Team: Savage and Average
Team Members: Mallory Huston and Sory Diagouraga
Date: Monday, July 18th, 2022

Project Step 3 Draft: Design HTML Interface + DML SQL

URL to an index.html page:

https://web.engr.oregonstate.edu/~diagours/cs340/index.html

Project Outline

Clearwater Arena is a stadium near multiple small towns in the state of Wisconsin that provide world-class entertainment through their venues. They provide 30 concerts per month from different artists around the country. It requires a database to store their ticket sales, artists, fan information, and price per ticket. The venue has 5,000 seats and the Employees have to make sure that it never goes over maximum capacity. There are currently 25 employees at Clearwater Arena who provide different services to fans such as bartenders, customer service reps, cashiers, and security. A database driven website will record all of the sales of Tickets for Fans, Artists, and Employees.

Database Outline

Object Entities:

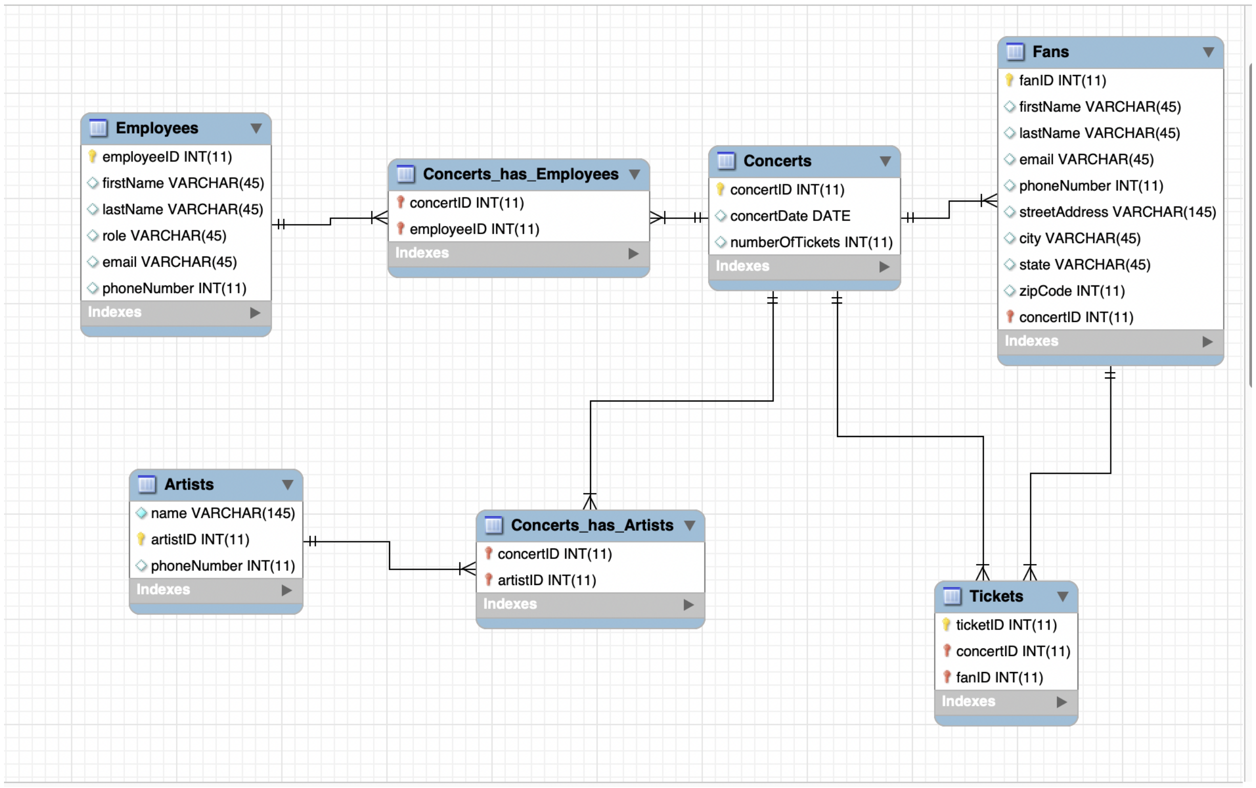
- Fans
    - fanID: int, auto_increment, unique, not NULL, PK
    - email: varchar
    - firstName: varchar
    - lastName: varchar
    - phone: varchar
    - address: varchar
    - city: varchar
    - state: varchar
    - zipCode: varchar
    - Relationships:
        - A 1:M relationship between Fans and Tickets is implemented with fan_ID as a foreign key inside of Tickets. One fan may buy a ticket and a ticket can be purchased by multiple fans.

