

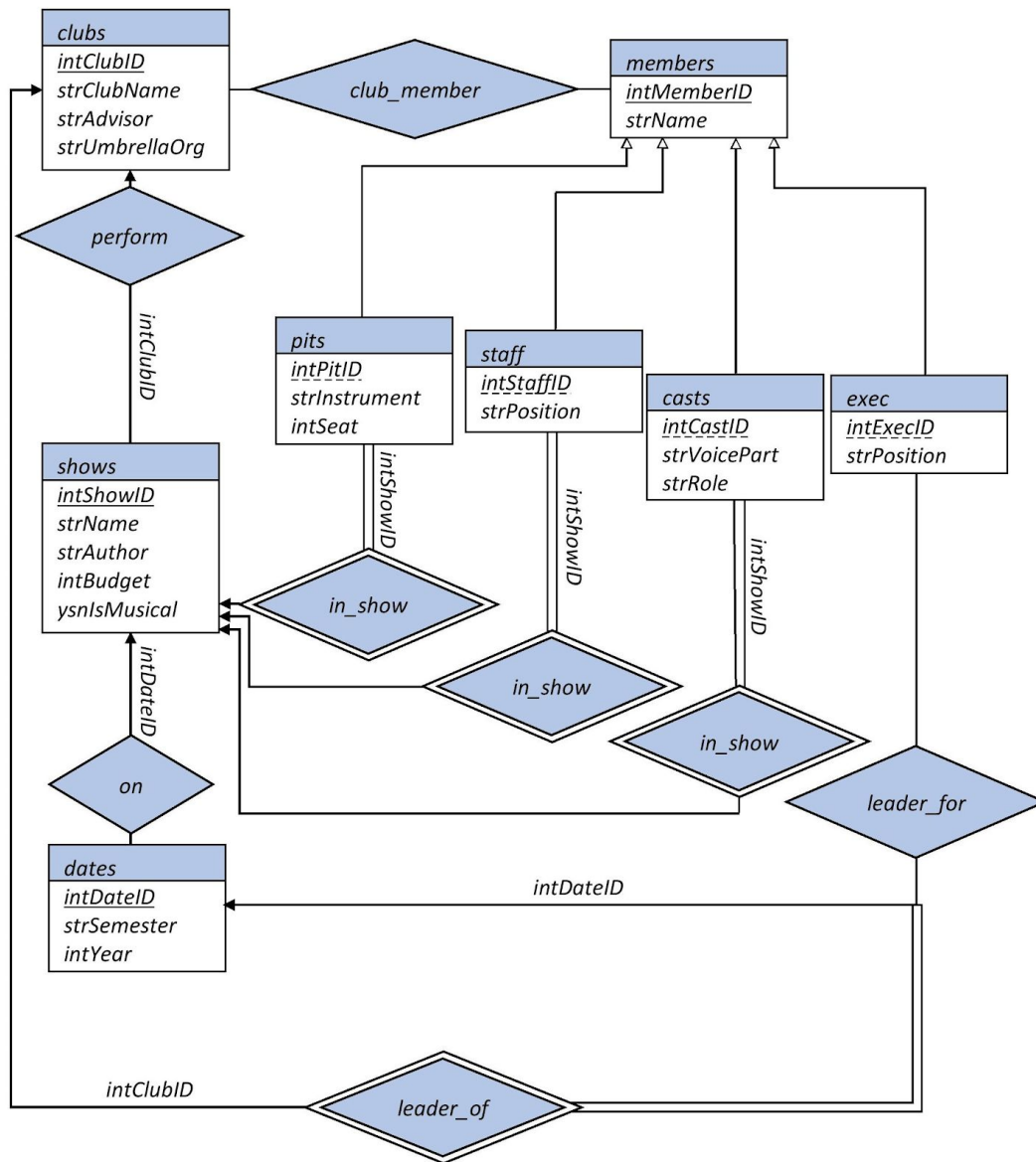
**Name:** Theater Production History

**Team Members:** Ryan Arvizu, Giuliana Conte, Erik Espinosa, Nicholas Charles

**Application Background & Use:** Most of our team participates in theater clubs on campus, and we realized there is no centralized history of all the shows that the clubs have done. Clubs and shows contain a plethora of data, the most significant of which being cast members, pit members, executive boards, production staff, and dates. Considering that multiple shows are produced each semester, this data can quickly become overwhelming to keep track of if not organized in a concise and cohesive format. This database will also help track information by managing searches for data automatically, instead of manually, and easily present many years of data quickly. Our goal is to create a resource for club members to have a comprehensive history of student theater productions on campus at their fingertips to provide clubs an easy way to see and analyze their show/member history.

