Ross Bollinger

Dr. M. Owrang

25 April 2019

CSC-434

Theatre Database SQL Code

**Create Database**

Code

create database if not exists Theatre;

use Theatre;

create table if not exists Season

(

play\_title varchar(50) not null primary key,

playwright\_name varchar(50),

opening\_date date,

closing\_date date,

theater\_name varchar(50) not null

);

create table if not exists Theater

(

name varchar(50) not null primary key,

street varchar(15),

max\_seats integer(4)

);

create table if not exists Audience\_member

(

ticket\_number varchar(12) not null primary key,

confirmation\_number varchar(13) not null,

first\_name varchar(25),

last\_name varchar(25)

);

create table if not exists Ticket

(

ticket\_number varchar(12) not null primary key,

seat\_number varchar(5),

price integer(3),

seat\_section varchar(50),

theater\_name varchar(50) not null,

play\_title varchar(50) not null

);

create table if not exists Creative\_team

(

name varchar(50) not null,

job\_title varchar(50),

salary\_per\_hour integer(5),

play\_title varchar(50) not null

);

create table if not exists Cast

(

role varchar(50) not null,

actor\_name varchar(50) not null,

play\_title varchar(50) not null

);

create table if not exists Actor

(

name varchar(50) not null primary key,

salary\_per\_hour integer(5),

equity\_status varchar(10)

);

create table if not exists Staff

(

ID varchar(7) not null primary key,

first\_name varchar(25),

last\_name varchar(25),

job\_title varchar(50),

salary\_per\_hour integer(5),

theater\_name varchar(50) not null

);

insert into Season values("Guys and Dolls", "Frank Loesser", '2018-08-30', '2018-11-04', "Eisenhower Theater");

insert into Season values("Harvey", "Mary Chase", '2018-10-25', '2018-12-09', "Kreeger Theater");

insert into Season values("Hand to God", "Robert Askins", '2019-01-17', '2019-03-17', "Studio Theater");

insert into Season values("Cat on a Hot Tin Roof", "Tennessee Williams", '2019-05-02', '2019-05-29', "Sidney Harman Hall");

select \* from Season;

insert into Theater values("Eisenhower Theater", "F Street", 1161);

insert into Theater values("Kreeger Theater", "Sixth", 514);

insert into Theater values("Studio Theater", "Fourteenth", 200);

insert into Theater values("Sidney Harman Hall", "F Street", 774);

select \* from Theater;

insert into Audience\_member values("T65532926857", "11I1AYJNWDMXL", "Tyson", "Bristol");

insert into Audience\_member values("T12194953219", "K82ATILX5JEPI", "Bettie", "Mann");

insert into Audience\_member values("T53466048953", "HDV7QRFTNXU0A", "Kimmie", "Burnham");

insert into Audience\_member values("T52583813549", "V9YXB5GLG3706", "Jordana", "York");

insert into Audience\_member values("T60301255854", "2OQ1RC6UOLT00", "Allana", "Brook");

insert into Audience\_member values("T42925458325", "W5KJEBIJQGDEB", "Judi", "Larson");

insert into Audience\_member values("T11878498645", "K0ULD7N2C1RCM", "John", "Groves");

select \* from Audience\_member;

insert into Ticket values("T65532926857", "G107", 102, "Orchestra", "Kreeger Theater", "Harvey");

insert into Ticket values("T60596108598", "H101", 179, "Orchestra", "Eisenhower Theater", "Guys and Dolls");

insert into Ticket values("T12194953219", "R110", 92, "Orchestra Grand Tier", "Sidney Harman Hall", "Cat on a Hot Tin Roof");

insert into Ticket values("T53466048953", "A13", 109, "Balcony", "Eisenhower Theater", "Guys and Dolls");

insert into Ticket values("T52583813549", "M20", 179, "Orchestra", "Eisenhower Theater", "Guys and Dolls");

insert into Ticket values("T45050502323", "AA202", 45, "Premium", "Studio Theater", "Hand to God");

insert into Ticket values("T60301255854", "C104", 20, "Zone A", "Studio Theater", "Hand to God");

