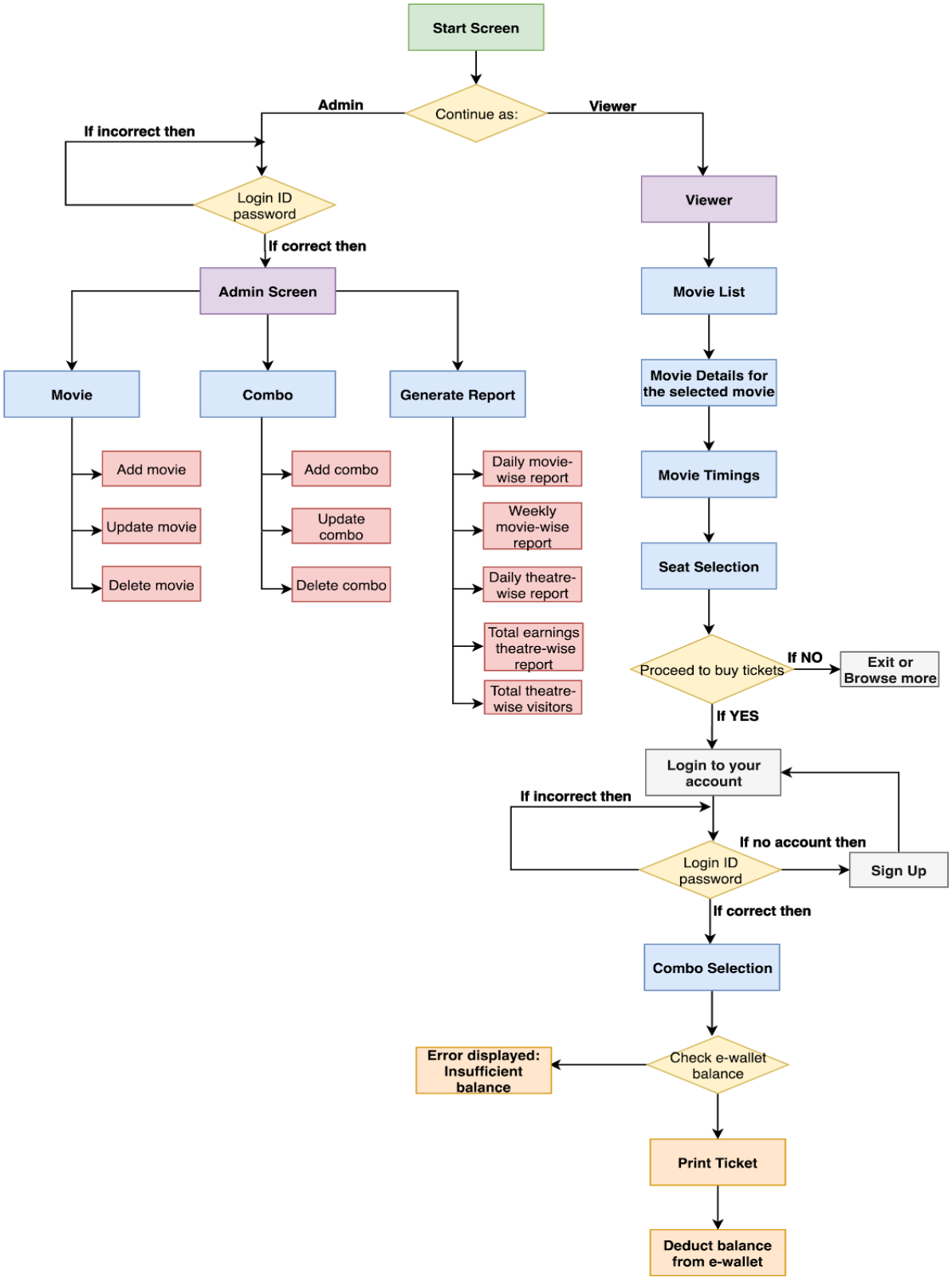
Once the seats are selected,the user needs to sign in/sign up to continue further to pay for the tickets. Then the viewer is asked if he/she wants to pre-order his/her refreshments. If a viewer wishes to do so, then he/she is redirected to the **Refreshments page**. The viewer can select the type and specify a quantity from a given menu of specified combos. The viewer can pay using his/her e-wallet which is linked with his/her account.

Upon the success of payment, the cost of the tickets is automatically deducted from the viewer's e-wallet and the ticket is generated and also pretends the user to book tickets and refreshments if he/she hasn't enough balance in his/her account.

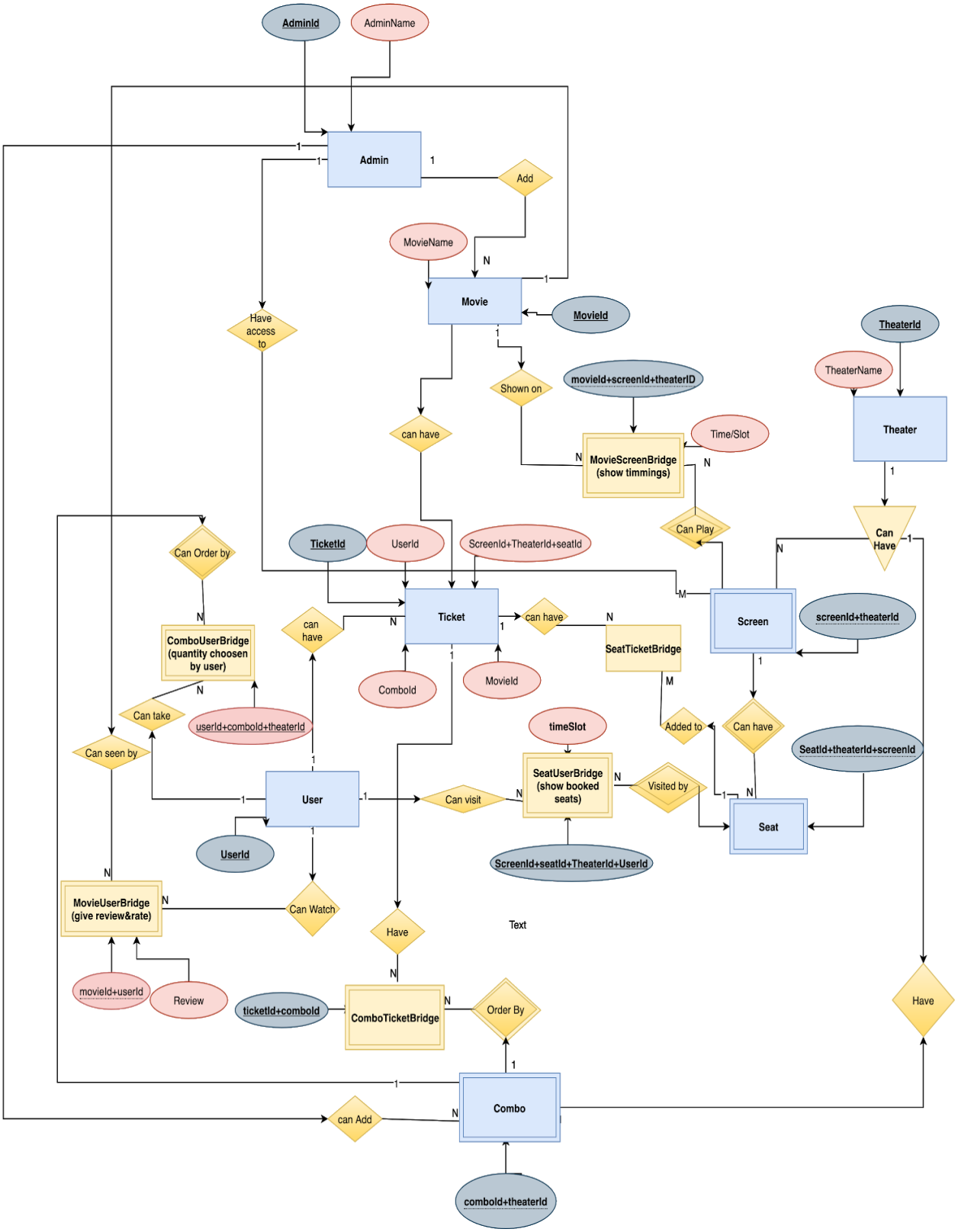
This project is made using:

- Frontend ⇒ HTML + CSS + Javascript
- Backend ⇒ Python (Flask framework)
- Database ⇒ MySQL

FLOW OF PROJECT



ENTITY-RELATIONSHIP DIAGRAM



DATA DICTIONARY

1. Admin Table

Column	Type	Null	Default	Links to
username *(Primary)*	varchar(30)	No		
password	varchar(8)	No		

2. Movie table

Column	Type	Null	Default	Links to
movieId *(Primary)*	int(11)	No		
movieName	varchar(40)	No		
movieDiscription	mediumtext	No		
movieDirector	varchar(30)	No		
movieProducer	varchar(30)	No		
movieGenre	varchar(20)	No		
movieCertificate	varchar(10)	No		
movieDuration	time	No		
movieRating	float	No		
movieReleaseDate	date	No		
movieLanguage	varchar(20)	No		
movieStatus	varchar(20)	No		
moviePosture	blob	Yes	*NULL*	

3. Moviecast table

Column	Type	Null	Default	Links to
mcmovieId	int(11)	No		movie -> movieId
mccastname	varchar(50)	No		

4. Movieweekly table

Column	Type	Null	Default	Links to
mwMovieId	int(11)	No		moviearchive -> movieId
mwNoUser	int(11)	No		
date	date	No		

5. Moviearchive table

Column	Type	Null	Default	Links to
movieId *(Primary)*	int(11)	No		
movieName	varchar(40)	No		
movieDiscription	mediumtext	No		
movieDirector	varchar(30)	No		
movieProducer	varchar(30)	No		
movieGenre	varchar(20)	No		
movieCertificate	varchar(10)	No		
movieDuration	time	No		
movieRating	float	No		
movieReleaseDate	date	No		
movieLanguage	varchar(20)	No		
movieStatus	varchar(20)	No		
moviePosture	blob	Yes	*NULL*	

6. Timmingofmovie table

Column	Type	Null	Default	Links to
slotMovieId *(Primary)*	int(11)	No		movie -> movieId
slotTheaterId *(Primary)*	int(11)	No		screen -> screenTheaterId
slotScreenId *(Primary)*	int(11)	No		screen -> screenId
time1	varchar(8)	No		
time2	varchar(8)	No		
time3	varchar(8)	No		

7. Slotweekly table

Column	Type	Null	Default	Links to
swSlot	varchar(20)	No		
swTheaterId	int(11)	No		theater -> theaterId
swNoUser	int(11)	No		
swdate	date	No		

8. Forslot table

Column	Type	Null	Default	Links to
fsmovieId	int(11)	No		movie -> movieId
fstheaterId	int(11)	No		screen -> screenTheaterId
fsscreenId	int(11)	No		screen -> screenId
fsslot	varchar(30)	No		

9. Theatre table

Column	Type	Null	Default	Links to
theaterId *(Primary)*	int(11)	No		
theaterName	varchar(30)	No		
location	varchar(30)	No		

10. Combo table

Column	Type	Null	Default	Links to
comboId *(Primary)*	int(11)	No		
comboTheaterId *(Primary)*	int(11)	No		theater -> theaterId
comboName	varchar(30)	No		
comboPrice	int(11)	No		
comboQuantity	int(11)	No		
comboDiscription	varchar(150)	No		

11. Screen table

Column	Type	Null	Default	Links to
screenId *(Primary)*	int(11)	No		
screenTheaterId *(Primary)*	int(11)	No		theater -> theaterId
screenCapacity	int(11)	No		
screenType	varchar(10)	No		

12. Seat table

Column	Type	Null	Default	Links to
seatId *(Primary)*	char(2)	No		
seatType	varchar(15)	No		
seatEprice	int(11)	No		

13. Seatbooked table

Column	Type	Null	Default	Links to
buserId	int(11)	Yes	*NULL*	user -> userId
bmovieId *(Primary)*	int(11)	No		movie -> movieId
btheaterId *(Primary)*	int(11)	No		screen -> screenTheaterId
bscreenId *(Primary)*	int(11)	No		screen -> screenId
bseatId *(Primary)*	char(2)	No		seat -> seatId
bslot	varchar(20)	Yes	*NULL*	
paymentStatus	int(11)	No	0	

14. User table

Column	Type	Null	Default	Links to
userId *(Primary)*	int(11)	No		
username	varchar(30)	No		
userPassword	varchar(15)	No		
userFirstName	varchar(30)	No		
userLastName	varchar(30)	No		
userEmail	varchar(30)	No		
userBirthdate	date	No		
userContact	bigint(10)	No		

15. Useraccount table

Column	Type	Null	Default	Links to
accountUserId	int(11)	No		user -> userId
accountId *(Primary)*	int(11)	No		
accountBalance	int(11)	No		
updateDate	datetime	Yes	CURRENT_TIMESTAMP	

16. Usercombobridge table

Column	Type	Null	Default	Links to
ucbUserId	int(11)	No		user -> userId
ucbComboId	int(11)	No		combo -> comboId
ucbTheaterId	int(11)	No		combo -> comboTheaterId
ucbComboQuantity	int(11)	No		
ucbTotalPrice	int(11)	No		
paymentStatus	int(11)	No	0	

17. Usermovie table

Column	Type	Null	Default	Links to
umUserId	int(11)	No		user -> userId
umMovieId	int(11)	No		moviearchive -> movieId
umReview	varchar(200)	Yes	*NULL*	
umRating	float	Yes	*NULL*	

18. Theatertotal table

Column	Type	Null	Default	Links to
ttheaterId	int(11)	No		theater -> theaterId
total	int(11)	No		

STORED PROCEDURES, FUNCTIONS, TRIGGERS, AND EVENTS

1) PROCEDURES:

1.1 Procedure to know the number of viewers movie-wise for daily box office report generation:

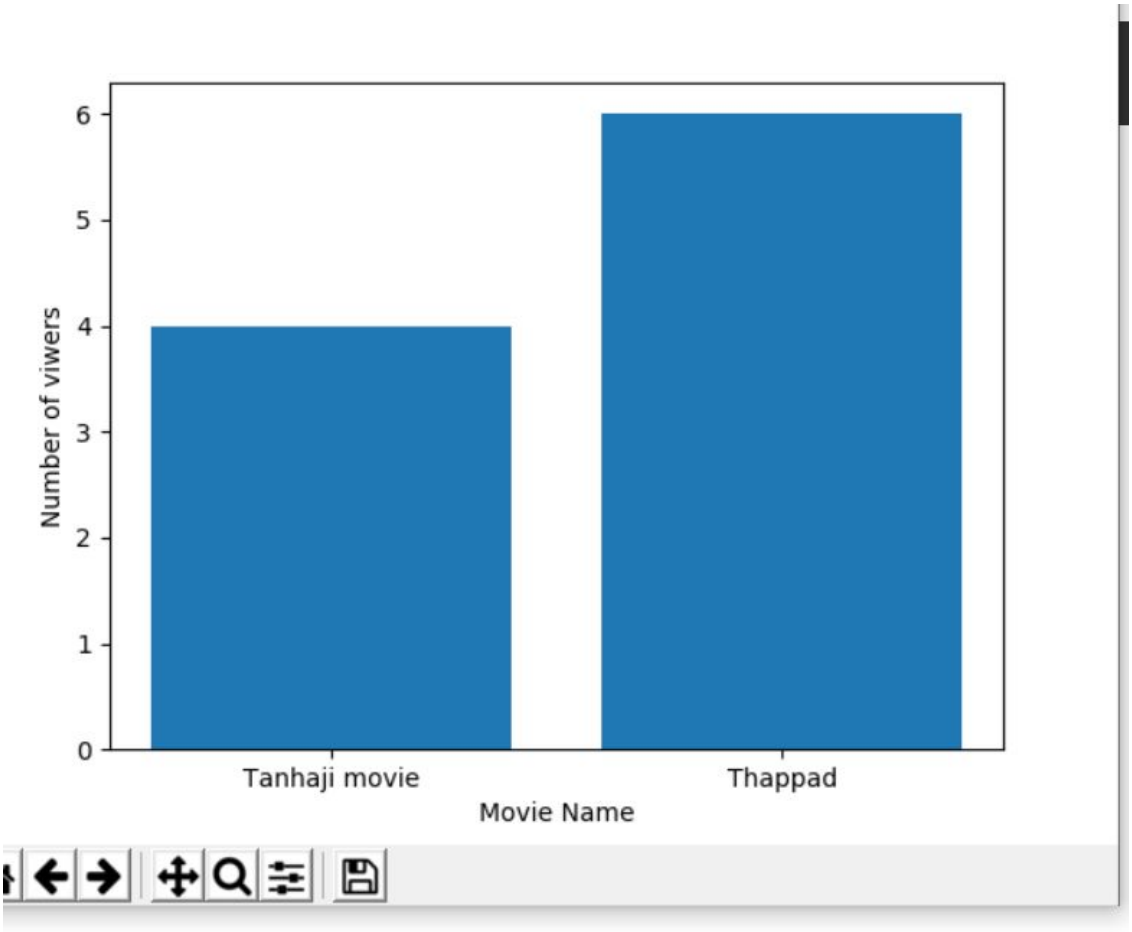
```
CREATE DEFINER=`root`@`localhost` PROCEDURE `noUserMovie`(OUT `name` VARCHAR(500)) NOT DETERMINISTIC NO SQL SQL SECURITY DEFINER
BEGIN
  DECLARE finished INTEGER DEFAULT 0;
  DECLARE movieId INTEGER DEFAULT 0;
  DECLARE mname INTEGER DEFAULT 0;
  DECLARE tp varchar(5000) DEFAULT "";
  DECLARE mid INTEGER DEFAULT 0;
  DECLARE moviename varchar(200) DEFAULT "";
  DECLARE name varchar(5000) DEFAULT "";
declare cur cursor for SELECT bmovieID,count(bseatId) FROM seatbooked WHERE seatbooked.paymentStatus=1 GROUP BY bmovieId ORDER BY bmovieId DESC;

declare cur2 cursor for SELECT movie.movieName FROM movie WHERE movie.movieId=mid;
  -- declare NOT FOUND handler
  DECLARE CONTINUE HANDLER
    FOR NOT FOUND SET finished = 1;

    DELETE FROM movieweekly WHERE date=CURRENT_DATE();
  OPEN cur;
  getm: LOOP
    FETCH cur INTO movieId,mname;
    IF finished = 1 THEN
      LEAVE getm;
    END IF;
```

```
    SET mid=movieId;
    OPEN cur2;
    FETCH cur2 into movieName;
    CLOSE cur2;

    INSERT INTO movieweekly VALUES(movieId,mname,CURRENT_DATE());
    SET tp = CONCAT(movieName,'=',mname,';',tp);
  END LOOP getm;
  SET name=tp;
  CLOSE cur;
  SELECT name;
END
```

Called Procedure :

```
@trail.route('/report', methods=['GET', 'POST'])
def report():
    cur = mysql.connection.cursor()
    cur.callproc('noUserMovie')
    st=cur.fetchall()
    #print(st)
    cur.close()
    l = list(st[0]['name'].split(";"))
    l.pop()
    for i in range(0, len(l)):
        l[i] = list(l[i].split("="))
    #print(l)
    x=['d']*len(l)
    y=[0]*len(l)
    for i in range(len(l)):
        x[i]=l[i][0]
        y[i]=int(l[i][1])

    plt.bar(x, y)
    plt.ylabel('Number of viwers')
    plt.xlabel('Movie Name')
    plt.show()
```

Output:


6
5
4
3
2
1
0
Number of viwers
Tanhaji movie
Thappad
Movie Name

1.2 Procedure to generate weekly report of movie-wise box office collection:

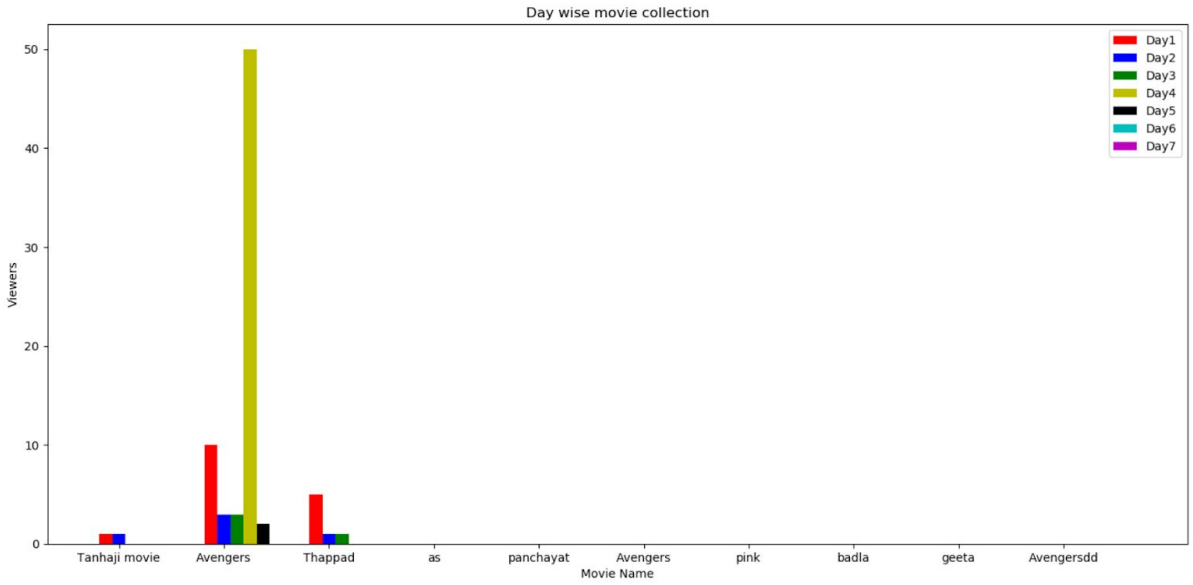
```
CREATE DEFINER=`root`@`localhost` PROCEDURE `movieWeekly`() NOT
DETERMINISTIC NO SQL SQL SECURITY DEFINER
BEGIN
   DECLARE finished INTEGER DEFAULT 0;
   DECLARE movieId INTEGER DEFAULT 0;
   DECLARE mid INTEGER DEFAULT 0;
   DECLARE number INTEGER DEFAULT 0;
   DECLARE moviename varchar(200) DEFAULT "";
   DECLARE tp varchar(5000) DEFAULT "";
   DECLARE name varchar(5000) DEFAULT "";
   DECLARE dat DATE ;
 declare cur1 cursor for  SELECT
moviearchive.movieId,moviearchive.movieName FROM moviearchive ;

declare cur2 cursor for SELECT movieweekly.mwNoUser,movieweekly.date
FROM movieweekly WHERE movieweekly.mwMovieId=mid;

   -- declare NOT FOUND handler
   DECLARE CONTINUE HANDLER
     FOR NOT FOUND SET finished = 1;
   OPEN cur1;
   getm: LOOP
     FETCH cur1 INTO movieId,movieName;
     IF finished = 1 THEN
       LEAVE getm;
     END IF;
     SET tp = CONCAT(tp,movieName,'=');
     SET mid=movieId;

     OPEN cur2;
     getn: LOOP
     FETCH cur2 INTO number,dat;
     IF finished = 1 THEN
       SET finished =0;
       LEAVE getn;
     END IF;
```

```
    SET tp = CONCAT(tp,number,',',dat,';');
    END LOOP getn;
    CLOSE cur2;
    SET tp = CONCAT(tp,'+');
  END LOOP getm;
  CLOSE cur1;
  SET name=tp;
  SELECT name;
END
```