**Data Description:** In this database, we will be tracking the following main entities: members, clubs, and shows. The members can be associated with any show in which they participate, with a specified acting role/pit position/staff positions in each show. Members with an acting role are designated as part of the Cast for the show where a cast member is comprised of a member, show, role, and a voice part for the role. Members with a band position are designated as part of the Pit for the show where a pit member is comprised of a member, Instrument the member plays, their seat number for their instrument, and the show in which they played. Members with a staff position are designated as part of the Staff for the show where a staff member is comprised of a member, show, and their position on the show. Shows are comprised of their name, the author of the show, the budget for the show, a boolean value of whether or not the show is a musical, the club that put on the show, and the date on which the show was performed. Dates are simply an academic semester defined by a year and a semester (Fall or Spring). Members with an exec position are designated as part of Exec where an exec member is comprised of a particular role, member, and date being the semester for which they hold that position, though an exec member is not linked to a particular club. These positions can be President, Vice President, Treasurer, Secretary, or Technical Director.

## ER Diagram:



## Functional Dependencies:

members: (intMemberID, strName)

$\text{intMemberID} \rightarrow \text{strName}$

Each intMemberID is correlated to exactly one name. BCNF form.

exec: (intExecID, strPosition, intMemberID, intDateID, intClubID)

$\text{intExecID} \rightarrow \text{strPosition}, \text{intMemberID}, \text{intDateID}, \text{intClubID}$

Each intExecID is associated with exactly one position, member, and date. A member may have multiple exec positions in different clubs and therefore have multiple intExecIDs because of the unique dates. BCNF form.

casts: (intCastID, strVoicePart, strRole, intMemberID, intShowID)

intCastID → strVoicePart, strRole, intMemberID, intShowID

Each intCastID is associated with exactly one voice part, role, member, and show. A member may be cast in different shows and therefore have multiple intCastIDs because of the unique roles and shows. BCNF form.

staff: (intStaffID, strPosition, intMemberID, intShowID)

intStaffID → strPosition, intMemberID, intShowID

Each intStaffID is associated with exactly one position, member, and show. A member may be a part of the production staff in different shows and therefore have multiple intStaffIDs because of the unique shows. BCNF form.

pits: (intPitID, intInstrument, intSeat, intMemberID, intShowID)

intPitID → intInstrument, intSeat, intMemberID, intShowID

Each intPitID is associated with exactly one instrument, seat, member, and show. A member may be a part of different shows' pits and therefore have multiple intPitIDs because of the unique shows. BCNF form.

dates: (intDateID, strSemester, intYear)

intDateID → strSemester, intYear

Each intDateID is correlated to exactly one semester and one year. BCNF form.

clubs: (intClubID, strClubName, strAdvisor, strUmbrellaOrg)

intClubID → strClubName, strAdvisor, strUmbrellaOrg

Each intClubID is correlated to exactly one name, advisor, and umbrella organization. BCNF form.

shows: (intShowID, strName, strAuthor, intBudget, ysnIsMusical)

intShowID → strName, strAuthor, intBudget, ysnIsMusical

Each intShowID is correlated to exactly one name, author, budget, and boolean indicating if it is a musical or not. An author may be listed for different shows and budgets for the shows may be the same. BCNF form.

pit\_in\_show: (intPitID, intShowID)

intPitID → intShowID

Each pit ID is only associated with one show. BCNF form.

staff\_in\_show: (intStaffID, intShowID)

intStaffID → intShowID

Each staff ID is only associated with one show. BCNF form.

cast\_in\_show: (intCastID, intShowID)

intCastID → intShowID

Each cast ID is only associated with one show. BCNF form.

club\_member: (intClubID, intMemberID)

There are no functional dependencies for this because clubs can have multiple members and

members can be in multiple clubs.

perform: (intClubID, intShowID)

There are no functional dependencies for this because clubs perform multiple shows and different clubs can perform the same show.

on: (intShowID, intDateID)

$\text{intShowID} \rightarrow \text{intDateID}$

Each show ID is only associated with one date. BCNF form.

leader\_for: (intExecID, intDateID)

$\text{intExecID} \rightarrow \text{intDateID}$

Each exec ID is only associated with one date. BCNF form.

leader\_of: (intExecID, intClubID)

$\text{intExecID} \rightarrow \text{intClubID}$

Each exec ID is only associated with one club. BCNF form.