insert into Ticket values("T42925458325", "AA12", 102, "Balcony", "Kreeger Theater", "Harvey");

insert into Ticket values("T11878498645", "D121", 92, "Orchestra Front", "Sidney Harman Hall", "Cat on a Hot Tin Roof");

select \* from Ticket;

insert into Creative\_team values("Holly Stevens", "Director", 30, "Hand to God");

insert into Creative\_team values("Mel Caulfield", "Costume Designer", 20, "Guys and Dolls");

insert into Creative\_team values("Peter Blakeley", "Director", 50, "Cat on a Hot Tin Roof");

insert into Creative\_team values("Caleigh Glass", "Lighting Designer", 46, "Guys and Dolls");

insert into Creative\_team values("Milo Sanders", "Set Designer", 25, "Harvey");

select \* from Creative\_team;

insert into Cast values("Nathan", "Alex Brightman", "Guys and Dolls");

insert into Cast values("Sky", "Armie Hammer", "Guys and Dolls");

insert into Cast values("Elwood", "Paul Schneider", "Harvey");

insert into Cast values("Veta", "Janet McTeer", "Harvey");

insert into Cast values("Jason", "Steven Boyer", "Hand to God");

insert into Cast values("Margery", "Geneva Carr", "Hand to God");

insert into Cast values("Brick", "Jack O'Connell", "Cat on a Hot Tin Roof");

insert into Cast values("Maggie", "Sienna Miller", "Cat on a Hot Tin Roof");

select \* from Cast;

insert into Actor values("Alex Brightman", 42, "equity");

insert into Actor values("Armie Hammer", 42, "equity");

insert into Actor values("Paul Schneider", 35, "nonequity");

insert into Actor values("Janet McTeer", 35, "nonequity");

insert into Actor values("Steven Boyer", 35, "equity");

insert into Actor values("Geneva Carr", 35, "nonequity");

insert into Actor values("Jack O'Connell", 37, "equity");

insert into Actor values("Sienna Miller", 37, "nonequity");

select \* from Actor;

insert into Staff values("4874979", "John", "Starr", "Set Builder", 30, "Eisenhower Theater");

insert into Staff values("1600277", "Peter", "Mikhail", "Set Painter", 30, "Kreeger Theater");

insert into Staff values("3016914", "Alex", "Damman", "Master Electrician", 60, "Eisenhower Theater");

insert into Staff values("5881833", "Erin", "Sullivan", "Lighting Coordinator", 45, "Studio Theater");

insert into Staff values("9784995", "Jack", "Johnson", "Set Builder", 33, "Sidney Harman Hall");

insert into Staff values("4397949", "Zack", "Jordan", "Usher", 9, "Studio Theater");

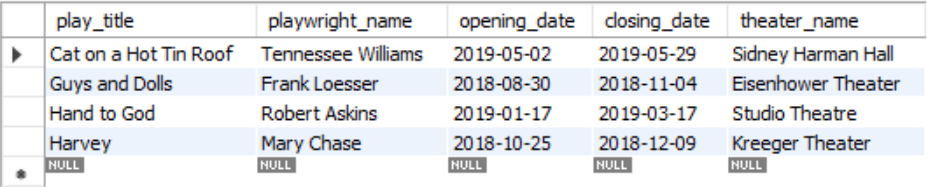
insert into Staff values("6784866", "Edward", "Hope", "Lighting Coordinator", 50, "Sidney Harman Hall");

insert into Staff values("1201393", "Randy", "Howenstein", "Assistant Electrician", 46, "Kreeger Theater");

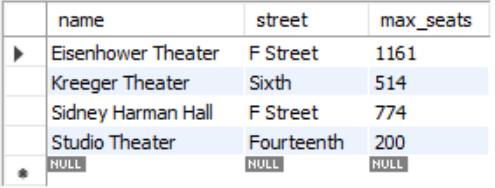
select \* from Staff;

