ECHO Database Final Project

Database Management Systems – I
Final Project Report

CS4380/7380 - Spring 2020

Instructor: Dr. Chi Ren Shyu

Team Members
Brett Eckert
Sweta Praharaj
Rebecca Shyu
Kaitlyn Zahn

Contents

Introduction:

The client, Missouri Telehealth Network at the University of Missouri, runs the Show-Me ECHO (Extension for Community Healthcare Outcomes) program. This program is a telemedicine-based program that uses video-conferencing technologies to connect experts with primary care providers around the world. It allows providers to bring cases up in sessions for expert opinion and discussion about new developments in medicine in various areas. Show-Me ECHO began in 2015 and the concept was ”developed by the University of New Mexico to educate and train participating providers in specific disease states or conditions.” This program has been designated a “super-hub” which is a training organization (1 of 7 total). There are 49 total ECHO categories, such as Asthma, Autism, Child Psychology, COVID-19, Dermatology, and more. A variety of attendees register and participate in sessions for each ECHO category. At each of these sessions, attendees/facilitators were able to bring specific cases within their care up for discussion and recommendations.

The data we received was in the form of CSV files and written notes from different sources. We had CSVs for tables such as Organizations, Sessions, ECHOs, and Attendees. To create and successfully load data into our tables, we had to perform data carpentry on the data sets to link together. There was a lot of human error between Organizations and Attendees/Contacts when registering that needed to be fixed. We also used public-domain data (County Health Rankings and Census) for background information for Missouri Counties.

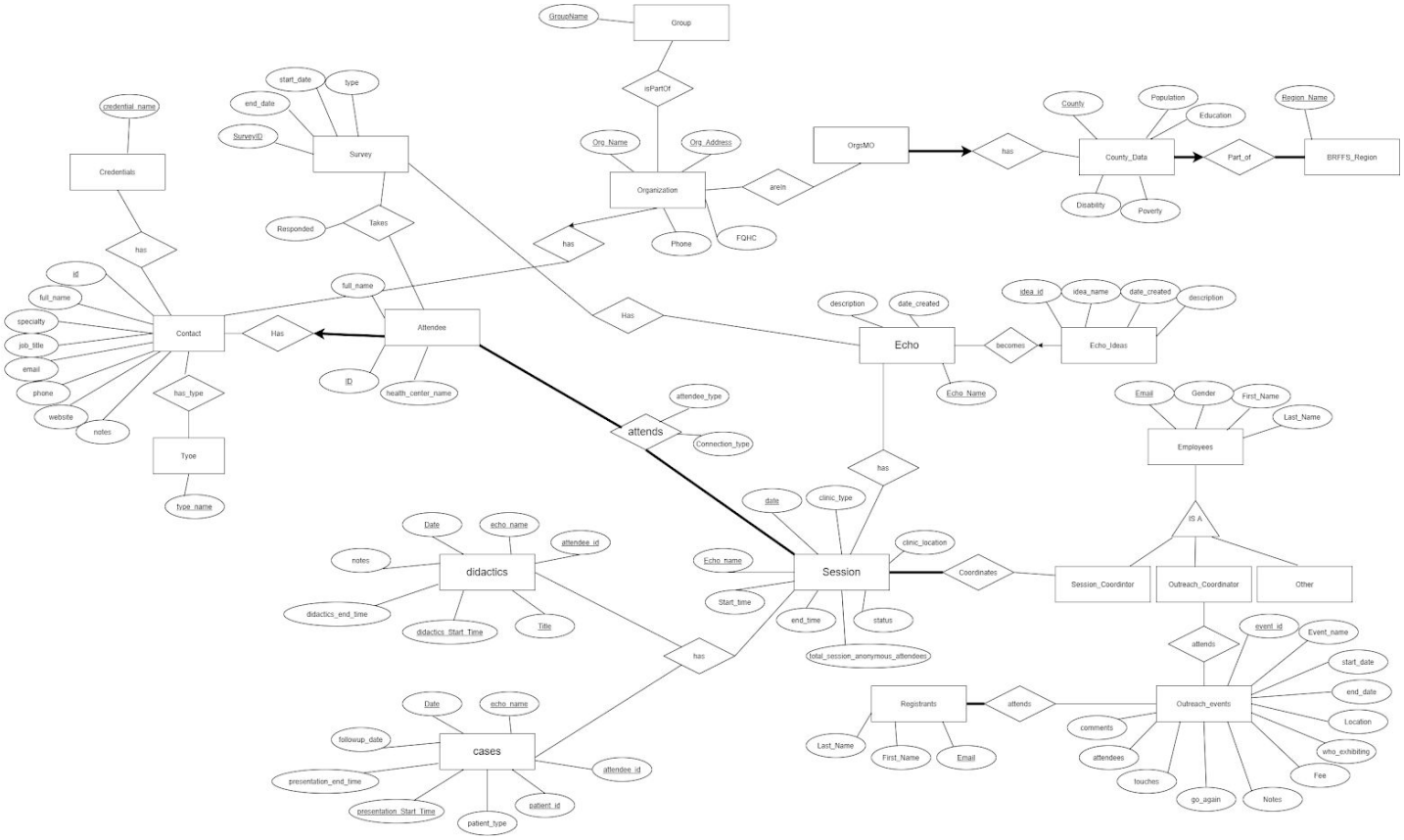
Our final project uses Postgresql and Amazon EC2 services. The final product will be used for both commercial and research use. The website will be transferred to Missouri Telehealth Network’s domain and will be displayed/sent to other ECHO programs around the United States, such as New Mexico. The database component will be utilized for years to come as they continue to accumulate data. This will also be used for research purposes, from analyzing counties to quality of health due to the Show-Me ECHO program.

Link: http://ec2-3-16-114-247.us-east-2.compute.amazonaws.com/index.php

Username: test

Password: pass

ERD:


GroupName
Group
isPartOf
start_date
type
end_date
SurveyID
Survey
credential_name
Credentials
has
Org_Name
Org_Address
Organization
arein
OrgsMO
has
County
Population
Education
County_Data
Part_of
Region_Name
BRFFS_Region
Disability
Poverty
Responded
Takes
has
Phone
FQHC
id
full_name
specialty
job_title
email
phone
website
notes
Contact
Has
full_name
Attendee
ID
health_center_name
has_type
Tyoe
type_name
Has
description
date_created
Echo
Echo_Name
becomes
idea_id
idea_name
date_created
description
Echo_Ideas
attendee_type
attends
Connection_type
has
Email
Gender
First_Name
Last_Name
Employees
IS A
Session_Coordintor
Outreach_Coordinator
Other
Coordinates
date
clinic_type
clinic_location
Echo_name
Session
Start_time
end_time
status
total_session_anonymous_attendees
Date
echo_name
attendee_id
notes
didactics
didactics_end_time
didactics_Start_Time
Title
has
Registrants
attends
Last_Name
First_Name
Email
attends
event_id
Event_name
start_date
end_date
Location
who_exhibiting
Fee
Outreach_events
comments
attendees
touches
go_again
Notes
Date
echo_name
followup_date
cases
presentation_end_time
presentation_Start_Time
patient_type
patient_id
attendee_id