## Schemas:

Members: (intMemberID, strName)

intMemberID: id for the member, primary key, not null

strName: name of member, not null

Exec: (intExecID, strPosition, intMemberID, intDateID, intClubID)

intExecID: id for the exec member, primary key, not null

strPosition: position held, not null

intMemberID: id of member, foreign key, references Members set

intDateID: id of date that exec was in office, foreign key, references Dates set

intClubID: id of club that exec was presiding over, foreign key, references Clubs set

Casts: (intCastID, strVoicePart, strRole, intMemberID, intShowID)

intCastID: id for the cast member, primary key, not null

strVoicePart: voice part for singing role (can be null if only speaking part and not singing part)

strRole: role held in show, not null

intMemberID: id of member, foreign key, references Members set

intShowID: id of show that cast member was part of, foreign key, references Shows set

Staff: (intStaffID, strPosition, intMemberID, intShowID)

intStaffID: id for the staff member, primary key, not null

strPosition: position held by member, not null

intMemberID: id of member, foreign key, references Members set

intShowID: id of show that staff member was part of, foreign key, references Shows set

Pits: (intPitID, intInstrument, intSeat, intMemberID, intShowID)

intPitID: id for the pit member, primary key, not null  
intInstrument: instrument pit member plays, not null  
intSeat: seat of pit member, not null, default 1  
intMemberID: id of member, foreign key, references Members set  
intShowID: id of show that pit member was part of, foreign key, references Shows set

Dates: (intDateID, strSemester, intYear)  
intDateID: id for the date, primary key, not null  
strSemester: semester of the date, not null  
strYear: year of the date, not null

Clubs: (intClubID, strClubName, strAdvisor, strUmbrellaOrg)  
intClubID: id for the club, primary key, not null  
strClubName: name of the club, not null  
strAdvisor: advisor for the club, not null  
strUmbrellaOrg: umbrella organization that club is under, not null

Shows: (intShowID, strName, strAuthor, intBudget, ysnIsMusical)  
intShowID: id for the show, primary key, not null  
strName: name of the show, not null  
strAuthor: author of the show, not null  
intBudget: budget for the show's production, not null  
ysnIsMusical: indicates if the show is a musical (true) or not (false), not null

pit\_in\_show: (intPitID, intShowID)  
intPitID: id for the pit member, primary key, not null  
intShowID: id for the show, primary key, not null

staff\_in\_show: (intStaffID, intShowID)  
intStaffID: id for the staff member, primary key, not null  
intShowID: id for the show, primary key, not null

cast\_in\_show: (intCastID, intShowID)  
intCastID: id for the cast member, primary key, not null  
intShowID: id for the show, primary key, not null

club\_member: (intClubID, intMemberID)  
intClubID: id for the club, primary key, not null

perform: (intClubID, intShowID)  
intClubID: id for the club, primary key, not null  
intShowID: id for the show, primary key, not null

on: (intShowID, intDateID)  
intShowID: id for the show, primary key, not null  
intDateID: id for the date, primary key, not null

leader\_for: (intExecID, intDateID)

intExecID: id for the exec member, primary key, not null  
intDateID: id for the date, primary key, not null

leader\_of: (intExecID, intClubID)

intExecID: id for the exec member, primary key, not null  
intClubID: id for the club, primary key, not null

### **Queries Supported:**

Add a new member to the members table:

```
INSERT INTO members ("strName") VALUES ('name')
```

Add a new club to the clubs table:

```
INSERT INTO clubs ("strClubName", "strAdvisor", "strUmbrellaOrg") VALUES ('club_name',  
'advisor_name', 'umbrellaOrg_name')
```

Add a new date to the dates table:

```
INSERT INTO dates ("strSemester", "intYear") VALUES ('semester', 'year')
```

Add a new show to the shows table:

```
INSERT INTO shows ("strName", "strAuthor", "intBudget", "ysnIsMusical", "intClubID",  
"intDateID") VALUES ('name', 'author', budget, isMusical, clubID, dateID)
```