Output

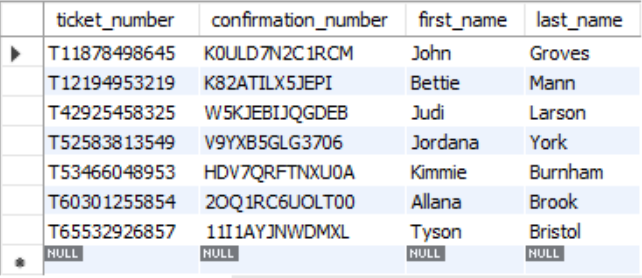
Season:



Theater:



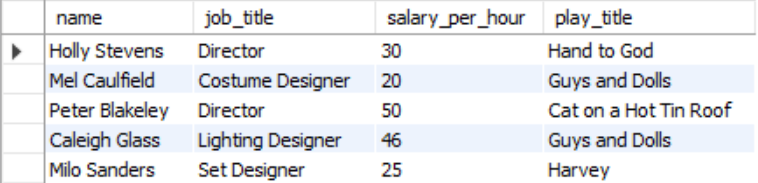
Audience\_member:



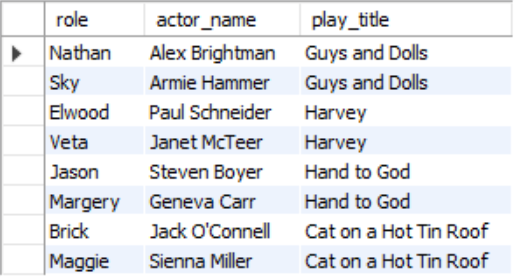
Ticket:



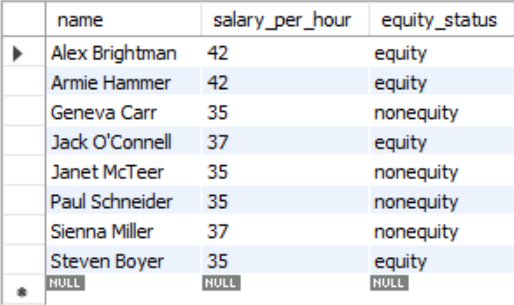
Creative\_team:



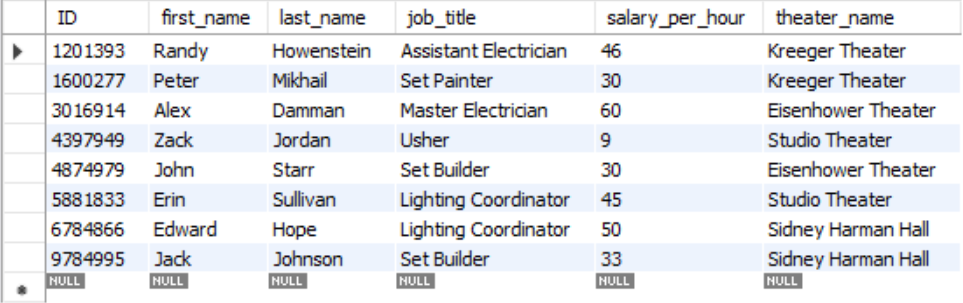
Cast:



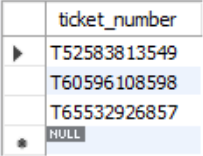
Actor:



Staff:



1. **Queries**

* **SELECT:**
* select involving one/more conditions in Where Clause
* Select all ticket numbers where price is greater than or equal to $100 and the seat section says "Orchestra".
  + Code:   
    select ticket\_number from Ticket   
    where price >= 100   
    and seat\_section like '%Orchestra%';
  + Output:  
    
* select with aggregate functions (i.e., SUM,MIN,MAX,AVG,COUNT)
* SUM and Nested SELECT
* Find the sum of all hourly salaries for all Actors working at the Kreeger Theater.
  + Code:

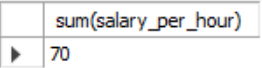
select sum(salary\_per\_hour) from Actor

where name in

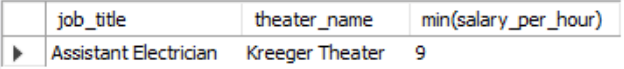
(select actor\_name from Cast where play\_title in