DDL:

Table	DDL	Constraint Explanation
Contact	CREATE TABLE Contact(ID SERIAL PRIMARY KEY, Full_name TEXT, First_name TEXT, Last_name TEXT, Job_title TEXT, Specialty TEXT, Health_Center_Name TEXT, Street_address TEXT, Mobile TEXT, Fax TEXT, Other_phone TEXT, Email TEXT, Other_Email TEXT, Website TEXT, Notes TEXT, FOREIGN KEY (Health_Center_Name, Street_address) REFERENCES Organizations(name, streetaddress))	ID is a unique number given to every contact Each contact is part of 0 or 1 organizations
Attendee	CREATE TABLE Attendee(ID SERIAL PRIMARY KEY, Full_name TEXT, Health_Center_Name TEXT, FOREIGN KEY (ID) REFERENCES Contact(ID));	ID is a foreign key of contact since each attendee must be in the contact table, but a contact doesn't have to be an attendee
Attends	CREATE TABLE Attends(ID SERIAL, Echo_name TEXT, Date TIMESTAMP, Connection_type TEXT, attendee_type TEXT, PRIMARY KEY(ID, Echo_name, Date, Connection_type, attendee_type), FOREIGN KEY (ID) REFERENCES	The primary key is id, echo_name, date, connection_type, and attendee type since those combined attributes will be unique to every row in attends. This allows for more an attendee to participate in 0 or many sessions and a

	Attendee, FOREIGN KEY (Echo_name,Date) REFERENCES Sessions(Echo_name, Date))	session to have 0 or many attendees. Attends connects attendee to session, so it's foreign keys are the Primary keys of sessions and attendee.
Type	CREATE TABLE Type(Type TEXT PRIMARY KEY)	This table's key is Type. This table exists to make sure a contact can only have types listed in this table.
Has_Type	CREATE TABLE Has_Type(Type TEXT, ID SERIAL, PRIMARY KEY(Type, ID), FOREIGN KEY (Type) REFERENCES Type(Type), FOREIGN KEY (ID) REFERENCES Contact(ID))	The Primary key is a combination of the contact table and type table to allow for a contact to have 0 or many types. The foreign keys connect type to contact.
Credentials	CREATE TABLE Credentials(credential_name TEXT PRIMARY KEY)	This table's key is Credential. This table exists to make sure a contact can only have credentials listed in this table.
Has_credential	CREATE TABLE Has_credential(Credential_name VARCHAR(100), ID SERIAL, PRIMARY KEY(Credential_name, ID), FOREIGN KEY (ID) REFERENCES Contact(ID), FOREIGN KEY (Credential_name) REFERENCES Credentials(Credential_name));	The Primary key is a combination of the contact table and credential table primary keys to allow for a contact to have 0 or many types. The foreign keys connect credential to contact
Survey	CREATE TABLE Survey(SurveyID SERIAL PRIMARY KEY, Start_date TIMESTAMP, End_date TIMESTAMP, TYPE TEXT, Echo_name TEXT, FOREIGN KEY (Echo_name)	The primary key is a unique number assigned to each different survey. The foreign key to echo_name in table echo exists to create a 0 to many relationship. Each survey

	REFERENCES Echos(Echo_name));	belongs to one ECHO, but an ECHOcan have many surveys.
taken_survey	CREATE TABLE Taken_Survey(SurveyID SERIAL, ID SERIAL, Responded BOOLEAN, PRIMARY KEY(SurveyID,ID), FOREIGN KEY (ID) REFERENCES Attendee(ID), FOREIGN KEY (SurveyID) REFERENCES Survey(SurveyID));	The primary key is a combination to connect the survey table to attendees. This allows for an attendee to take 0 or many surveys and for a survey to have 0 or many people take the survey.
Cases	CREATE TABLE cases (date date, echo_name text, full_name text, health_center_name text, patient_id text, patient_type text, presentation_start_time text, presentation_end_time text , followup_date text , contact_id integer, PRIMARY KEY (date, echo_name, full_name, patient_id, presentation_start_time), FOREIGN KEY (echo_name) references echo(echo_name)), FOREIGN KEY (echo_name) references echo(echo_name)) REFERENCES public.sessions (echo_name, date) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION	The Primary Key is a combination of date,echo_name from sessions and full_name,patient_id and presentation_start_time. Each session can have 0 or many cases
didactics	CREATE TABLE public.didactics (date date,	The Primary Key is date,echo_name from sessions and full_name,title

	echo_name text ,pg_catalog."default" NOT NULL, full_name text COLLATE pg_catalog."default" NOT NULL, title text COLLATE pg_catalog."default" NOT NULL, didactic_start_time character varying(50) COLLATE pg_catalog."default" NOT NULL, didactic_end_time character varying(50) COLLATE pg_catalog."default", notes text COLLATE pg_catalog."default", id integer, CONSTRAINT didactics_presents_1_pkey PRIMARY KEY (date, echo_name, full_name, title, didactic_start_time)	and didactic_start_time. Each session can have 0 or many didactics
Echos	CREATE TABLE public.echos (echo_name text COLLATE pg_catalog."default" NOT NULL, CONSTRAINT echo_1_pkey PRIMARY KEY (echo_name))	The Primary key is echo_name because each echo_name is distinct. An echo is connected to session. An echo can have 0 or many sessions.
Sessions	CREATE TABLE public.sessions (date date NOT NULL, echo_name text COLLATE pg_catalog."default" NOT NULL, start_time character varying(100) COLLATE pg_catalog."default", end_time character varying(100) COLLATE pg_catalog."default", total_session_anonymous_attendees character varying(50) COLLATE pg_catalog."default", clinic_status text COLLATE pg_catalog."default", clinic_location text COLLATE pg_catalog."default",	The Primary Key is date, echo_name because an echo can occur once in a day.

	clinic_type text COLLATE pg_catalog."default", CONSTRAINT session_1_pkey PRIMARY KEY (date, echo_name), CONSTRAINT session_1_echo_name_fkey FOREIGN KEY (echo_name) REFERENCES public.echos (echo_name) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION)	
BRFSS	CREATE TABLE brfss (BRFSS_ID int, RegionName varchar(30), PRIMARY KEY (BRFSS_ID));	The PK is the Behavioral Risk Factor Surveillance System (BRFSS) region number (1-7). These are groups of counties in MO.
CountyData	CREATE TABLE CountyData(FIPS char(5), State varchar(20), County varchar(30), BRFSS_ID int, LandSQMI real, AreaSQMI real, TotalPop2010 int, pctOver62 real, pctMales real, pctFemales real, pctWhite1 real, pctBlack1 real, pctIndian1 real, pctAsian1 real, pctHawnPI1 real, pctOther1 real, AvgHHInc real, AvgHHSocSec real, pctPoort real, pctPublicTrans real, pctBachelorsormore real, pctBroadbandInt real, pctNoInternet real, pctNoVehicles real,	This table has various demographic, economic, and health factors/rankings for each county in Missouri. The PK is County because it is strictly in MO. If they wanted to expand it in the future to more states, FIPS would be a good PK.