Add a new staff member to the staff table:

```
INSERT INTO staff ("strPosition", "intMemberID", "intShowID") VALUES ('position',  
memberID, showID)
```

Add a new pit member to the pits table:

```
INSERT INTO pit ("strInstrument", "intSeatID", "intMemberID", "intShowID") VALUES  
(‘instrument’, seatID, memberID, showID)
```

Add a new exec member to the exec table:

```
INSERT INTO exec ("strPosition", "intMemberID", "intDateID", "intClubID") VALUES  
(‘position’, memberID, dateID, clubID)
```

Add a new cast member to the casts table:

```
INSERT INTO casts ("strVoicePart", "strRole", "intMemberID", "intShowID") VALUES  
(‘voice_part’, ‘role_name’, memberID, showID)
```

Change the name for an existing member:

```
UPDATE members SET "strName" = ‘new_name’ WHERE "strName" = ‘original_name’
```

Change the name of an existing club:

```
UPDATE clubs SET "strClubName" = ‘new_club_name’ WHERE "strClubName" =  
‘original_club_name’
```

Change the budget of an existing show:

```
UPDATE shows SET "intBudget" = new_budget WHERE "intBudget" = original_budget
```

Change the name of any member that has been a staff member and currently has the name “original\_name”:

```
UPDATE members SET “strName” = ‘new_name’ WHERE (SELECT * FROM members m, staff s WHERE m.intMemberID=s.intMemberID AND m.strName = ‘original_name’)
```

Change the name of any member that has been a pit member and has currently has the name “original\_name”:

```
UPDATE members SET “strName” = ‘new_name’ WHERE (SELECT * FROM members m, pit p WHERE m.intMemberID=p.intMemberID AND m.strName = ‘original_name’)
```

Change the name of any member that has been an exec member and has currently has the name “original\_name”:

```
UPDATE members SET “strName” = ‘new_name’ WHERE (SELECT * FROM members m, exec e WHERE m.intMemberID=e.intMemberID AND m.strName = ‘original_name’)
```

Change the name of any member that has been a cast member and has currently has the name “original\_name”:

```
UPDATE members SET “strName” = ‘new_name’ WHERE (SELECT * FROM members m, casts c WHERE m.intMemberID=c.intMemberID AND m.strName = ‘original_name’)
```

Delete any members that have the name “name”:

```
DELETE FROM members m WHERE m.strName = ‘name’
```

Delete any clubs that have the name “club\_name”:



```
DELETE FROM clubs c WHERE c.strClubName = 'club_name'
```

Delete any shows with the name “show\_name” that have a staff and delete staff members from these shows:

```
DELETE FROM staff st, shows sh WHERE st.intShowID = sh.intShowID AND  
sh.strShowName = 'show_name'
```

Select all clubs that have done at least one show as well as the average budget of their shows:

```
"SELECT cl."strClubName", AVG(s."intBudget") as intAvgBudget FROM clubs cl, shows s  
WHERE s."intClubID" = cl."intClubID" GROUP BY cl."strClubName"
```

Show the number of shows that have been put on that are musicals:

```
SELECT COUNT(s.*) FROM shows s WHERE s."ysnIsMusical" = TRUE
```

Show the number of shows that have been put on that are not musicals:

```
SELECT COUNT(s.*) FROM shows s WHERE s."ysnIsMusical" = FALSE
```

Show each position that has been held by a staff member:

```
SELECT DISTINCT "strPosition" FROM staff
```

Show the name of any member that has held the staff position “strPositionName” at least twice and number of times they have held it:

```
SELECT m."strName", count(m.*) FROM members m, staff s WHERE s."strPosition" =
```

```
'strPositionName' AND s."intMemberID" = m."intMemberID" GROUP BY m."strName"  
HAVING count(m.*) >= 2
```

Show each voice part that a cast member has held:

```
SELECT DISTINCT "strVoicePart" FROM casts
```

Show the name of each cast member that has performed with the voice part “strVoicePart” and how many times they have done so:

```
SELECT m."strName", COUNT(m."strName") FROM members m, casts c WHERE  
m."intMemberID" = c."intMemberID" AND c."strVoicePart" = 'strVoicePart' GROUP BY  
m."strName"
```

Show only once each show and instrument in that show such that the instrument contains the string “strSearch”:

```
SELECT DISTINCT s."intShowID", p."strInstrument" FROM pits p, shows s WHERE  
p."intShowID" = s."intShowID" and p."strInstrument" LIKE '%' + strSearch + '%'
```