(select play\_title from Season

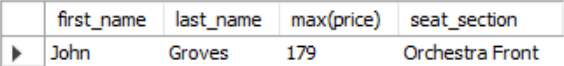
where theater\_name = 'Kreeger Theater'));

* + Output:  
    
* MIN
* Find the job title, theater name, and salary of the staff job with the lowest hourly pay.
  + Code:

select job\_title, theater\_name, min(salary\_per\_hour) from Staff;

* + Output:  
    
* MAX
* Find the full name of the audience member, price, and seat section of the most expensive ticket purchase for any of the productions in the season.
  + Code:

select first\_name, last\_name, max(price), seat\_section from Audience\_member, Ticket;

* + Output:  
    
* AVG
* Find the average price of a ticket at the Eisenhower Theater.
  + Code:

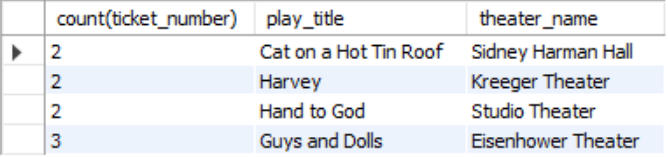
select avg(price) from Ticket

where theater\_name = 'Eisenhower Theater';

* + Output:  
    
* COUNT and GROUP BY
* Find the number of tickets purchase per production. Display count, play title, and theater name.
  + Code:

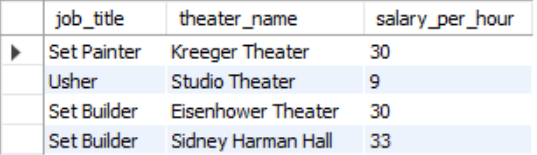
select count(ticket\_number), play\_title, theater\_name from Ticket

group by play\_title;

* + Output:  
    
* Select with Having, Group By (already completed), Order By clause
* HAVING
* Find the jobs, theater names, and salaries associated with staff members with salaries below $40/hr.
  + Code:

select job\_title, theater\_name, salary\_per\_hour from Staff

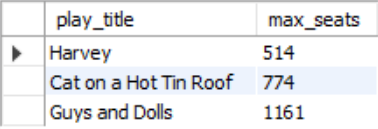
having salary\_per\_hour < 40;

* + Output:  
    
* ORDER BY
* Find the titles of plays grouped by the max seats of the theaters at which they are performed.
  + Code:

select play\_title, max\_seats from Season, Theater

where Season.theater\_name = Theater.name

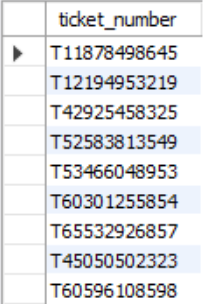
order by max\_seats;

* + Output:  
    
* Select involving the Union operation
* Find all ticket numbers recorded with audience members. Not all ticket numbers are recorded into the audience\_member table, so this is a way to double check if all tickets are logged.
  + Code:

select ticket\_number from Audience\_member

union

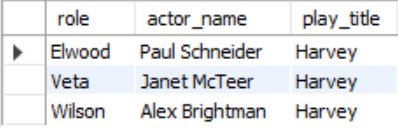
select ticket\_number from Ticket;

* + Output:  
    
* **INSERT:**
* Insert one tuple into a table
* Alex Brightman will also be playing the role of Wilson in Harvey, add him to that cast list.
  + Code:

insert into Cast values("Wilson", "Alex Brightman", "Harvey");

select \* from Cast

where play\_title = 'Harvey';

* + Output:  
    
* Insert a set of tuples (by using another select statement)/insert involving two tables
* Create a separate table and fill it with only the info about equity actors who are in

the musical Guys and Dolls. Fill specifically with the equity status and name of the actor.