	SEF_Rank int, HO_Rank int, HF_Rank int, pctUninsured real, NumofPCP int, pctFluVacc real, pctSomeCollege real, pctUnemployed real, LifeExpect real, pctNoHealthyFood real, pctRural real, PRIMARY KEY (County), FOREIGN KEY (BRFSS_ID) REFERENCES brfss(BRFSS_ID) ON DELETE NO ACTION);	
Organizations	CREATE TABLE Organizations(Name varchar(100), StreetAddress varchar(200), City varchar(100), State varchar(50), ZipCode varchar(20), County varchar(200), Country varchar(20), Phone varchar(50), PRIMARY KEY (Name, StreetAddress));	This table contains all of the organizations that contacts are in. The PK is Name, StreetAddress because multiple organizations may have the same name and different organizations may be in the same location.
OrgsMO	CREATE TABLE OrgsMO(Name varchar(100), StreetAddress varchar(200), City varchar(100), ZipCode varchar(20), County varchar(30), FQHC boolean, PRIMARY KEY (Name, StreetAddress), FOREIGN KEY (Name, StreetAddress) REFERENCES Organizations(Name, StreetAddress) ON DELETE CASCADE, FOREIGN KEY (County) REFERENCES CountyData(County));	This table includes the organizations in Missouri. The PK is Name, StreetAddress because it references Organization. It is on DELETE CASCADE because if it's removed from the overall Organization table, it should be removed from OrgsMO.
Groups	CREATE TABLE Groups(	This table groups multiple

	GroupName varchar(200), Name varchar(100), StreetAddress varchar(200), PRIMARY KEY (GroupName), FOREIGN KEY (Name, StreetAddress) REFERENCES OrgsMO (Name, StreetAddress) ON DELETE CASCADE);	organizations together. For example, Access Family Care - Anderson and Access Family Care - Aurora are both a part of the group "Access Family Care." The PK is the GroupName.
echo_ideas	CREATE TABLE echo_ideas(idea_id SERIAL PRIMARY KEY, idea_name VARCHAR(100), date_created DATE, description TEXT);	This table contains all of the ideas that the ECHO team is brainstorming. It has a primary key of a serial, the idea's name, date created, and any notes or description.
Employees	CREATE TABLE employees(email VARCHAR(40) PRIMARY KEY, last_name VARCHAR(20), first_name VARCHAR(20), gender CHAR(1), CHECK(gender='M' OR gender='F' OR gender='O'));	This table holds employee information. It has contact information, name, gender, and a check to make sure that the gender constraint is met. It is the superclass of an IS-A clause, referenced by session_coordinator, outreach_coordinator, and other_employee
session_coordinator	CREATE TABLE session_coordinator(email VARCHAR(40) PRIMARY KEY, FOREIGN KEY(email) REFERENCES employees(email));	This table is a subclass of an IS-A clause, referencing the employees table. This table was decomposed from BCNF for efficient querying and the client's request for future adaptation to the table.
outreach_coordinator	CREATE TABLE outreach_coordinator(email VARCHAR(40) PRIMARY KEY, FOREIGN KEY(email) REFERENCES employees(email));	This table is a subclass of an IS-A clause, referencing the employees table. This table was decomposed from BCNF for efficient querying and the client's request for future adaptation to the table.for simplistic querying.
other_employe	CREATE TABLE other_employee(	This table is a subclass of an

e	email VARCHAR(40) PRIMARY KEY, FOREIGN KEY(email) REFERENCES employees(email));	IS-A clause, referencing the employees table. This table was decomposed from BCNF for efficient querying and the client's request for future adaptation to the table.
outreach_events	CREATE TABLE outreach_events(event_id SERIAL PRIMARY KEY, event_name VARCHAR(300), date VARCHAR(30), location VARCHAR(200), who_exhibiting VARCHAR(50), fee INTEGER DEFAULT NULL, notes TEXT, go_again VARCHAR(10), touches INTEGER DEFAULT NULL, attendees INTEGER DEFAULT NULL, comments TEXT);	This table holds information for all of the outreach events hosted to market ECHOs. These events can be in the past or future, their key is a serial and it holds information such as the event name, date, location, fees, notes, and reflection information on who coordinated, whether or not they should go again, touches, attendees, and comments.
login_credentials	CREATE TABLE login_credentials (username VARCHAR(20) PRIMARY KEY, password VARCHAR(255));	This is the table used for login credentials. It holds a username and password. When inserting data into this table, the md5 encoding method is used to insure security.

Useful Queries/Analytics

Natural Language & Justification	SQL
What locations have the most cases (top 20)? This shows which locations are the most active.	SELECT COUNT(*), clinic_location FROM sessions GROUP BY clinic_location ORDER BY clinic_location LIMIT 20
How many contacts does each organization have? This shows which organizations have the most people in the system.	SELECT COUNT(*), o.name FROM Organizations o, contact c WHERE o.name = c.health_center_name AND o.streetaddress = c.street_address AND o.name != 'None' GROUP BY o.name ORDER BY COUNT(*) DESC LIMIT 15;
Top ten people who have presented cases This is important to identify because then they can assess who has been stepping up. When we demo-ed this analytic, they already knew who the #1 would be and it confirmed it.	SELECT full_name, count(*) FROM CASES GROUP BY full_name ORDER BY count(*) DESC LIMIT 10
Which type has presented the most cases? This is important to show because they want more attendees to present cases rather than facilitators to encourage participation. This was one of the key things they wanted to know.	SELECT attendee_type, count(*) FROM cases c, attends a, sessions s WHERE c.date=s.date AND s.echo_name=a.echo_name AND s.date=a.date AND c.contact_id=a.id GROUP BY attendee_type ORDER BY COUNT(*) DESC
Amount of people that participate in sessions from each county This shows which counties in Missouri have the most attendees and the reach.	SELECT cd.county, count(DISTINCT(c.id)) AS Number_Of_People FROM attends a, attendee at, contact c, organizations o, countydata cd