Called Procedure :

```
@trail.route('/report2', methods=['GET', 'POST'])
def report2():
    cur = mysql.connection.cursor()
    cur.callproc('movieWeekly')
    st=cur.fetchall()
    #print(st)
    cur.close()
```

Output:



1.3 Procedure to suggest trending movies to viewers:

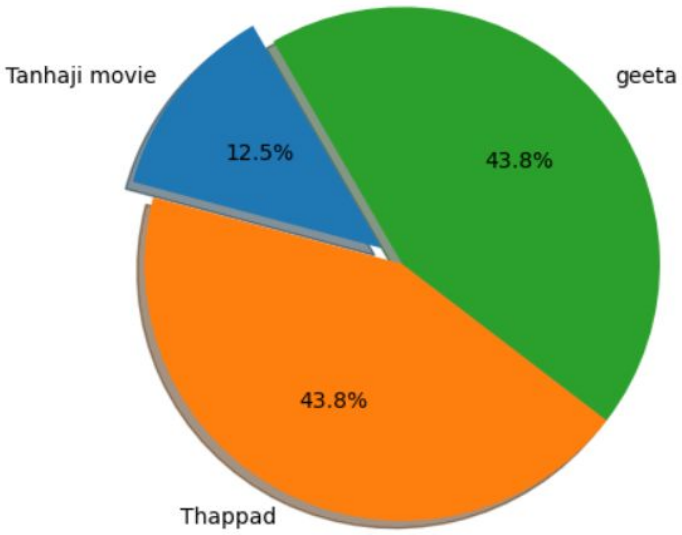
```
DROP PROCEDURE `userSuggestion`; CREATE DEFINER=`root`@`localhost`
PROCEDURE `userSuggestion`() NOT DETERMINISTIC NO SQL SQL SECURITY
DEFINER
BEGIN
  DECLARE finished INTEGER DEFAULT 0;
  DECLARE movieId INTEGER DEFAULT 0;
  DECLARE num INTEGER DEFAULT 0;
  DECLARE mname INTEGER DEFAULT 0;
  DECLARE tp varchar(5000) DEFAULT "";
  DECLARE mid INTEGER DEFAULT 0;
  DECLARE moviename varchar(200) DEFAULT "";
  DECLARE name varchar(5000) DEFAULT "";
declare cur1 cursor for SELECT DISTINCT timmingofmovie.slotMovieId from
timmingofmovie;

declare cur2 cursor for SELECT movie.movieName FROM movie WHERE
movie.movieId=mid;

DECLARE cur3 CURSOR for SELECT SUM(movieweekly.mwNoUser) FROM
movieweekly WHERE mwMovieId=mid group BY movieweekly.mwMovieId;

  -- declare NOT FOUND handler
  DECLARE CONTINUE HANDLER
    FOR NOT FOUND SET finished = 1;
  OPEN cur1;
  getm: LOOP
    FETCH cur1 INTO movieId;
    IF finished = 1 THEN
      LEAVE getm;
    END IF;

    SET mid=movieId;

    OPEN cur2;
    FETCH cur2 into movieName;
    SET tp = CONCAT(tp,movieName,':');
    CLOSE cur2;
```

```
            OPEN cur3;
      FETCH cur3 into num;
      SET tp = CONCAT(tp,num,'+');
      CLOSE cur3;

   END LOOP getm;
   SET name=tp;
   CLOSE cur1;
   SELECT name;
END
```

Called Procedure :

```python
@trail.route('/userSuggestion', methods=['GET', 'POST'])
def userSuggestion():
    cur = mysql.connection.cursor()
    cur.callproc('userSuggestion')
    st=cur.fetchall()
    #print(st)
    cur.close()
```

Output:



1.4 Procedure to print the ticket:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `showTicket`(IN `mid` INT, IN
`tid` INT, IN `sid` INT, IN `uName` VARCHAR(200)) NOT DETERMINISTIC NO SQL
SQL SECURITY DEFINER
BEGIN
  DECLARE finished INTEGER DEFAULT 0;
  DECLARE uid INTEGER DEFAULT 0;
  DECLARE mname INTEGER DEFAULT 0;
  DECLARE total INTEGER DEFAULT 0;
  DECLARE comboName varchar (200) DEFAULT "";
  DECLARE type varchar (50) DEFAULT "";
  DECLARE comboPrice INTEGER DEFAULT  0;
  DECLARE ucbComboQuantity INTEGER DEFAULT 0;
  DECLARE ucbTotalPrice INTEGER DEFAULT 0;
  DECLARE movieName varchar(200) DEFAULT "";
  DECLARE theaterName varchar(200) DEFAULT "";
  DECLARE location varchar(100) DEFAULT "";
  DECLARE slot varchar(20) DEFAULT "";
  DECLARE seatId varchar(2) DEFAULT "";
  DECLARE tp varchar(5000) DEFAULT "";
  DECLARE name varchar(5000) DEFAULT "";

declare cur1 cursor for SELECT movie.movieName from movie where
movie.movieId=mid;

declare cur2 cursor for SELECT theater.theaterName, theater.location from
theater where theater.theaterId=tid;

DECLARE cur3 cursor for SELECT screen.screenType FROM screen where
screen.screenId=sid and screen.screenTheaterId=tid;


DECLARE cur4 cursor for SELECT userId from user where userName=uName;

DECLARE cur5 cursor for SELECT DISTINCT bslot FROM seatbooked,seat WHERE
seatbooked.bseatId=seat.seatId and seatbooked.bmovieId=mid and
seatbooked.btheaterId=tid and seatbooked.bscreenId=sid and
seatbooked.paymentStatus=0 and seatbooked.buserId=uid order by bseatId desc;
```