* + Code:

create table if not exists G\_and\_D\_Equity(

actor\_name varchar(50) not null primary key,

equity\_status varchar(10)

);

insert into G\_and\_D\_Equity

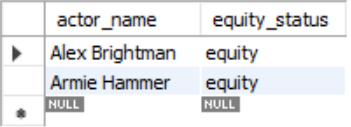
select name, equity\_status from Actor

where equity\_status = 'equity'

and name in

(select actor\_name from Cast where play\_title = 'Guys and Dolls');

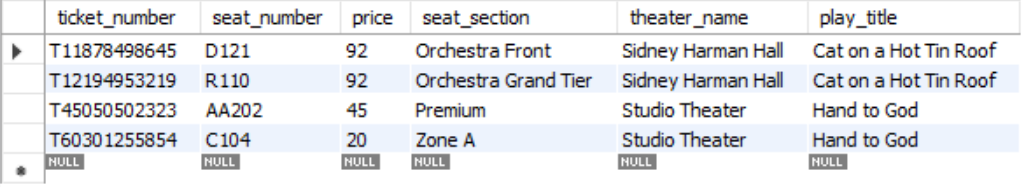
select \* from G\_and\_D\_Equity;

* + Output:  
    
* **DELETE:**
* Delete one tuple or a set of tuples: from one table, from multiple tables
* Delete all tickets that cost above $100.
  + Code:

delete from Ticket

where price > 100;

select \* from Ticket;

* + Output:  
    
* **UPDATE:**
* Update one tuple or a set of tuples: from one table, from multiple tables
* Add a bonus of 5% to actors with salaries below $40/hr.
  + Code:

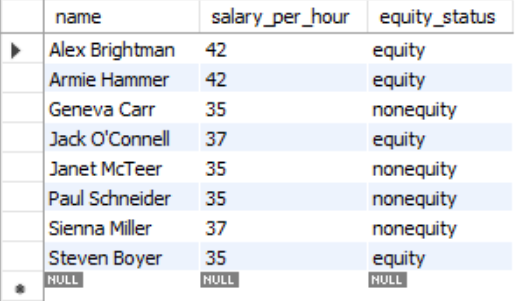
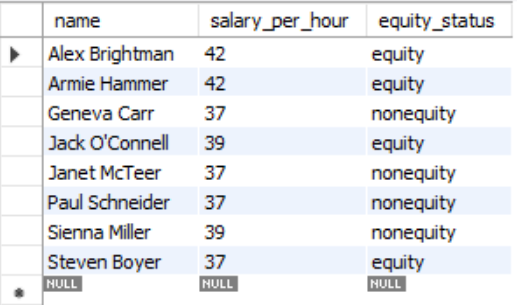
select \* from Actor;

update Actor

set salary\_per\_hour = (salary\_per\_hour \* .05) + salary\_per\_hour

where salary\_per\_hour < 40;

select \* from Actor;

* + Output:  
    Before:  
      
    After:  
    
* **CREATE VIEW:**
* Create a view of all ticket info with audience member names.
  + Code:

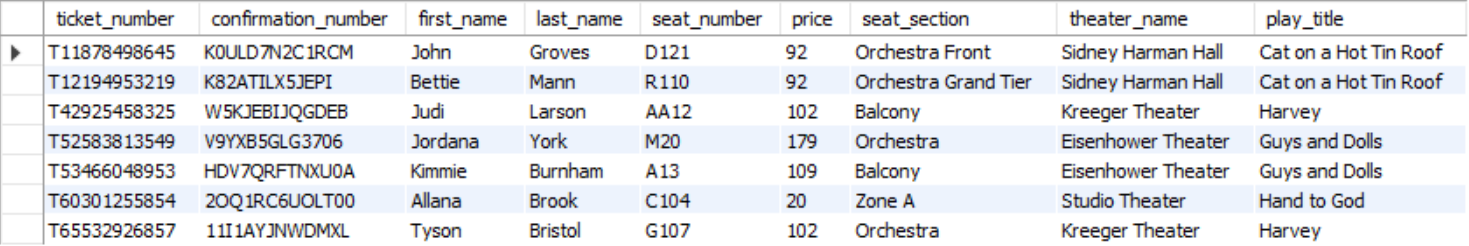
create view allTicketInfo

as select Audience\_member.ticket\_number, confirmation\_number, first\_name, last\_name, seat\_number, price, seat\_section, theater\_name, play\_title

from Audience\_member, Ticket

where Audience\_member.ticket\_number = Ticket.ticket\_number;