Show all members that have been part of a cast of a musical with a voice part of null:

```
SELECT m."intMemberID" FROM members m WHERE EXISTS (SELECT * FROM casts c,  
shows s WHERE c."intMemberID" = m."intMemberID" AND c."intShowID" = s."intShowID"  
AND s."ysnIsMusical" = TRUE AND c."strVoicePart" IS NULL)
```

Show each member that has been in a cast but was not in a cast for the show with show ID = intShowID:

```
SELECT DISTINCT m."intMemberID" FROM members m, casts c WHERE m."intMemberID"
```

= c."intMemberID" EXCEPT (SELECT m2."intMemberID" FROM members m2, shows s, casts c2 WHERE m2."intMemberID" = c2."intMemberID" AND c2."intShowID" = s."intShowID" AND s."intShowID" = intShowID)

Select all cast members with the cast ID "intCastID":

```
SELECT * FROM casts WHERE "intCastID" = 'intCastID'
```

Select all data from cast table:

```
SELECT * FROM casts
```

Select all clubs with the club ID "intClubID":

```
SELECT * FROM clubs WHERE "intClubID" = 'intClubID'
```

Select the club ID's, club names, advisors, and umbrella organizations from each club:

```
SELECT "intClubID", "strClubName", "strAdvisor", "strUmbrellaOrg" FROM clubs
```

Select all staff members with the staff ID "intStaffID":

```
SELECT * FROM staff WHERE "intStaffID" = 'intStaffID'
```

Select all data from staff table:

```
SELECT * FROM staff
```

Select all shows with show ID “intShowID”:

```
SELECT * FROM shows WHERE "intShowID" = 'intShowID'
```

Select all data from shows table:

```
SELECT * FROM shows
```

Select all pit members with pit ID “intPitID”:

```
SELECT * FROM pits WHERE "intPitID" = 'intPitID'
```

Select all data from pit table:

```
SELECT * FROM pits
```

Select all members with member ID “intMemberID”:

```
SELECT * FROM members WHERE "intMemberID" = 'intMemberID'
```

Select all data from members table:

```
SELECT * FROM members
```

Select all exec members with ecec ID “intExecID”:

```
SELECT * FROM exec WHERE "intExecID" = 'intExecID'
```

Select all data from exec table:

```
SELECT * FROM exec
```

Select all dates with date ID "intDateID":

```
SELECT * FROM dates WHERE "intDateID" = 'intDateID'
```

Select the date ID, semester, and year from each date:

```
SELECT "intDateID", "strSemester", "intYear" FROM dates
```

### **Implementation:**

Instead of listing all the creates and inserts here, please see the folder "~/Content/sql/" in the zip file for the .sql files containing the scripts run to create the database and insert initial data. They should be run in the order according to the number at the beginning of the file (01, 02, etc).

### **Roles and Contributions:**

Giuliana Conte: Project Lead, wrote all javascript, html, css, & view code and some controller, DAL, & model code. completed the analytics portion of the site independently.

Ryan Arvizu: Creating E-R diagram, populating all tables with data in SQL, helped with Models and Views in code, Functional Dependencies, Schemas

Erik Espinosa:

Nicholas Charles: Creating Data Access Layer for each table, English for queries, some query writing, functional testing