- Employees
    - employeeID: int, auto_increment, unique, not NULL, PK
    - role: varchar
    - firstName: varchar
    - lastName: varchar
    - email: varchar
    - phoneNumber: int
    - Relationships:
        - A M:M relationship between Employees and Concerts is implemented with an intersection table, since one Employee may be responsible for multiple Concerts, and one Concert may be served by multiple Employees.

- Concerts
    - concertID: int, auto_increment, unique, not NULL, PK
    - concertDate: date
    - numberOfTickets: int
    - Relationships:
        - A 1:M relationship exists between Concerts and Fans. It's implemented with concertID as a foreign key inside of Fans. One Fan may only attend one Concert and a Concert may have many Fans.
        - A M:M relationship exists between Concerts and Artists. A concert can have many artists and an artist can have multiple concerts.
- Artists
    - artistID: int, auto_increment, unique, not NULL, PK
    - name: varchar, not NULL
    - phoneNumber: int
    - Relationship:
        - A M:M relationship between Artists and Fans is implemented with an intersection table, since one Artist may perform at multiple concerts and a Concert may have multiple Artists performing.

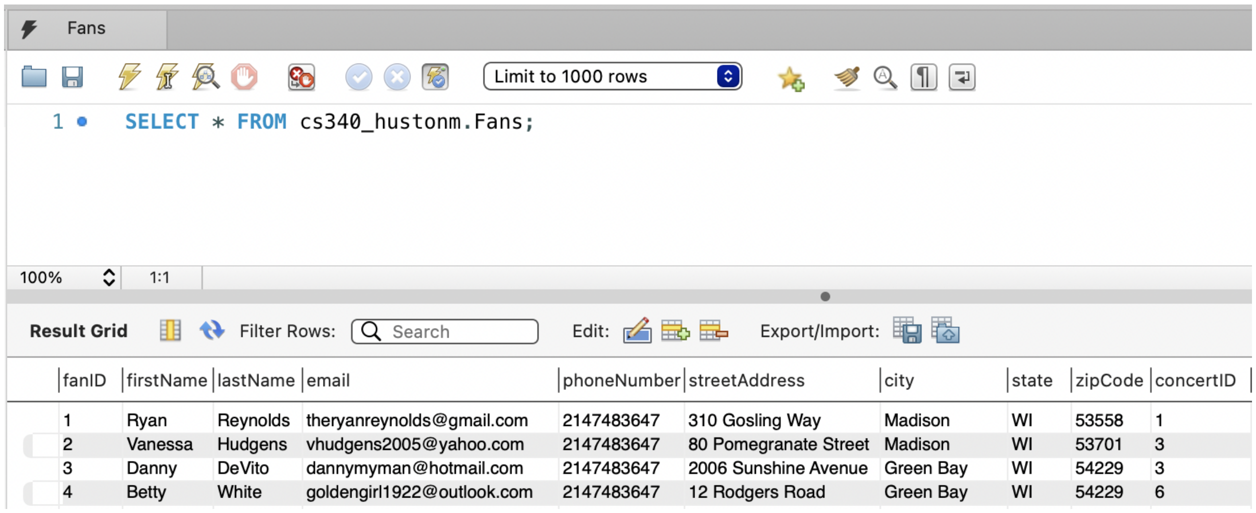
- Tickets
    - ticketID: int, auto_increment, unique, not NULL, PK
    - Relationships:
        - A 1:M relationship exists between Tickets and Fans. A ticket may only be bought by one fan, but a fan can buy multiple tickets.
        - A 1:M relationship exists between Tickets and Concerts. A ticket may only be for one concert, while a concert may have many tickets.