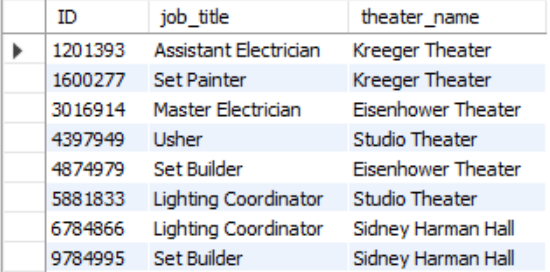
select \* from allTicketInfo;

* + Output:  
    
* Create a new view of just employee IDs and job titles from staff.
  + Code:

create view simpleStaff

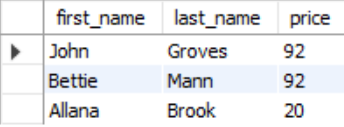
as select ID, job\_title, theater\_name from Staff;

select \* from simpleStaff;

* + Output:  
    
* Operate on View
* SELECT
* Find the names of all audience members who paid under $100 for a ticket.
  + Code:

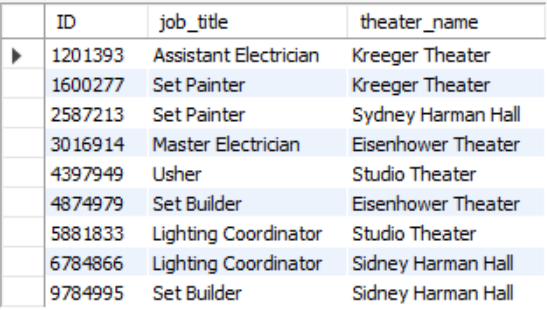
select first\_name, last\_name, price from allTicketInfo

where price < 100;

* + Output:  
    
* INSERT
* Insert a new item into the Employee view.
  + Code:

insert into simpleStaff values ("2587213", "Set Painter", "Sydney Harman Hall");

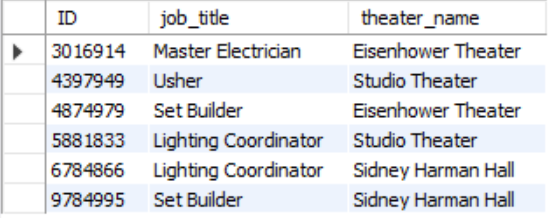
select \* from simpleStaff;

* + Output:  
    
* DELETE
* Delete all staff info of workers at Kreeger Theater using the Staff View.
  + Code:

delete from simpleStaff

where theater\_name = 'Kreeger Theater';

select \* from simpleStaff;

* + Output:  
    
* UPDATE
* All employees at the Studio Theater are getting laid off. They moved to the Eisenhower Theater.
  + Code:

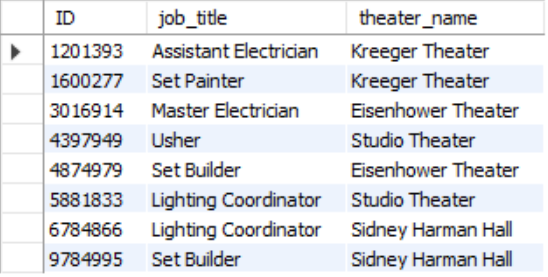
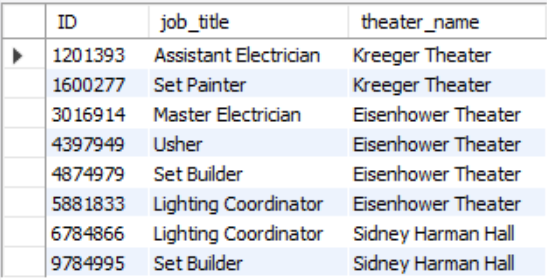
select \* from simpleStaff;

update simpleStaff

set theater\_name = 'Eisenhower Theater'

where theater\_name = 'Studio Theater';

select \* from simpleStaff;

* + Output:  
    Before:  
      
    After:  
    