### **Technology Stack Used:**

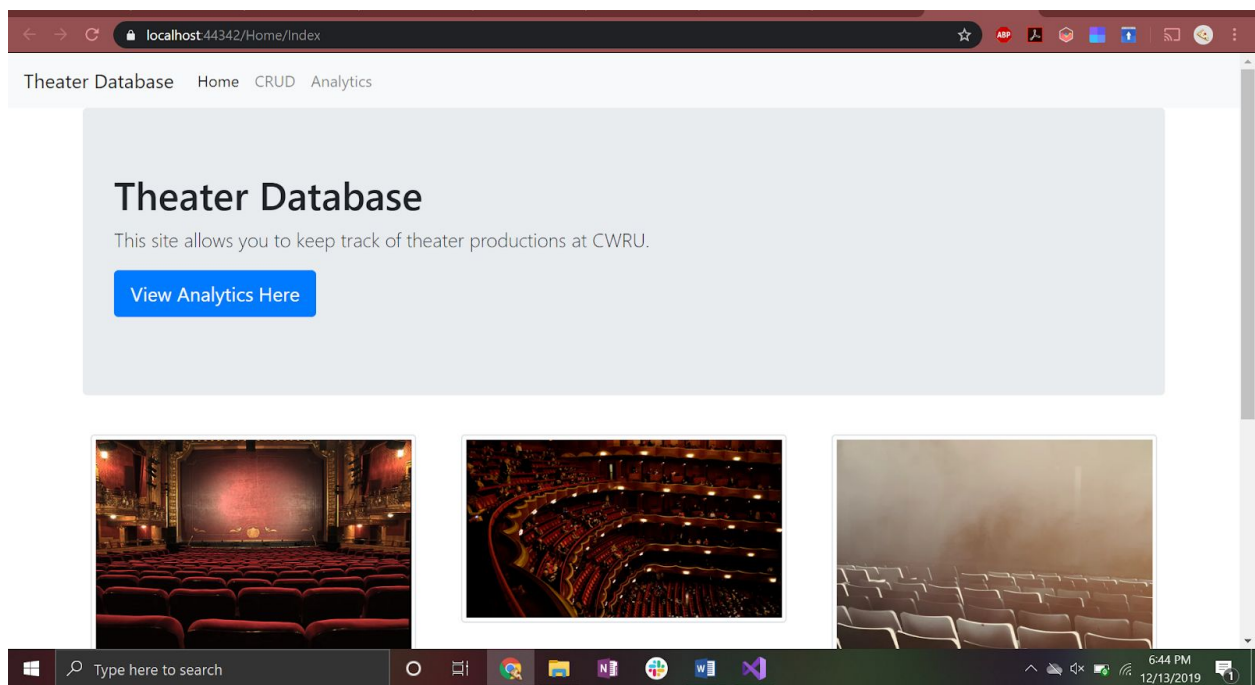
We implemented this solution using ASP.NET MVC, which uses C# for the backend code and HTML/CSS/javascript for the front end. For the database we used PostgreSQL. We used GitHub for version control and Visual Studio 2019 as our development environment.

### **What you have learned from this project:**

We have learned many valuable lessons about implementing a CRUD app so that users can insert, delete, and update their own data from the database. We also learned about the MVC design pattern, which is very popular in modern web development. Some of us used C# for the first time in this project, which was a good learning experience.

### **Appendix -- screenshots:**

home page:



CRUD shows (reading only):

localhost:44342/Home/Crud

Theater Database Home CRUD Analytics







## CRUD Operations

Create, Read, Update, and Delete table info here

Shows Members Exec Casts Staff Pits

### Shows

Show / Hide Columns Search...

| Show ID | Name           | Author           | Budget | Is Musical                          | Club         | Date        |  |
|---------|----------------|------------------|--------|-------------------------------------|--------------|-------------|--|
| 1       | Into the Woods | Stephen Sondheim | 10520  | <input checked="" type="checkbox"/> | Footlighters | FALL 2017   | <br> |
| 2       | Be More Chill  | Joe Tracz        | 11150  | <input checked="" type="checkbox"/> | Footlighters | SPRING 2018 | <br> |
| 3       | If/Then        | Brian Yorkey     | 19500  | <input checked="" type="checkbox"/> | Footlighters | FALL 2018   | <br> |

Type here to search

6:45 PM 12/13/2019

CRUD shows (when editing budget for one item):

localhost:44342/Home/Crud

Theater Database Home CRUD Analytics








## CRUD Operations

Create, Read, Update, and Delete table info here

Shows Members Exec Casts Staff Pits

### Shows

Show / Hide Columns Search...







| Show ID | Name           | Author           | Budget                          | Is Musical                          | Club         | Date        |   |
|---------|----------------|------------------|---------------------------------|-------------------------------------|--------------|-------------|---|
| 1       | Into the Woods | Stephen Sondheim | 10520                           | <input checked="" type="checkbox"/> | Footlighters | FALL 2017   | <br>  |
| 2       | Be More Chill  | Joe Tracz        | <input type="text" value="10"/> | <input checked="" type="checkbox"/> | Footlighters | SPRING 2018 | <br><br> |
| 3       | If/Then        | Brian Yorkey     | 19500                           | <input checked="" type="checkbox"/> | Footlighters | FALL 2018   | <br>  |