Entity Relationship Diagram / Schema



Sample Data

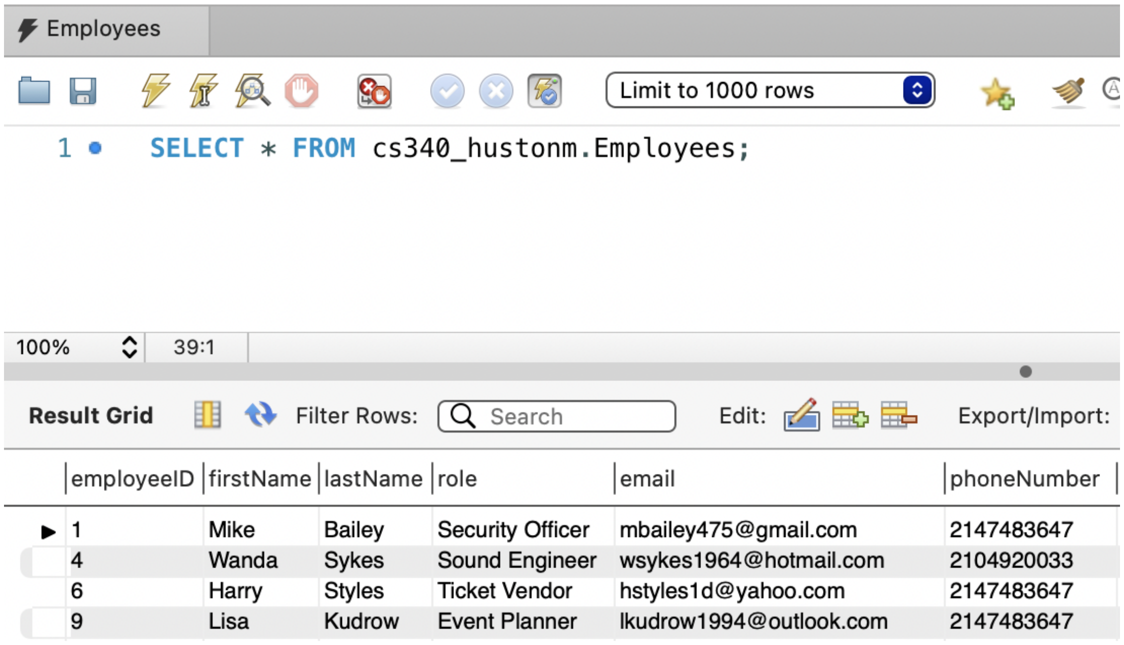
Fans:



```
SELECT * FROM cs340_hustonm.Fans;
```

fanID	firstName	lastName	email	phoneNumber	streetAddress	city	state	zipCode	concertID
1	Ryan	Reynolds	theryanreynolds@gmail.com	2147483647	310 Gosling Way	Madison	WI	53558	1
2	Vanessa	Hudgens	vhudgens2005@yahoo.com	2147483647	80 Pomegranate Street	Madison	WI	53701	3
3	Danny	DeVito	dannymyman@hotmail.com	2147483647	2006 Sunshine Avenue	Green Bay	WI	54229	3
4	Betty	White	goldengirl1922@outlook.com	2147483647	12 Rodgers Road	Green Bay	WI	54229	6

Employees:



Employees

Limit to 1000 rows

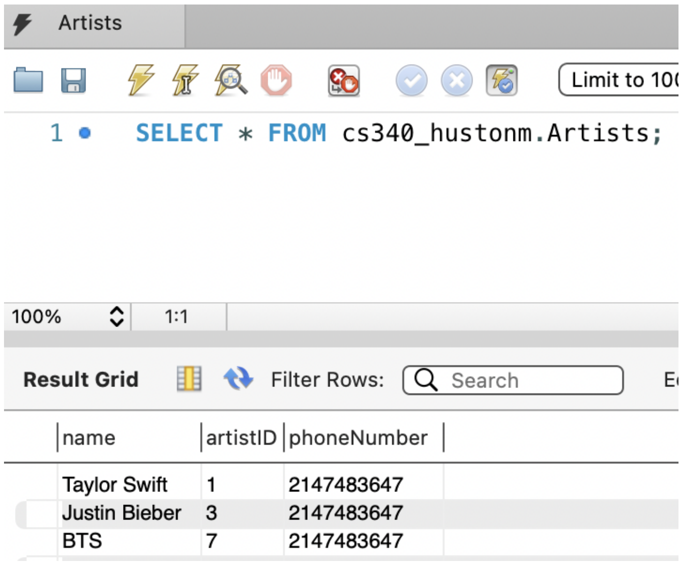
```
SELECT * FROM cs340_hustonm.Employees;
```

100% 39:1

Result Grid Filter Rows: Search Edit: Export/Import:

employeeID	firstName	lastName	role	email	phoneNumber
1	Mike	Bailey	Security Officer	mbailey475@gmail.com	2147483647
4	Wanda	Sykes	Sound Engineer	wsykes1964@hotmail.com	2104920033
6	Harry	Styles	Ticket Vendor	hstyles1d@yahoo.com	2147483647
9	Lisa	Kudrow	Event Planner	lkudrow1994@outlook.com	2147483647

Artists:



Artists

Limit to 10(

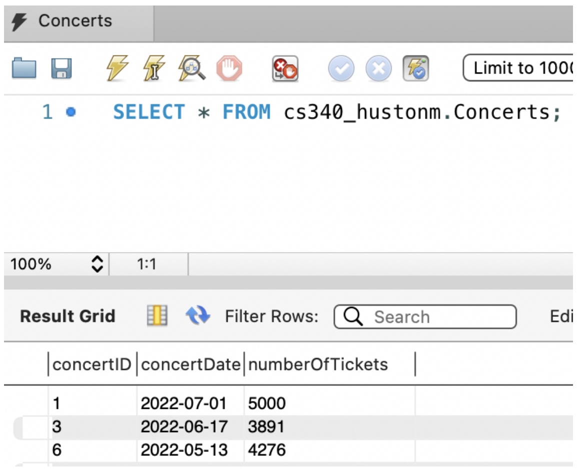
```
SELECT * FROM cs340_hustonm.Artists;
```

100% 1:1

Result Grid Filter Rows: Search E

name	artistID	phoneNumber
Taylor Swift	1	2147483647
Justin Bieber	3	2147483647
BTS	7	2147483647

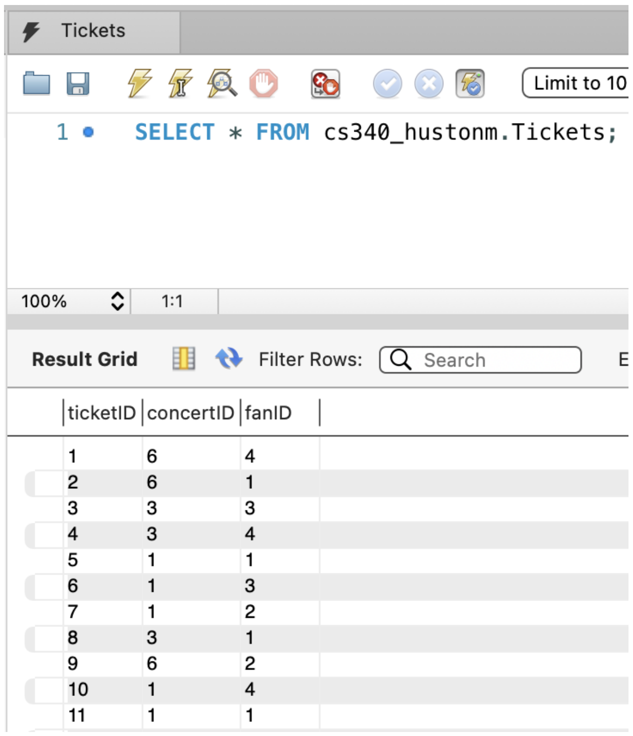
Concerts:



```
SELECT * FROM cs340_hustonm.Concerts;
```