	WHERE a.id=at.id AND c.id=at.id AND o.name=c.health_center_name AND o.streetaddress=c.street_address AND cd.county = o.county GROUP BY cd.county ORDER BY count(DISTINCT(c.id)) DESC LIMIT 20;
What percent of contacts have attended at least one session? They wanted to see how many people registered vs how many actually have attended.	SELECT ((SELECT 1.0*count(DISTINCT a.id) FROM attends a) / (SELECT count(*) FROM contact))AS Percent
Number of FQHCS in Missouri that has had at least one person attending an ECHO session This is relevant because they want to know more about their impact on FQHCs	SELECT COUNT(DISTINCT (m.name, m.streetaddress)) FROM OrgsMO m, organizations o, contact c, attends a WHERE o.name=c.health_center_name AND o.streetaddress=c.street_address AND c.id=a.id AND m.name = o.name AND o.streetaddress=m.streetaddress AND m.fqhc = 'TRUE';
Number of Contacts in Each State They reach multiple states so it would be nice to know their reach.	SELECT state, count(*) FROM contact c, organizations o WHERE o.name=c.health_center_name AND o.streetaddress=c.street_address AND LENGTH(state)<3 AND LENGTH(state)>0 GROUP BY o.state
Number of contacts in each county in Missouri This map is important because it allows them to visualize their reach in Missouri, especially	SELECT o.county, cd.fips, count(*) FROM countydata cd LEFT JOIN organizations o ON cd.county=o.county AND o.state='MO' LEFT JOIN contact c ON o.name=c.health_center_name AND o.streetaddress=c.street_address

in the Bootheel which is an area of concern for them.	GROUP BY o.county, fips
What credentials a contact has. This allows them to find which providers can prescribe medications and which ones are academics.	SELECT full_name, o.name, UPPER(credential_name) AS Credential FROM contact c FULL OUTER JOIN has_credential h ON h.id=c.id, Organizations o WHERE o.name=c.health_center_name AND o.streetaddress=c.street_address AND h.credential_name!='None' AND LOWER(c.full_name)~LOWER('$full_name')
Find total number of FQHCs This shows how many FQHCs are in Missouri to be used in other queries.	SELECT COUNT(*) FROM OrgsMO WHERE fqhc='true';
Find the rate of touches to attendees at Outreach Events This can show the effectiveness of outreach events.	SELECT round(((avg(touches) / avg(attendees)) *100)) AS rate FROM outreach_events
Amount of Sessions Over Time By Month This shows their trend between each month - which ones don't hold as many sessions?	SELECT TO_CHAR(date,'Mon') as mon, EXTRACT(year from date) as yyyy, date_trunc('month',date) as month, count(*) FROM SESSIONS GROUP BY month, mon, yyyy ORDER BY month ASC
Amount of Attendees to sessions By Month This shows which months attendees are more likely to tune in and attend.	SELECT TO_CHAR(s.date,'Mon') as mon, EXTRACT(year from s.date) as yyyy, date_trunc('month',s.date) as month, count(*) FROM SESSIONS s, Attends a WHERE s.echo_name=a.echo_name AND s.date=a.date GROUP BY month, mon, yyyy ORDER BY month ASC

Top 20 Outreach Events with the most attendees This is important because it can be used in the future for their outreach coordinators to focus on those events to get their maximum exposure.	SELECT event_name, attendees FROM outreach_events ORDER BY attendees DESC LIMIT 20;
Ten people who coordinated the most outreach events This shows who have done the most in outreach and can track who went where.	SELECT who_exhibiting, count(*) FROM outreach_events WHERE who_exhibiting != 'None' GROUP BY who_exhibiting ORDER BY count(*) DESC LIMIT 10
Amount of cases presented over time Attendee vs Hub Team This shows how many cases are being presented by which type over time. They wanted to know the breakdown.	Hub team: SELECT TO_CHAR(s.date,'Mon') as mon, EXTRACT(year from s.date) as yyyy, date_trunc('month',s.date) as month, count(*) FROM cases c, attends a, sessions s WHERE c.date=s.date AND s.echo_name=a.echo_name AND s.date=a.date AND c.contact_id=a.id AND attendee_type='Facilitator' GROUP BY month, mon, yyyy ORDER BY month ASC Participant: SELECT TO_CHAR(s.date,'Mon') as mon, EXTRACT(year from s.date) as yyyy, date_trunc('month',s.date) as month, count(*) FROM cases c, attends a, sessions s WHERE c.date=s.date AND s.echo_name=a.echo_name AND s.date=a.date AND c.contact_id=a.id AND attendee_type=’Attendee’ GROUP BY month, mon, yyyy

	ORDER BY month ASC
Counties with no attendees This shows the 3 counties in MO that don't have anyone attending. This is interesting because it shows which ones they need to do more outreach to.	SELECT county FROM countydata EXCEPT SELECT cd.county FROM countydata cd, organizations o, contact c, attends a WHERE cd.county = o.county AND o.name=c.health_center_name AND o.streetaddress=c.street_address AND c.id=a.id;
Bottom 20 Counties' data This is an interesting query because it highlights the counties that are not actively attending ECHO sessions, but have certain rural, uninsured, and poor percentages.	SELECT A.county, A.pctpoort, A.pctuninsured, A.pctrural, 100.0*B.countAttendees/A.totalpop2010 AS Percent FROM (SELECT county, pctpoort, pctuninsured, pctrural, totalpop2010 FROM countydata ORDER BY totalpop2010 DESC) A JOIN (SELECT cd.county, count(DISTINCT(at.id)) AS countAttendees FROM countydata cd LEFT JOIN organizations o ON cd.county=o.county LEFT JOIN contact c ON o.name=c.health_center_name AND o.streetaddress=c.street_address LEFT JOIN attendee at ON at.id=c.id LEFT JOIN attends a ON a.id=at.id GROUP BY cd.county ORDER BY count(DISTINCT(at.id)) DESC) B ON A.county=B.county ORDER BY Percent ASC LIMIT 20

Number of Organizations that have attendees This shows how many organizations out of all actually have attended sessions.	SELECT COUNT(DISTINCT(o.name, o.streetaddress)) FROM Organizations o, contact c, attends a WHERE o.name=c.health_center_name AND o.streetaddress=c.street_address AND c.id=a.id
Number of FQHCs in Each County This allows for interpretation on what counties have the most FQHCs for geospatial analysis.	Select cd.county, cd.fips, COUNT(o.name) FROM countydata cd LEFT JOIN orgsmo o ON cd.county=o.county AND fqhc=true GROUP BY cd.county, cd.fips
Number of FQHCs in each county that have participated in a session This query allows the client to see where the FQHCs are that are participating in ECHO and where the FQHCs are that don't attend sessions.	SELECT cd.county, cd.fips, COUNT(DISTINCT(om.name)) FROM countydata cd LEFT JOIN orgsmo om ON cd.county=om.county AND FQHC=true WHERE om.name IN(SELECT DISTINCT(o.name) FROM Organizations o, contact c, attends a WHERE o.name=c.health_center_name AND o.streetaddress=c.street_address AND c.id=a.id) GROUP BY cd.county, cd.fips
The Percent of attendees by population of a county who have attended an ECHO session. This is an interesting query because it allows the client to interpret their reach in each county. This query normalizes each county, not allowing skewed results if a county has a higher population.	SELECT A.county, A.pctpoort, A.pctuninsured, A.pctrural, 100.0*B.countAttendees/A.totalpop2010 AS Percent FROM (SELECT county, pctpoort, pctuninsured, pctrural, totalpop2010 FROM countydata ORDER BY totalpop2010 DESC) A JOIN (SELECT cd.county, count(DISTINCT(at.id)) AS countAttendees FROM countydata cd