Type here to search

6:46 PM 12/13/2019

CRUD shows (inserting):

localhost:44342/Home/Crud

|    |                              |             |     |                          |                       |             |  |
|----|------------------------------|-------------|-----|--------------------------|-----------------------|-------------|--|
| 8  | Revolt She Said Revolt Again | Alice Birch | 250 | <input type="checkbox"/> | Players Theatre Group | FALL 2019   | <br> |
| 9  | Beware of Falling Deer       | Adam Steel  | 500 | <input type="checkbox"/> | Players Theatre Group | FALL 2019   | <br> |
| 14 | Circle Mirror Transformation | Annie Baker | 150 | <input type="checkbox"/> | Players Theatre Group | SPRING 2019 | <br> |

Previous 1 2 Next

---

☐ Musical?

FALL 2017  
 FALL 2017  
 SPRING 2018  
 FALL 2018  
 SPRING 2019  
 FALL 2019  
 SPRING 2020

EECS 341 Final Project      Giuliana Conte, Ryan Arvizu, Nick Charles, Erik Espinosa      © 2019

See each tab on the live version of the GUI for the other tabs, which are more or less identical to this one for shows, but support insert/update/delete queries for the respective tables.

Analytics (search instruments):

localhost:44342/Home/Analytics

Theater Database   Home   CRUD   Analytics

## Analytics

---

### Shows by Instrument

Search an instrument and see what shows we have done with that instrument!

Search Term: horn

| Name                       | Instrument        |
|----------------------------|-------------------|
| Into the Woods             | Horn              |
| The Mystery of Edwin Drood | French Horn       |
| The Mystery of Edwin Drood | Oboe/English Horn |

---

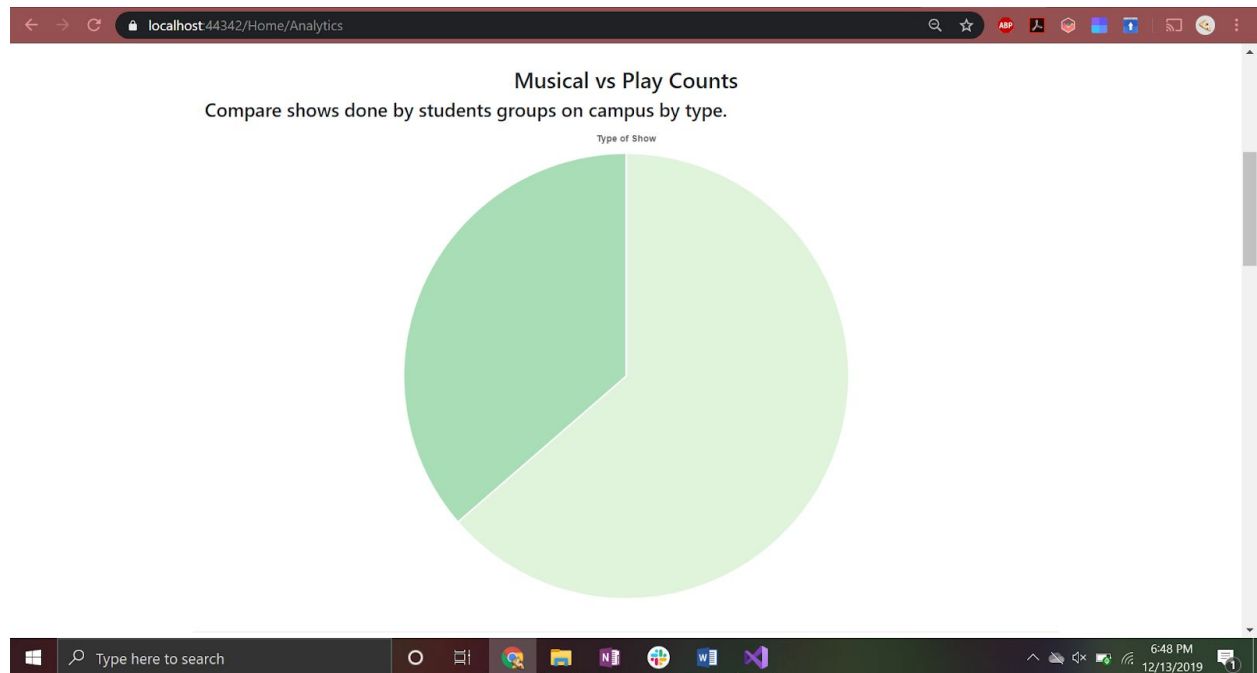
### Musical vs Play Counts

Compare shows done by students groups on campus by type.