concertID	concertDate	numberOfTickets
1	2022-07-01	5000
3	2022-06-17	3891
6	2022-05-13	4276

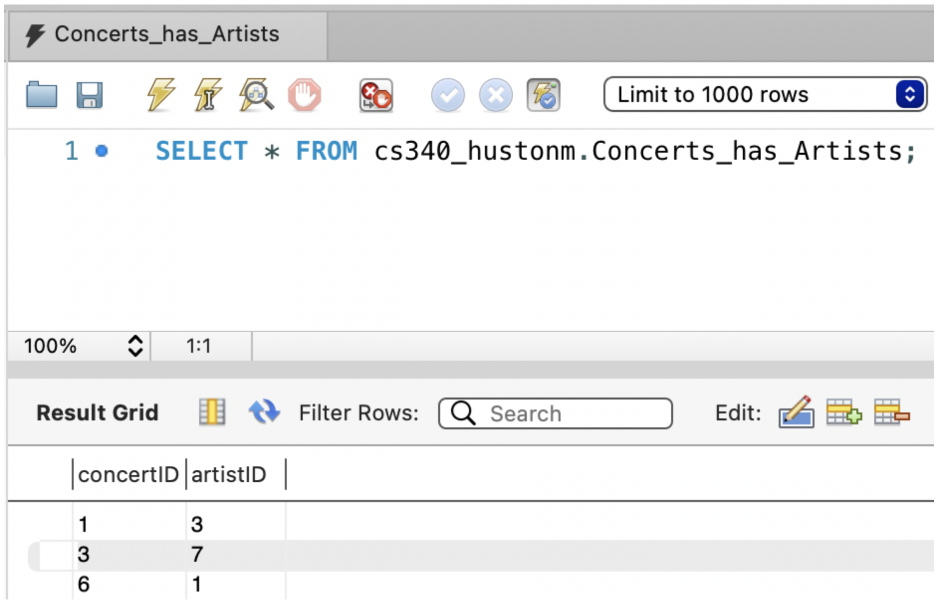
Tickets:



```
SELECT * FROM cs340_hustonm.Tickets;
```

ticketID	concertID	fanID
1	6	4
2	6	1
3	3	3
4	3	4
5	1	1
6	1	3
7	1	2
8	3	1
9	6	2
10	1	4
11	1	1