	LEFT JOIN organizations o ON cd.county=o.county LEFT JOIN contact c ON o.name=c.health_center_name AND o.streetaddress=c.street_address LEFT JOIN attendee at ON at.id=c.id LEFT JOIN attends a ON a.id=at.id GROUP BY cd.county ORDER BY count(DISTINCT(at.id)) DESC) B ON A.county=B.county ORDER BY Percent DESC LIMIT 20
No. of Sessions for each Attendees Returns the number of sessions that each attendee has attended. Allows interpretation of attendee's activity.	select at.health_center_name,c.count,c.id,c.full_name from(select count(*) as count,at.id as id,at.full_name as full_name from attendee at,attends a where at.id=a.id group by at.id,at.full_name order by count DESC) as c,attendee at where at.id = c.id
Search Query for Contacts Allows users to search for specific data from the contacts table.	select * from contact WHERE full_name LIKE '$word' OR health_center_name LIKE '$word' OR job_title LIKE '$word'
Contacts per health center This highlights what health centers they have a good reach in and which health centers they should pursue more connections in.	select count(*) as count,health_center_name from contact group by health_center_name order by count DESC
Dynamic generation of attendees for a particular Session Allows viewer to see who attended what session dynamically.	select a.id,a.connection_type,a.attendee_type,at.full_na me,at.health_center_name from attends a INNER JOIN attendee at ON a.id = at.id where a.echo_name='$echo_name' and a.date='$session_date'

Normalization

We created our tables on the basis of the CSV files provided to us by ECHO Team. In our database, we created tables that broke the CSV table files into multiple tables and we decomposed the CSV table files to eliminate problems of bad design like anomalies, inconsistencies, and unnecessary redundancy. We ensured any lossless join and dependency preservation. After creation of our tables, we found that for some tables, we had data redundancy. We settled on it because it allowed faster results. For instance, we could have put an attribute of boolean type called has_attended on the contacts table to know if he attended at least a session, but we created another table called attendee that has the attendee id, name and health center name as attributes as our queries required these attributes. This allowed us to get rid of more than 2000 rows from the contacts table. The horizontal decomposition is lossless and preserves the dependency. All of our tables are in BCNF allowing efficient queries while using less memory relative to the large data we were provided with.

Indexing Selection

Index: CREATE INDEX index_date ON Sessions (date);

Our clients were very interested in looking at data during specific points of time so we created an index on the date attribute in sessions to allow for quicker queries on this table.

Index: CREATE INDEX index_name ON contact (last_name, first_name); When displaying contact data or when searching for specific contacts, we wanted to display the names by descending order by last name. To speed up these queries we added an index on the last_name attribute of contacts and then had it indexed by first name.

Index: CREATE INDEX index_org_county ON organizations (county);

Many of our location based queries rely on where an organization is located by county. Creating an index in our organizations table on county will increase the speed of queries where we group by county.

The Attends table is our largest table with 30,000 rows. This table was naturally indexed through the primary keys to increase query speed on this table.

Optimization and Tuning

Our database design used horizontal decomposition in several places in order to separate data that would be queried more often. For example, our Contacts table was decomposed into a more specific Attendee table that contained only Contacts that had attended a session before. We chose this design since queries on attendees would be distinct from a query about contacts as a whole. Additionally, more queries would revolve around attendees, so in order to make queries more efficient, we put them in a separate table. Without this design, each time a query would need to be done on an attendee, it would first need to check for a contact's id in the entire session table. Another time we applied horizontal decomposition was creating a table of organizations that were specifically only in Missouri. Since ECHO predominately functions inside of Missouri they have a special interest in queries that revolve around only organizations that are in Missouri. By creating a separate table, queries on organizations in Missouri could be more efficient without needing to check which state the organization was in each time a query was run. This also speeds it up because then we don't have to go through all 2983 organizations.

Security Setting

To begin, we submitted an IRB to take precautions because we are handling some medical data (not specific patients though). In response, IRB decided that "the project has been determined to be a quality improvement activity not requiring IRB review." However, we still created a view to hide specific attendee/contact names because some could be tied back to specific patients if they were to attend and present cases. [CREATE VIEW justNames (id, job_title, health_center_name) AS SELECT id, job_title, health_center_name FROM Contact;]. In the future, the database administrators can allow others to use this view when displaying data. They should most likely be the only ones who can CREATE, ALTER, and DROP tables alongside the other privileges. Others, such as the Show-Me ECHO coordinators, may only have INSERT and DELETE tuples for their specific ECHO or strictly Outreach events. Others may also be outreach coordinators who are presenting the data to other ECHOs or interested providers and could be granted SELECT privileges to display the information. (GRANT SELECT ON Attendees TO Lincoln, GRANT INSERT ON Sessions TO ECHO_Coordinator). The Bell-LaPadula Model can also be implemented based on their role at Show-Me ECHO. Dr. Sheets, Dr. Becevic, Emmanuelle, and Becky could have higher security classes than other staff in case they want to keep things confidential, such as future ECHO ideas.

Other Topics

We have created a trigger for them to use in the future when they add to the Organizations table when new attendees register. This would make sure that it connects smoothly between OrgsMO and Organizations to continue quick queries on strictly Missouri organizations.

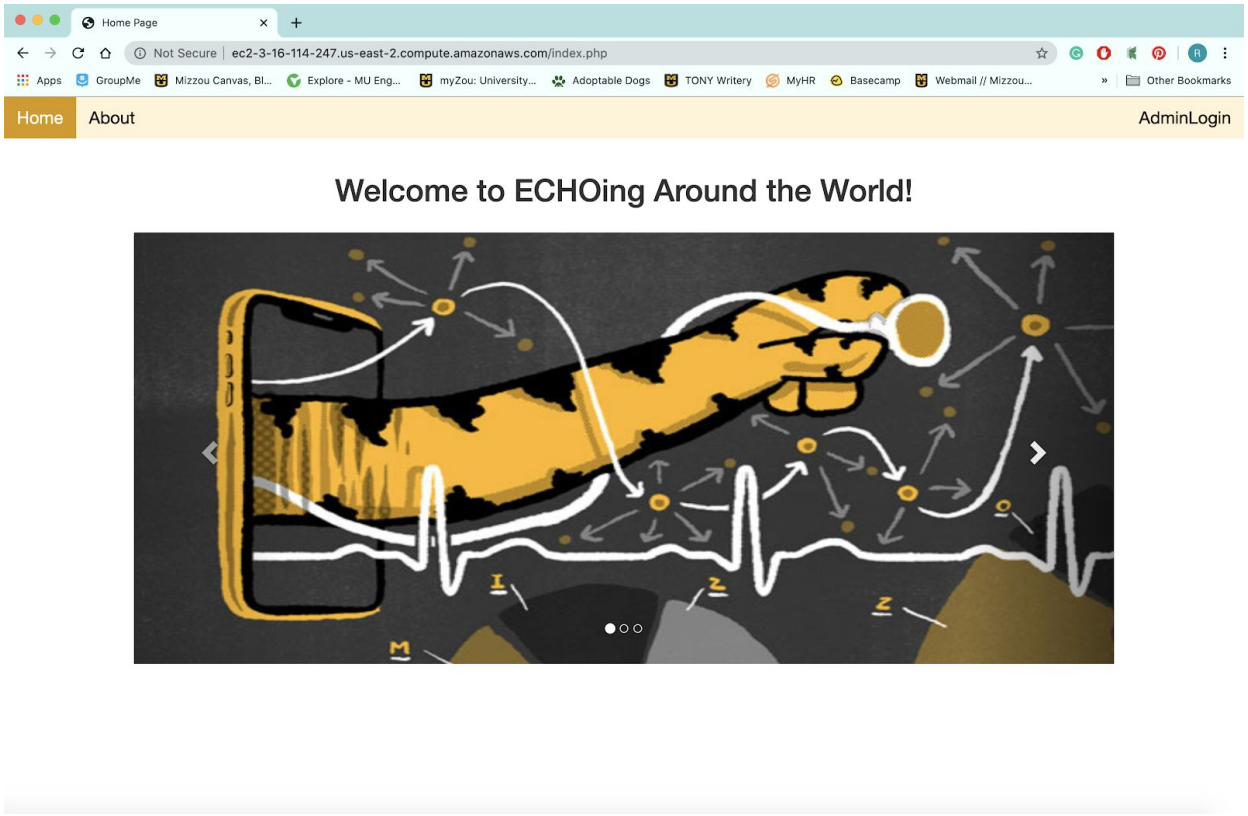
```
CREATE TRIGGER moveOrgsMO
   AFTER INSERT ON Organizations
   FOR EACH ROW
   EXECUTE PROCEDURE add_OrgMO();


CREATE FUNCTION add_OrgMO()
   RETURNS trigger AS
$$
BEGIN
   IF NEW.state = 'MO' THEN
      INSERT INTO OrgsMO (name, streetaddress, city, zipcode, county)
      VALUES (NEW.name, NEW.streetaddress, NEW.city, NEW.zipcode, NEW.county);
   END IF;
   RETURN NOW;
END;
```