Type of Show



Analytics (pie graph counts):



Analytics (interactive drop down of member by voice part):

localhost:44342/Home/Analytics

### Members By Voice Part

View list of members & how many times they have played a role with a given voice part.

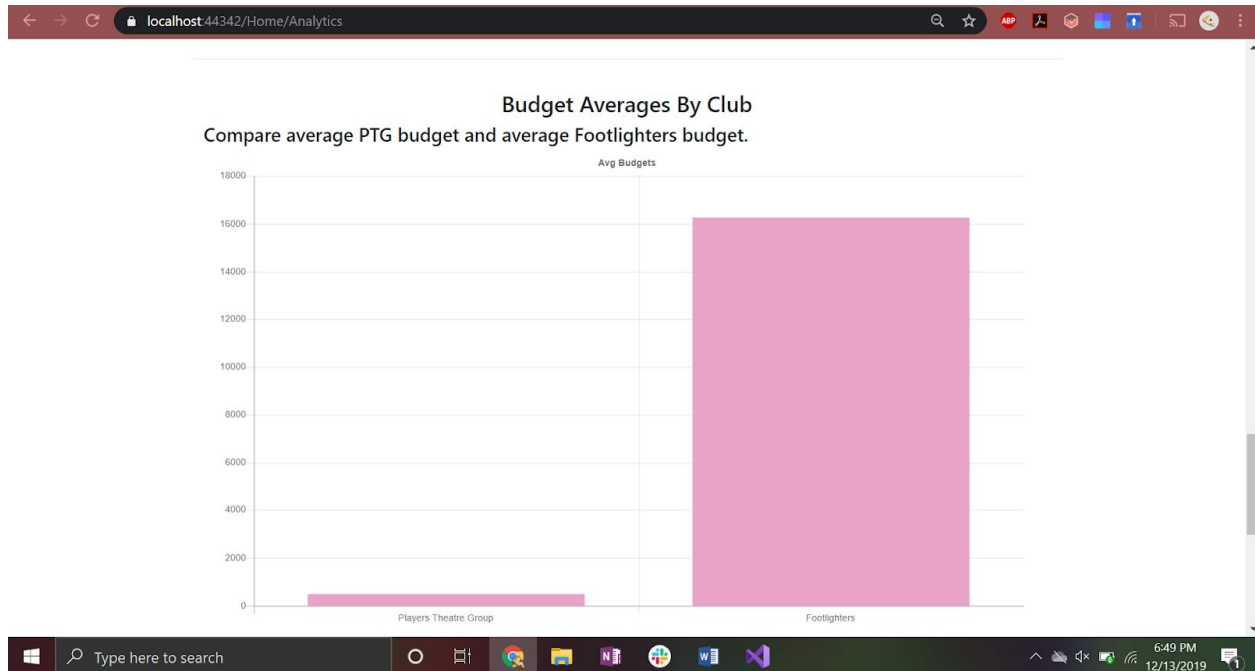
Tenor

Voice Part: Tenor

| Name               | Count |
|--------------------|-------|
| Dylan Kulbacki     | 1     |
| Jack Petito        | 1     |
| Abigail McAllister | 1     |
| Luke Dotson        | 2     |
| Kyle Rickert       | 2     |
| Jacob Erney        | 1     |
| Johnathan Sequeira | 1     |
| Shaun Furter       | 1     |
| Kegan Sulamoyo     | 1     |
| Ben Baierl         | 1     |

Windows taskbar: 6:48 PM 12/13/2019

Analytics (budget average for shows by club):



Analytics (count/having query and except query):

localhost:44342/Home/Analytics

### Repeat Staffs

View list of members who have served on a production staff in a given role more than once.

Director

Position: Director

| Name           | Count |
|----------------|-------|
| Giuliana Conte | 4     |
| Paulina Martz  | 3     |

### All Actors Not In ...

View list of actors who have acted but NOT in a given show.

Into the Woods

Not In: Into the Woods

| Name            |
|-----------------|
| Marissa Lahr    |
| Jennifer Newton |
| Sam Aram        |

Windows taskbar: 6:50 PM, 12/13/2019