```
DECLARE cur6 cursor for SELECT bseatId FROM seatbooked,seat WHERE
seatbooked.bseatId=seat.seatId and seatbooked.bmovieId=mid and
seatbooked.btheaterId=tid and seatbooked.bscreenId=sid and
seatbooked.paymentStatus=0 and seatbooked.buserId=uid order by bseatId desc;

DECLARE cur7 cursor for SELECT
combo.comboName,combo.comboPrice,usercombobridge.ucbComboQuantity,us
ercombobridge.ucbTotalPrice FROM combo,usercombobridge WHERE
usercombobridge.paymentStatus=0 and
combo.comboId=usercombobridge.ucbComboId and
combo.comboTheaterId=usercombobridge.ucbTheaterId and
usercombobridge.ucbUserId=uid and usercombobridge.ucbTheaterId=tid;

DECLARE cur8 cursor for SELECT totalPricePay(uid,tid);

  -- declare NOT FOUND handler
  DECLARE CONTINUE HANDLER
    FOR NOT FOUND SET finished = 1;
OPEN cur1;
FETCH cur1 into movieName;
CLOSE cur1;
SET tp = CONCAT(tp,movieName,'+');

OPEN cur2;
FETCH cur2 into theaterName,location;
CLOSE cur2;
SET tp = CONCAT(tp,theaterName,'+',location,'+');

OPEN cur3;
FETCH cur3 into type;
CLOSE cur3;
SET tp = CONCAT(tp,type,'+');

OPEN cur4;
FETCH cur4 into uid;
CLOSE cur4;

OPEN cur5;
FETCH cur5 into slot;
```

```
CLOSE cur5;
SET tp = CONCAT(tp,slot,'+');

OPEN cur8;
FETCH cur8 into total;
CLOSE cur8;
SET tp = CONCAT(tp,total,'+');

OPEN cur6;
 getm: LOOP
     FETCH cur6 INTO seatId;
     IF finished = 1 THEN
        SET finished=0;
       LEAVE getm;
     END IF;
     SET tp = CONCAT(tp,seatId,',');
  END LOOP getm;
CLOSE cur6;
SET tp = CONCAT(tp,'+');

OPEN cur7;
 getm: LOOP
     FETCH cur7 INTO
comboName,comboPrice,ucbComboQuantity,ucbTotalPrice;
     IF finished = 1 THEN
        SET finished=0;
       LEAVE getm;
     END IF;
     SET tp =
CONCAT(tp,comboName,',',comboPrice,',',ucbComboQuantity,',',ucbTotalPrice,';');
  END LOOP getm;
CLOSE cur7;

SET name=tp;
SELECT name;

END
```

Called Procedure :

```
@trail.route('/ticket/<int:movieId>/<int:theaterId>/<int:screenId>/<string:username>', methods=['GET', 'POST'])
def ticket(movieId,theaterId,screenId,username):
    cur = mysql.connection.cursor()
    cur.callproc('showTicket',(movieId,theaterId,screenId,username))
    st = cur.fetchone()
```

Output:

Username: Mr./Mrs.dppawar
Movie Name:Thappad
Date: 2020-04-20
Theater Name:Cineplex,ahmedabad
Show Timings: 8:00
Screen number:2D

For Platinum ticket:140 rupees
For Gold ticket:120 rupees
For Silver ticket:100 rupees

Total Number of seats booked:0

Grand Total: 240

Proceed to pay

1.5 Procedure to deduct the quantity of combo and the amount of ticket from user's e-wallet:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `tableUpdate`(IN `aId` INT, IN
`price` INT, IN `uid` INT, IN `mid` INT, IN `tid` INT, IN `sid` INT) NOT
DETERMINISTIC NO SQL SQL SECURITY DEFINER
BEGIN

  DECLARE finished INTEGER DEFAULT 0;
  DECLARE cid INTEGER DEFAULT 0;
  DECLARE qua INTEGER DEFAULT 0;

declare cur1 cursor for SELECT
usercombobridge.ucbComboId,usercombobridge.ucbComboQuantity from
usercombobridge WHERE usercombobridge.ucbUserId=uid and
usercombobridge.ucbTheaterId=tid and usercombobridge.paymentStatus=0;

  -- declare NOT FOUND handler
  DECLARE CONTINUE HANDLER
    FOR NOT FOUND SET finished = 1;
  OPEN cur1;
  getm: LOOP
    FETCH cur1 INTO cid,qua;

    IF finished = 1 THEN
      LEAVE getm;
    END IF;

   UPDATE combo set combo.comboQuantity=combo.comboQuantity-qua where
combo.comboId=cid and combo.comboTheaterId=tid;
   UPDATE usercombobridge SET usercombobridge.paymentStatus=1 WHERE
usercombobridge.ucbUserId=uid and usercombobridge.ucbComboId=cid and
usercombobridge.ucbTheaterId=tid;

  END LOOP getm;

  CLOSE cur1;
```

INSERT INTO usermovie(usermovie.umUserId,usermovie.umMovieId)
VALUES(uid,mid);

UPDATE seatbooked SET seatbooked.paymentStatus=1 WHERE buserId =uid and seatbooked.paymentStatus=0 and seatbooked.bmovieId=mid and seatbooked.btheaterId=tid and seatbooked.bscreenId=sid;

UPDATE useraccount set useraccount.accountBalance=useraccount.accountBalance-price,useraccount.updateDate=CURRENT_DATE() where useraccount.accountId=aId;
END

Called Procedure :

```
def pdf(accountId,price,movieId, theaterId, screenId, username):
    cur = mysql.connection.cursor()
    cur.execute("SELECT userId from user where userName=%s", [username])
    userId = cur.fetchone()
    cur.callproc('tableUpdate', (accountId,price,userId['userId'],movieId, theaterId, screenId))
    st = cur.fetchone()
    cur.execute("UPDATE useraccount SET `accountBalance`=-10 WHERE useraccount.accountUserId=%s",[userId['userId']])
    mysql.connection.commit()
    cur.close()
```

Output:

Payment and Account Details

Username: Mr./Mrs. dppawar

Account Number: 7

Current Balance: 45320

Total bill amount is:240

Confirm To Pay

2) FUNCTIONS:

2.1 Function to calculate total bill the viewer has to pay:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `totalPricePay`(`userId` INT)
RETURNS INT(11) NOT DETERMINISTIC NO SQL SQL SECURITY DEFINER
BEGIN
    DECLARE done INT DEFAULT 0;
    declare i_myfield int;
    declare i_myfield1 varchar(10);
    declare i_myfield2 int;
    declare tid int;
    declare iter int;
    declare cur1 cursor for SELECT ucbTheaterId, ucbTotalPrice from
usercombobridge WHERE ucbUserId=userId and
usercombobridge.paymentStatus=0;
    declare cur2 cursor for SELECT seatbooked.bseatId, seat.seatEprice FROM
seat,seatbooked WHERE seat.seatId=seatbooked.bseatId AND buserId=userId and
seatbooked.paymentStatus=0;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    set iter = 0;
  open cur2;
    read_loop: loop
      fetch cur2 into i_myfield1,i_myfield2;
      IF done=1 THEN
      set done=0;
        LEAVE read_loop;
      END IF;
      IF (i_myfield1='G1' OR i_myfield1='G2' OR i_myfield1='G3') THEN
      SET iter = iter + 100+i_myfield2;
      END IF;
       IF (i_myfield1='S1' OR i_myfield1='S2' OR i_myfield1='S3') THEN
      SET iter = iter + 100+i_myfield2;
      END IF;
      IF (i_myfield1='P1' OR i_myfield1='P2' OR i_myfield1='P3') THEN
      SET iter = iter + 100+i_myfield2;
      END IF;
    END loop;
```