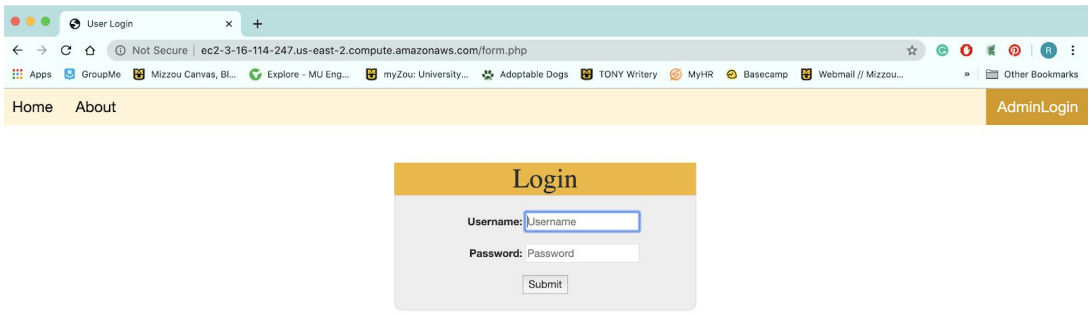
User's Manual

Our website is very straightforward for the end user.

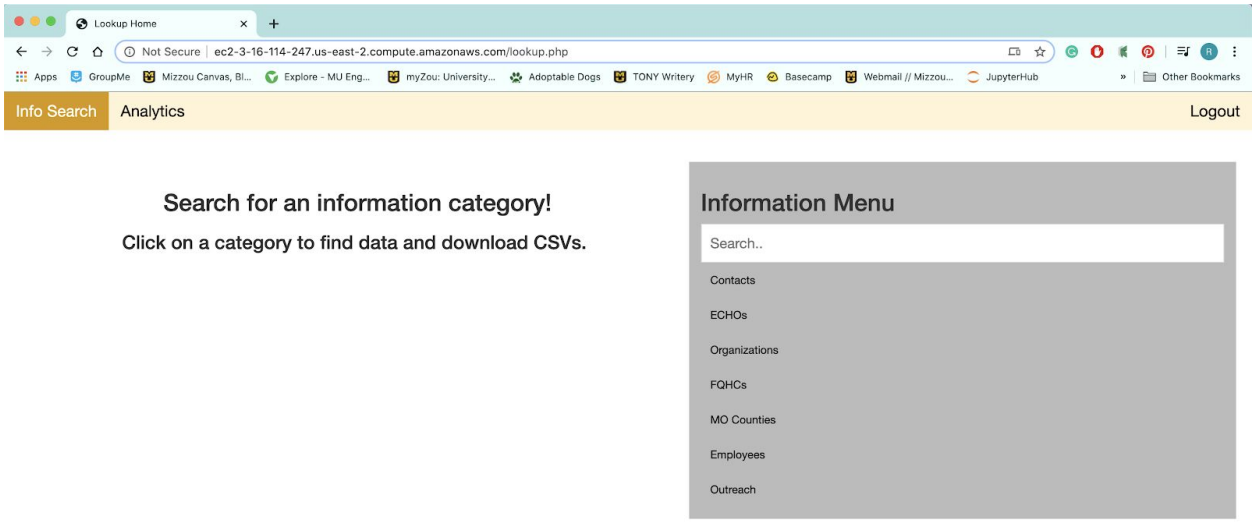
- The first page is the welcome page that consists of a horizontal navigation bar (Home by default), an About page, and AdminLogin in the top right corner.



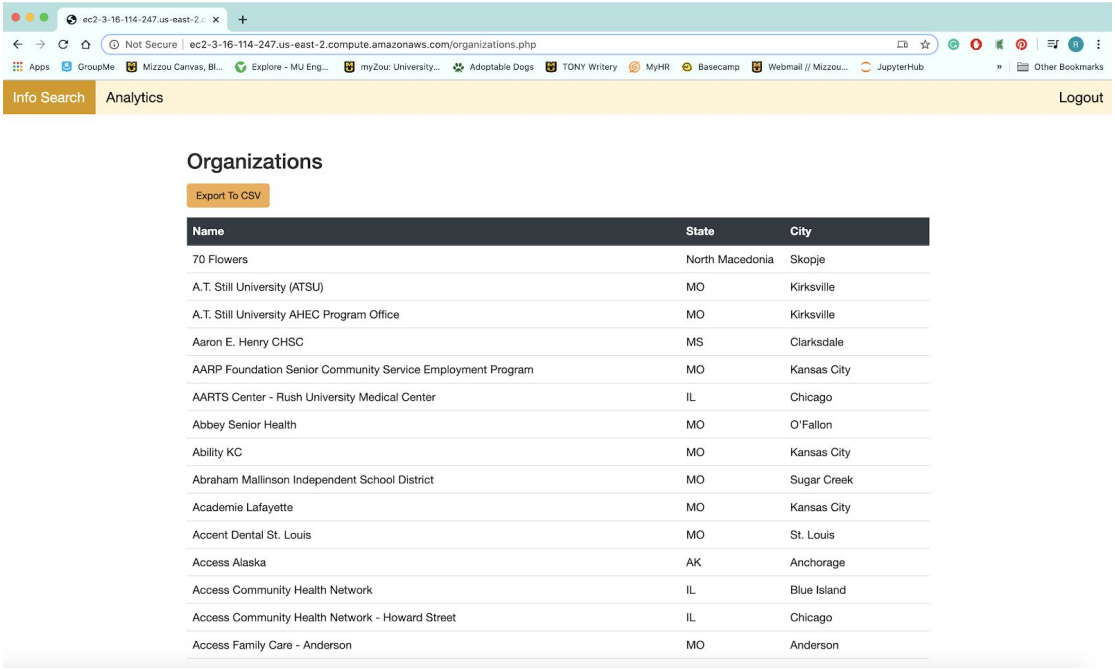
- The AdminLogin page allows the user to login by entering the username and password and clicking on submit.