1. **Triggers**

* Enforcing referential integrity
* Create a trigger that ensures that no duplicate entries will be inserted or updated on Cast.
  + Code:

delimiter $$

create trigger no\_Cast\_Duplicates

before insert on Cast

for each row

begin

if (exists(select 1 from Cast where role = new.role

and actor\_name = new.actor\_name

and play\_title = new.play\_title))

then signal sqlstate value '45000' set message\_text = 'INSERT failed due to duplicate entry';

end if;

end$$

delimiter ;

insert into Cast values("Elwood", "Paul Schneider", "Harvey");

* + Output:  
    
* Enforcing attribute domain constraints
* Create 3 triggers that all ensure that all inserts on Ticket, Audience\_member, and Staff follow the number of digits per key.
  + Code:

delimiter $$

create trigger ticket\_Num\_Limit

before insert on Ticket

for each row

begin

if (exists(select 1 from Ticket where length(ticket\_number) <>  
 length(new.ticket\_number)))

then signal sqlstate value '45000' set message\_text = 'INSERT failed due to incorrect length of ticket number';

end if;

end$$

delimiter ;

insert into Ticket values("T655329268", "G107", 102, "Orchestra", "Kreeger Theater", "Harvey");

delimiter $$

create trigger aud\_Num\_Limit

before insert on Audience\_member

for each row

begin

if (exists(select 1 from Audience\_member where length(ticket\_number)<> length(new.ticket\_number)))

then signal sqlstate value '45000' set message\_text = 'INSERT failed due to incorrect length of ticket number';

end if;

end$$

delimiter ;

insert into Audience\_member values("T655329268", "11I1AYJNWDMXL", "Tyson", "Bristol");

delimiter $$

create trigger staff\_Num\_Limit

before insert on Staff

for each row

begin

if (exists(select 1 from Staff where length(ID) <> length(new.ID)))

then signal sqlstate value '45000' set message\_text =  
'INSERT failed due to incorrect length of ID number';

end if;

end$$

delimiter ;

insert into Staff values("678486", "Edward", "Hope", "Lighting Coordinator", 50, "Sidney Harman Hall");

* + Output:  
    Ticket:  
      
    Audience\_member:  
      
    Staff:  
    
* Creating a database log
* Create a trigger that keeps track of which seats (number, section, and theater) are filled in each theater on insert into ticket.
  + Code:

delimiter $$

create trigger seat\_Logger

after insert on Ticket

for each row

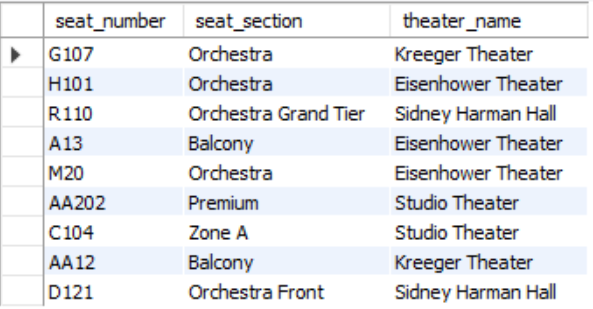
begin

insert into seat\_Log values(new.seat\_number, new.seat\_section, new.theater\_name);

end$$;

delimiter ;

select \* from Seat\_log;

* + Output:  
    
* Gathering statistics
* Create a trigger that keeps track of the average staff member salary after insert into Staff.
  + Code:

create table if not exists Staff\_sal\_avg (theater\_name varchar(50), avg\_salary integer(4));

insert into Staff\_sal\_avg values("Eisenhower Theater", 0);

insert into Staff\_sal\_avg values("Kreeger Theater", 0);

insert into Staff\_sal\_avg values("Studio Theater", 0);

insert into Staff\_sal\_avg values("Sidney Harman Hall", 0);

delimiter $$

create Trigger salary\_Averager

after insert on Staff

for each row

begin

update Staff\_sal\_avg

set avg\_salary = (select avg(salary\_per\_hour) from Staff

where Staff.theater\_name = new.theater\_name)

where theater\_name = new.theater\_name;

end$$

delimiter ;

select \* from Staff\_sal\_avg;

* + Output:  
    