```
  close cur2;
  open cur1;
    read_loop: loop
      fetch cur1 into tid,i_myfield;
      IF done=1 THEN
        LEAVE read_loop;
      END IF;

      SET iter = iter + i_myfield;
    END loop;

  close cur1;
  INSERT INTO theatertotal VALUES(theaterid,iter);

  return iter;
end
```

Called Function :

This function is called inside another Procedure.

2.2 Function to generate the report of daily collection theatre-wise:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `dailyTheaterVisiter`()
RETURNS VARCHAR(5000) CHARSET latin1 NOT DETERMINISTIC NO SQL SQL
SECURITY DEFINER
BEGIN
  DECLARE finished INTEGER DEFAULT 0;
  DECLARE slot INTEGER DEFAULT 0;
  DECLARE slotname varchar(20) DEFAULT "";
  DECLARE tp varchar(5000) DEFAULT "";
  DECLARE name varchar(5000) DEFAULT "";

declare cur1 cursor for SELECT bslot,count(bseatId) FROM seatbooked WHERE
seatbooked.paymentStatus=1 and seatbooked.btheaterId=1 GROUP BY
seatbooked.bslot ORDER BY bslot;

declare cur2 cursor for SELECT bslot,count(bseatId) FROM seatbooked WHERE
seatbooked.paymentStatus=1 and seatbooked.btheaterId=2 GROUP BY
seatbooked.bslot ORDER BY bslot;
  -- declare NOT FOUND handler

  DECLARE CONTINUE HANDLER
    FOR NOT FOUND SET finished = 1;
 DELETE FROM slotweekly WHERE slotweekly.swdate=CURRENT_DATE();
  OPEN cur1;
  getm: LOOP
    FETCH cur1 INTO slotname,slot;
    IF finished = 1 THEN
      SET finished=0;
       LEAVE getm;
    END IF;
    INSERT INTO slotweekly VALUES(slotname,1,slot,CURRENT_DATE());
    SET tp = CONCAT(tp,slotname,',',slot,';');
  END LOOP getm;
  CLOSE cur1;
  SET tp= CONCAT(tp,'+');
  OPEN cur2;
  getm: LOOP
```

```
    FETCH cur2 INTO slotname,slot;
    IF finished = 1 THEN
      LEAVE getm;
    END IF;
   INSERT INTO slotweekly VALUES(slotname,2,slot,CURRENT_DATE());
    SET tp = CONCAT(tp,slotname,',',slot,';');
  END LOOP getm;
  CLOSE cur2;
  SET name=tp;
  RETURN name;
END
```

Called Function :

```
def report3():
    cur = mysql.connection.cursor()
    cur.execute('SELECT dailyTheaterVisiter() ')
    st=cur.fetchone()
    #print(st)
    cur.close()
    l = list(st['dailyTheaterVisiter()'].split("+"))
```

Output:

Next Day's Visitors(Theatre-wise)

In Theatre : 1

At time: 2:30
Number of visitors are: 2

In Theatre : 2

At time: 9:30
Number of visitors are: 4

2.3 Function to generate total collections of a theatre from day one till current date:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `historyTheaterVisiter`()
RETURNS VARCHAR(5000) CHARSET latin1 NOT DETERMINISTIC NO SQL SQL
SECURITY DEFINER
BEGIN
  DECLARE finished INTEGER DEFAULT 0;
  DECLARE slot INTEGER DEFAULT 0;
  DECLARE slotname varchar(20) DEFAULT "";
  DECLARE d1 date;
  DECLARE tp varchar(5000) DEFAULT "";
  DECLARE name varchar(5000) DEFAULT "";

declare cur1 cursor for SELECT DISTINCT slotweekly.swdate FROM slotweekly
WHERE slotweekly.swTheaterId=1 ORDER BY slotweekly.swdate;

declare cur2 cursor for SELECT slotweekly.swSlot,slotweekly.swNoUser FROM
slotweekly WHERE slotweekly.swTheaterId=1 and slotweekly.swdate=d1;

declare cur3 cursor for SELECT DISTINCT slotweekly.swdate FROM slotweekly
WHERE slotweekly.swTheaterId=2 ORDER BY slotweekly.swdate;

declare cur4 cursor for SELECT slotweekly.swSlot,slotweekly.swNoUser FROM
slotweekly WHERE slotweekly.swTheaterId=2 and slotweekly.swdate=d1;

  -- declare NOT FOUND handler

  DECLARE CONTINUE HANDLER
    FOR NOT FOUND SET finished = 1;
  SET tp = CONCAT(tp,'');
  OPEN cur1;
  getm: LOOP
    FETCH cur1 INTO d1;
    IF finished = 1 THEN
      SET finished=0;
       LEAVE getm;
    END IF;
    SET tp = CONCAT(tp,d1,'/');
```

```
    OPEN cur2;
    getn: LOOP
      FETCH cur2 INTO slotname,slot;
      IF finished = 1 THEN
      SET finished=0;
       LEAVE getn;
      END IF;
       SET tp = CONCAT(tp,slotname,'#',slot,',');
  END LOOP getn;
  CLOSE cur2;
  SET tp = CONCAT(tp,';');
  END LOOP getm;
  CLOSE cur1;
  SET tp= CONCAT(tp,'+');
   OPEN cur3;
  getm: LOOP
    FETCH cur3 INTO d1;
    IF finished = 1 THEN
      SET finished=0;
       LEAVE getm;
    END IF;
    SET tp = CONCAT(tp,d1,'/');
    OPEN cur4;
    getn: LOOP
      FETCH cur4 INTO slotname,slot;
      IF finished = 1 THEN
      SET finished=0;
       LEAVE getn;
      END IF;
       SET tp = CONCAT(tp,slotname,'#',slot,',');
  END LOOP getn;
  CLOSE cur4;
  SET tp = CONCAT(tp,';');
  END LOOP getm;
  CLOSE cur3;
  SET name=tp;
  RETURN name;
END
```

Called Function :

```
def report4():
    cur = mysql.connection.cursor()
    cur.execute('SELECT historyTheaterVisiter() ')
    st=cur.fetchone()
    #print(st)
    cur.close()
```

Output:

Theater's History from day one to now

In Theatre : 1

Date: 2020-04-11

At time: 9:30
Number of visitors are: 10

At time: 2:30
Number of visitors are: 12

Date: 2020-04-12

At time: 9:30
Number of visitors are: 1

At time: 2:30
Number of visitors are: 1

In Theatre : 2

Date: 2020-04-11

At time: 8:00
Number of visitors are: 15

At time: 2:30
Number of visitors are: 20

Date: 2020-04-12

At time: 9:30
Number of visitors are: 1

At time: 8:00
Number of visitors are: 2

Date: 2020-04-17

At time: 9:30
Number of visitors are: 5

2.4 Function to get movie review from viewer:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `movieRate`(`uid` INT)
RETURNS VARCHAR(5000) CHARSET latin1 NOT DETERMINISTIC NO SQL SQL
SECURITY DEFINER

BEGIN
    DECLARE done INT DEFAULT 0;
    declare mid int DEFAULT 0;
    declare mrate float DEFAULT 0;
    declare mname varchar(200);
    declare tp varchar(5000) DEFAULT "";

    declare cur1 cursor for SELECT movie.movieId
,movie.movieName,movieRating FROM movie,usermovie WHERE  umUserId=uid
and movie.movieId=umMovieId and umRating IS null;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

DELETE FROM usercombobridge where paymentStatus=0;
UPDATE seatbooked SET buserId=uid WHERE buserId IS NULL;
   SET tp='no';
  open cur1;
    read_loop: loop
      fetch cur1 into mid,mname,mrate;
      IF done=1 THEN
      set done=0;
        LEAVE read_loop;
      END IF;
     set tp=CONCAT(mid,',',mname,',',mrate) ;
    END loop;
  close cur1;
  return tp;
end
```