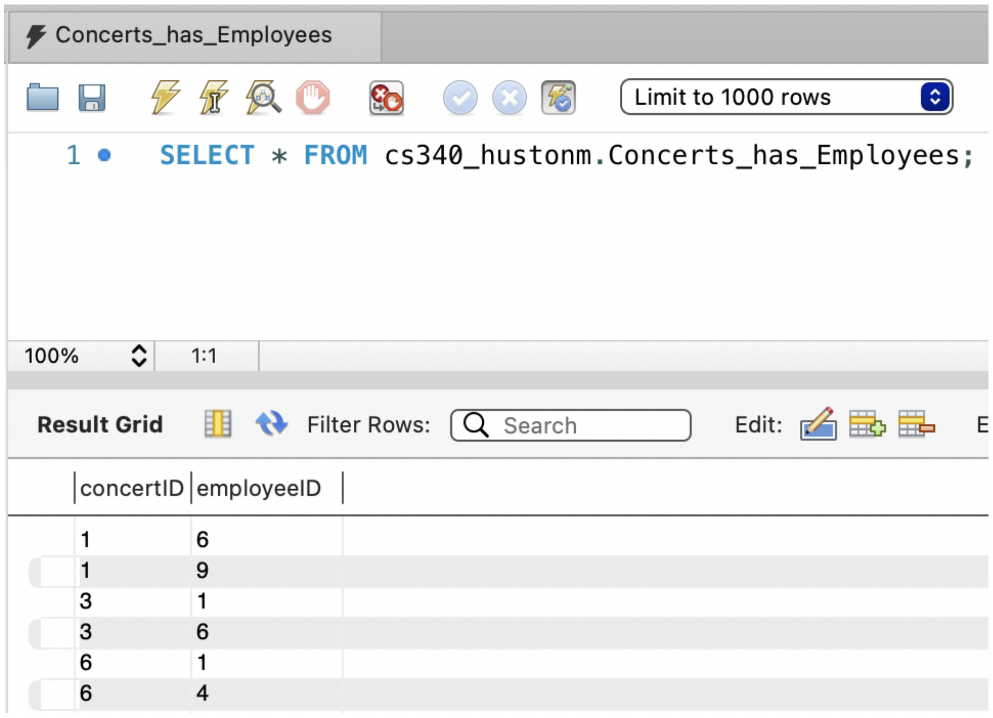
Concerts_has_Artists (Intersection Table):



```
SELECT * FROM cs340_hustonm.Concerts_has_Artists;
```

concertID	artistID
1	3
3	7
6	1

Concerts_has_Employees (Intersection Table):



```
SELECT * FROM cs340_hustonm.Concerts_has_Employees;
```

concertID	employeeID
1	6
1	9
3	1
3	6
6	1
6	4

<u>Fixes based on Feedback from Previous Steps</u>

Actions Taken:

<u>Overview:</u> We immediately changed the typo of “Clearwater Arena” to “Clearwater Casino” in the overview to avoid any further confusion. We then took out “promotions” and “genres” in the overview since we did not cover that whatsoever since it would make this project way too complicated.
<u>Employees and Artists:</u> We also took out any relationship between the Employees and Artists because it was not necessary and would be far too complicated. Plus, it is not how real life works and whichever employees would be helping out the artists are the ones who are on staff that night. There cannot be more than one show per night either.
<u>numberOfTickets:</u> We also put in the entity “numberOfTickets” under Concerts so Employees can keep track of how many people show up and that the venue never goes over max capacity.
<u>artistID:</u> We then made the artistID entity the same as the other entities that are unique, not NULL, auto_increment, etc.
<u>Fans:</u> We decided to rename the entity “Customers” to “Fans” because that type of demographic makes more sense for a concert arena anyway and it’s less confusing.
<u>MySQL Cascade:</u> We had to input CASCADE operations between two tables that were dependent on each other. We put ON DELETE CASCADE for the constraint \`fk_Tickets_Concerts1\`, ON UPDATE CASCADE for \`fk_Tickets_Fans1\`, and ON DELETE CASCADE for \`fk_Fans_Concerts1\`. We put in ON DELETE CASCADE for all the foreign keys in the intersection tables as well.
<u>Concerts_concertID:</u> We decided to remove “Concerts_concertID” and all other similar names from the intersection tables and other entities to avoid any further confusion.

Actions Not Taken:

<u>Genres and Promotions:</u> We did not follow the other suggestions of putting in genres and promotions since it would over complicate our database.
<u>Employee role:</u> We also want to make it clear that all Employees can only have one role each, not multiple.
<u>Artistscol and Artistscol1:</u> We did delete “Artistscol” and “Artistscol 1” since they did not make much sense anyway.
<u>concertID:</u> Additionally, we want to make it clear that the relationship between Concerts and Fans is implemented with “concertID” while Artists and Concerts has an implementation table.