- Once the user logs in, they will land on the Information Search page that has a new horizontal navigation bar(present throughout the website once logged in). There are two options in the top horizontal bar: InfoSearch and Analytics with the default as InfoSearch. InfoSearch has a vertical nav bar (Information Menu) with 7 categories. Each category navigates a user to that particular page.

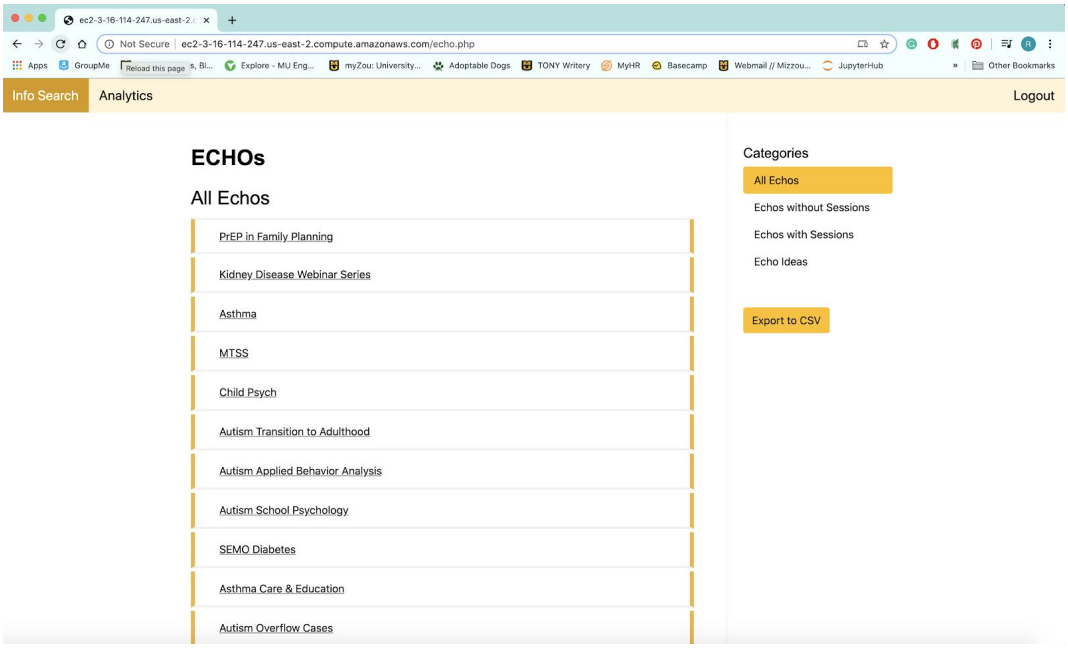


- All of the InfoSearch category pages consist of all the data of that category and shows some important data in view components. The user can click on the Export to csv button provided to download CSVs for those pages.

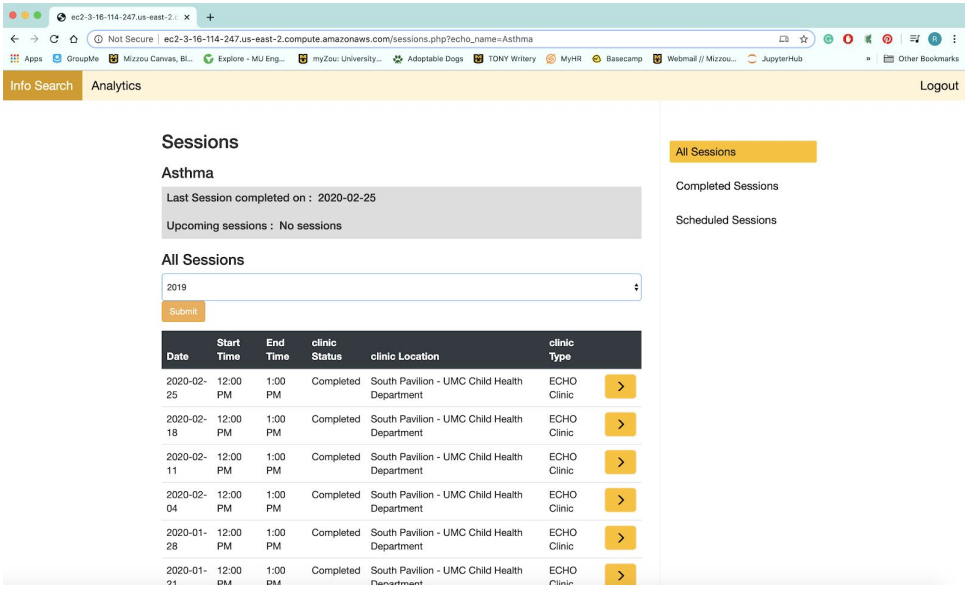


Name	State	City
70 Flowers	North Macedonia	Skopje
A.T. Still University (ATSU)	MO	Kirksville
A.T. Still University AHEC Program Office	MO	Kirksville
Aaron E. Henry CHSC	MS	Clarksdale
AARP Foundation Senior Community Service Employment Program	MO	Kansas City
AARTS Center - Rush University Medical Center	IL	Chicago
Abbey Senior Health	MO	O'Fallon
Ability KC	MO	Kansas City
Abraham Mallinson Independent School District	MO	Sugar Creek
Academie Lafayette	MO	Kansas City
Accent Dental St. Louis	MO	St. Louis
Access Alaska	AK	Anchorage
Access Community Health Network	IL	Blue Island
Access Community Health Network - Howard Street	IL	Chicago
Access Family Care - Anderson	MO	Anderson