Called Function:

```
cur.execute("select userId from user where userName=%s", [username])
id = cur.fetchone()
cur.execute('SELECT movieRate(%s) ',[id['userId']])
res=cur.fetchone()

ind='movieRate({})'.format(id['userId'])
#print(res[ind])
```

Output:

2.5 Function to generate report of total earnings of a theatre from sale of refreshments:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `theaterSale`() RETURNS
VARCHAR(5000) CHARSET latin1 NOT DETERMINISTIC NO SQL SQL SECURITY
DEFINER
BEGIN
  DECLARE finished INTEGER DEFAULT 0;
  DECLARE tsum INTEGER DEFAULT 0;
  DECLARE csum INTEGER DEFAULT 0;
  DECLARE tid INTEGER DEFAULT 0;
  DECLARE thid INTEGER DEFAULT 0;
  DECLARE tp varchar(5000) DEFAULT "";
  DECLARE name varchar(5000) DEFAULT "";

declare cur1 cursor for SELECT sum(ucbTotalPrice) FROM usercombobridge
WHERE usercombobridge.ucbTheaterId=thid GROUP BY ucbTheaterId ORDER BY
ucbTheaterId;

declare cur2 cursor for SELECT theatertotal.ttheaterId,sum(theatertotal.total)
FROM theatertotal GROUP BY theatertotal.ttheaterId;

  -- declare NOT FOUND handler

  DECLARE CONTINUE HANDLER
    FOR NOT FOUND SET finished = 1;
  OPEN cur2;
  getm: LOOP
    FETCH cur2 INTO tid,tsum;
    IF finished = 1 THEN
      SET finished=0;
       LEAVE getm;
    END IF;
    SET tp = CONCAT(tp,tsum,',');
    SET thid=tid;
    OPEN cur1;
    getn: LOOP
    FETCH cur1 INTO csum;
     IF finished = 1 THEN
```

```
      SET finished=0;
       LEAVE getn;
    END IF;
    SET tp = CONCAT(tp,csum);
    END loop;
    CLOSE cur1;
    SET tp = CONCAT(tp,'+');
  END LOOP getm;
  CLOSE cur2;
  SET name=tp;
  RETURN name;
END
```

Called Function:

```
def report5():
    cur = mysql.connection.cursor()
    cur.execute('SELECT theaterSale() ')
    st=cur.fetchone()
    l=list(st['theaterSale()'].split('+'))
    l[0]=list(l[0].split(','))
    l[1] = list(l[1].split(','))
    l.pop()
    tt1=0
    tt2=0
    ct1=0
```

Output:

Today's Total Sale:

Theatre 1:
Total collection:520
In which:
o Collection comes from refreshments!!!

Theatre 2:
total collection:1700
In which:
o Collection comes from refreshments!!!

3) TRIGGERS:

3.1 Trigger to insert a record in the Account table when a record is inserted in the User table:

```
DELIMITER $$
CREATE TRIGGER addAccount
AFTER INSERT
ON user FOR EACH ROW
BEGIN
    INSERT INTO useraccount
    VALUES(new.userId,NULL,50000,NULL);
END$$
DELIMITER ;
```

3.2 Trigger to check whether a movie exists for a given time in a given screen:

```
DELIMITER $$
CREATE TRIGGER checkClash
BEFORE INSERT
ON timmingofmovie FOR EACH ROW
BEGIN
                if (new.time1='True') THEN
        if 'True' in (SELECT time1 from timmingofmovie where
slotTheaterId=new.slotTheaterId
and slotScreenId=new.slotScreenId) THEN
                 signal sqlstate '45000' set message_text ='MyTriggerError:There is
clash with other movie screen timming (9:30)';
        END IF;
    end IF;
    if (new.time2='True') THEN
        if 'True' in (SELECT time2 from timmingofmovie where
slotTheaterId=new.slotTheaterId
and slotScreenId=new.slotScreenId) THEN
                 signal sqlstate '45000' set message_text = 'MyTriggerError:There is
clash with other movie screen timming (2:30)';
        END IF;
    end IF;
    if (new.time3='True') THEN
        if 'True' in (SELECT time3 from timmingofmovie where
slotTheaterId=new.slotTheaterId
and slotScreenId=new.slotScreenId) THEN
                 signal sqlstate '45000' set message_text = 'MyTriggerError:There is
clash with other movie screen timming (8:00)';
        END IF;
    end IF;
END$$
DELIMITER ;
```

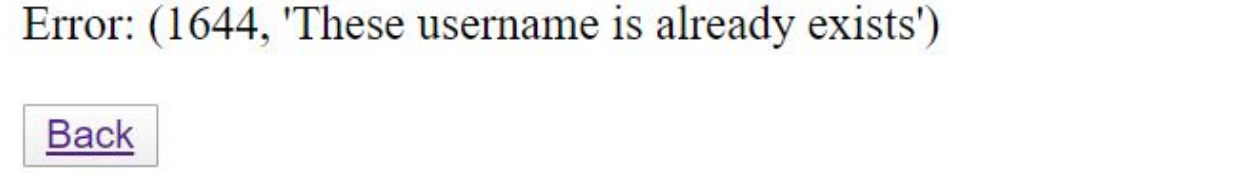
3.3 Trigger to check whether the e-wallet has enough balance to pay for the tickets or not:

```
DELIMITER $$
CREATE TRIGGER `accountDeduct`
BEFORE UPDATE
ON `useraccount`
FOR EACH ROW
BEGIN
	if(new.accountBalance<0) THEN
		signal sqlstate '46000' set message_text ='MyTriggerError:You do not have enough balance in our account.';
	 END IF;
END$$
DELIMITER ;
```

3.4 Trigger to check that the username selected by you is unique:

```
CREATE TRIGGER `checkDuplicateUsername` BEFORE INSERT ON `user`
FOR EACH ROW
BEGIN
        if (EXISTS( SELECT * from user where user.username=new.username))
          THEN
                signal sqlstate '47000' set message_text ='These username is already
exists';
         END IF;
END;
```

```
flag = 0
error = ""
try:
    cur = mysql.connection.cursor()
    cur.execute("INSERT INTO user VALUES (NULL, %s, %s, %s, %s, %s, %s, %s)",
                ( userName, userPassword, userFirstName, userLastName, userEmail, userBirthdate, userContact))
    mysql.connection.commit()
    cur.close()
except Exception as e:
    flag = 1
    error = e

if flag == 1:
    return  render_template("usernameExcept.html",error=error,movieId=movieId,theaterId=theaterId,screenId=screenId)
else:
    return redirect(url_for('userLogin',movieId=movieId,theaterId=theaterId,screenId=screenId))
```



3.5 Trigger to delete the combo details from the child table before deleting them from the main table:

```
DROP TRIGGER IF EXISTS `combochild`;
CREATE DEFINER=`root`@`localhost` TRIGGER `combochild`
BEFORE DELETE ON `combo`
FOR EACH ROW
begin
        DELETE FROM usercombobridge WHERE
        usercombobridge.ucbComboId=old.comboId and
        usercombobridge.ucbTheaterId=old.comboTheaterId;
END;
```

3.6 Trigger to add off-screened movies in the archive table:

```
DROP TRIGGER IF EXISTS `insertInArchive`;
CREATE DEFINER=`root`@`localhost` TRIGGER `insertInArchive`
        AFTER INSERT ON `movie`
        FOR EACH ROW
begin
INSERT INTO moviearchive
VALUES(new.movieId,new.movieName,new.movieDiscription,new.movieDirecto
r,new.movieProducer,new.movieGenre,new.movieCertificate,new.movieDuratio
n,new.movieRating,new.movieReleaseDate,new.movieLanguage,new.movieStat
us,NULL);
END;
```

3.7 Trigger to delete movie details from the child tables before adding the movie to archive table:

```
DROP TRIGGER IF EXISTS `moviedelete1`;
CREATE DEFINER=`root`@`localhost` TRIGGER `moviedelete1` BEFORE DELETE
ON `movie` FOR EACH ROW begin DELETE FROM forslot WHERE
forslot.fsmovieId=old.movieId; end

DROP TRIGGER IF EXISTS `moviedelete2`;
CREATE DEFINER=`root`@`localhost` TRIGGER `moviedelete2` BEFORE DELETE
ON `movie` FOR EACH ROW DELETE FROM timmingofmovie WHERE
timmingofmovie.slotMovieId=old.movieId;

DROP TRIGGER IF EXISTS `moviedelete3`;
CREATE DEFINER=`root`@`localhost` TRIGGER `moviedelete3` BEFORE DELETE
ON `movie` FOR EACH ROW DELETE FROM seatbooked WHERE
seatbooked.bmovieId=old.movieid;
```

3.8 Trigger to check whether there is no primary key violation in timming of movie table:

CREATE TRIGGER `checkPrimaryConstraintSlot` BEFORE INSERT ON `timmingofmovie` FOR EACH ROW BEGIN if (EXISTS(SELECT * from timmingofmovie where timmingofmovie.slotMovieId=new.slotMovieId and timmingofmovie.slotTheaterId=new.slotTheaterId and timmingofmovie.slotScreenId=new.slotScreenId)) THEN signal sqlstate '48000' set message_text ='You have already inserted record for same movie in same screen of theater'; END IF;
END

```
flag=0
err=""
try:
    cur = mysql.connection.cursor()
    cur.execute("INSERT INTO timmingofmovie VALUES (%s, %s, %s, %s, %s, %s)",(movieId,slotTheaterId,slotScreenId,time1,time2,time3))
    mysql.connection.commit()
    cur.close()
except Exception as e:
    err=e
    flag=1
if flag==0:
    return redirect(url_for('slotAdd',movieId=movieId))
else:
    return  render_template("slotAddExcep.html",error=err,movieId=movieId)
```

Error: (1644, 'You have already inserted record for same movie in same screen of theater')

Back

3.9 Check combo table’s primary key

DROP TRIGGER IF EXISTS `checkPrimaryConstraintCombo`;CREATE DEFINER=`root`@`localhost` TRIGGER `checkPrimaryConstraintCombo` BEFORE INSERT ON `combo` FOR EACH ROW BEGIN if (EXISTS(SELECT * from combo where combo.comboId=new.comboId and combo.comboTheaterId=new.comboTheaterId)) THEN signal sqlstate '49000' set message_text ='Already combo exists having same comboId and theaterId'; END IF;
END

```
flag=0
error=""
try:
    cur = mysql.connection.cursor()
    cur.execute("INSERT INTO combo VALUES ( %s, %s, %s, %s, %s, %s)",( comboId,comboTheaterId ,comboName,comboPrice ,comboQuantity , comboDiscription ))
    mysql.connection.commit()
    cur.close()
except Exception as e:
    flag=1
    error=e
if flag==1:
    return  render_template("comboExcept.html",error=error)
else:
    return redirect(url_for('dash'))
```

Error: (1644, 'Already combo exists having same comboId and theaterId')

Back

4) EVENTS

4.1 Event to delete the data of booked seats everyday:

CREATE DEFINER=`root`@`localhost` EVENT `deleteSeatBookedData` ON SCHEDULE EVERY 1 DAY STARTS '2020-04-17 00:00:00' ENDS '2020-04-30 00:00:00' ON COMPLETION NOT PRESERVE ENABLE DO DELETE FROM seatbooked WHERE 1

4.2 Event to update movie history every day:

CREATE DEFINER=`root`@`localhost` EVENT `updatetableHistory` ON SCHEDULE EVERY 1 DAY STARTS '2020-04-17 00:00:00' ENDS '2020-04-30 00:00:00' ON COMPLETION NOT PRESERVE ENABLE DO CALL `noUserMovie`();

4.3 Event to update theatre history every day:

CREATE EVENT `historyUpdate` ON SCHEDULE EVERY 1 DAY STARTS '2020-04-17 00:00:00.000000' ENDS '2020-04-18 00:00:00.000000' ON COMPLETION NOT PRESERVE ENABLE DO SELECT `dailyTheaterVisiter`() AS `dailyTheaterVisiter`;

4.4 Event to delete movies after 1 week:

CREATE EVENT `movieSeven` ON SCHEDULE EVERY 7 DAY STARTS '2020-04-10 00:00:00.000000' ENDS '2020-04-30 00:00:00.000000' ON COMPLETION NOT PRESERVE ENABLE DO DELETE FROM movieweekly WHERE 1

LIMITATIONS

- Here, the data about the theatres is limited to one city only.(i.e. ahmedabad)
- Each theatre is considered to have 2 screens limiting the number of seats and also has few combo varieties.
- There are limited time slots in which movies are available in the theatres.
- The report consists of weekly data and analysis.
- Currently there are options of online wallet and offline payment only.

FUTURE WORK

- The project can be expanded to have theatres across various cities and with each theatre having multiple screens.
- The user can be given multiple options for a seat as well as combo.
- There can be monthly analysis of the box office collection.
- The user can be provided with other payment options like net banking.
- Multiple logins or ticket booking can be handled.

REFERENCES

- **For Python (Flask Framework):**
  - For learning Flask framework:
    + https://www.youtube.com/playlist?list=PLillGF-RfqbbbPz6GSEM9hLQObuQjNoj
    + http://www.blog.pythonlibrary.org/2017/12/14/flask-101-adding-editing-and-displaying-data/
  - For Web Forms:
    + https://pythonspot.com/flask-web-forms/
    + https://wtforms.readthedocs.io/en/stable/fields.html#wtforms.fields.DateTimeField
  - For Checkbox:
    + https://gist.github.com/doobeh/4668212
  - For Error handling:
    + https://stackoverflow.com/questions/5836623/getting-lock-wait-timeout-exceeded-try-restarting-transaction-even-though-im
  - For Jinja template:
    + https://overiq.com/flask-101/basics-of-jinja-template-language/
  - For Flask and MySQL connection:
    + https://pynative.com/python-mysql-execute-stored-procedure/
  - For executing triggers on flask frontend:
    + https://dev.mysql.com/doc/connector-python/en/connector-python-api-errors-error.html
  - For generating graphs:
    + https://www.tutorialspoint.com/matplotlib/matplotlib_bar_plot.htm
- **For MySQL Database:**
  - For cursors:
    + https://www.mysqltutorial.org/mysql-cursor/
- **Basic Idea for project:**
  - https://in.bookmyshow.com/