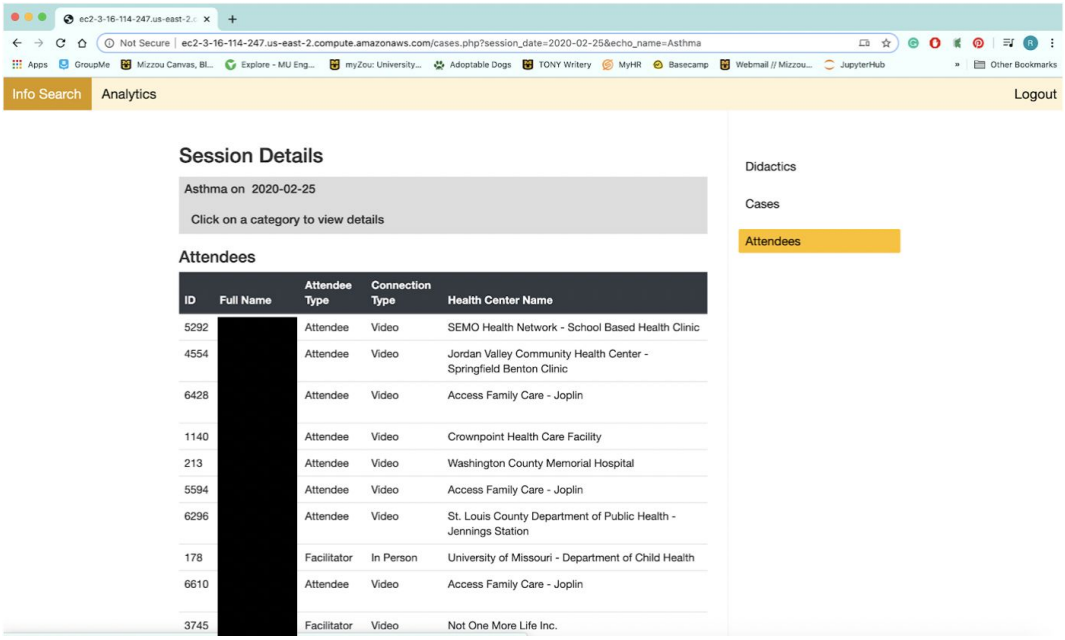
- There are also child pages within the ECHOs page which the user can navigate by clicking on that data. The data is shown either as a link, button, or arrow. When the user first clicks on ECHOs, all ECHOs are shown and the user can click on a specific ECHO, see ECHOs without Sessions, ECHOs with Sessions, or ECHO ideas.



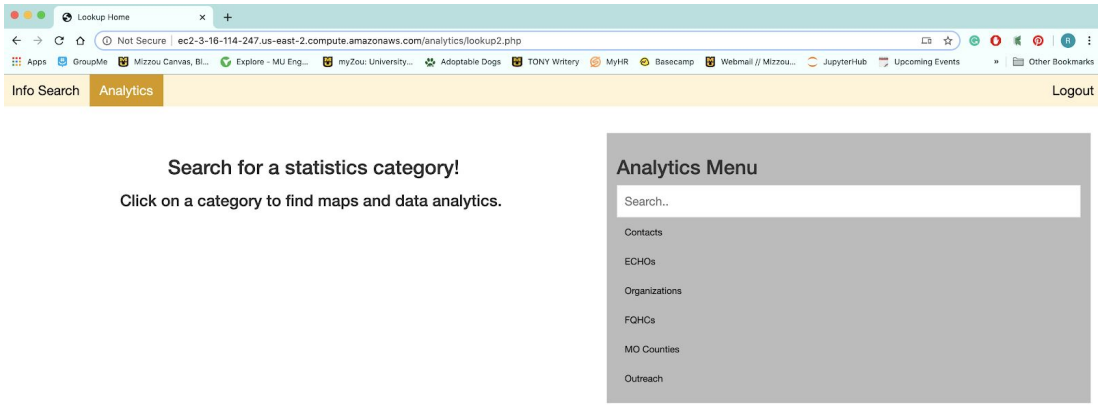
- When the user clicks on a specific ECHO, they see the last session in the database. They can then see All Sessions, Completed Sessions, and Scheduled Sessions.



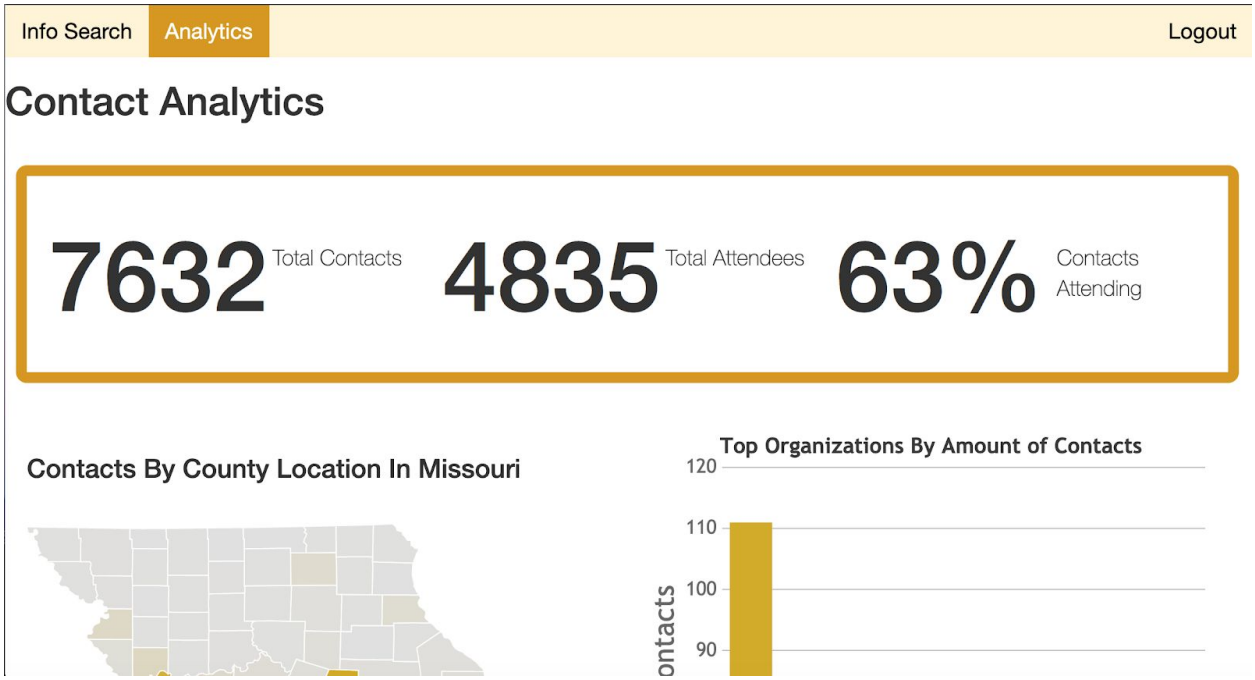
- From there, the user can go into a specific session by clicking on the black and gold arrow. Here, they will see Didactics, Cases Presented, and Attendees at that session.



- The Analytics page also has a vertical navigation bar (Information Menu) with 6 categories. Each category navigates a user to analytics about that particular category.



- All of the Analytics category pages consist of useful queries that present data visualizations in the form of graphs (bar charts, pie charts, Maps, search bars etc.). At the top of each page, there are general statistics and queries that result in numbers for the user to conceptualize each category. The user can move the cursor over the charts and maps to view data for any point.



- The user can log out by clicking the logout present in the top right corner throughout the website. This will take them back to the home page.