**A: Research Question**

The research question I am investigating in this project is, “What are the ages of customers that use phone services?” The question has the opportunity to investigate trends and patterns among customers which could be used to improve the profitability of a company. With this information, companies could potentially promote cable services heavier to people based on their age and demographics.

**A1: Identifying Data**

The research question examines three columns Customer\_id,Age, and Phone.. In the Services table imported and created, the customer\_id and phone fields are primarily going to be examined. The customer\_id column is available in both the customer table provided in the churn database and it's also available in the services.csv file we are importing into the database. Since it's available in both, the customer\_id field will be what is used to join the data together. The age field to determine a customer’s age will be available in the customers table in the churn database aws well. The phone field needed to observe if customers purchased the phone services is available in the add-on CSV file that will be imported into the database services.CSV. To answer the research question, the columns will be examined to form an analysis on the ages by joining the services table I created with the customer table provided so all the information is easily accessible and observable.From here it will be In the customer table that was pre existing, we will be examining the age column. The Customer id field will be used to join the two tables.

**B.Logical Data Model**

The add-on file for this assignment that I used was the services.csv file. I only needed the phone column from this file, however I imported the entire dataset into a table called services\_csv. The primary key for this table was the customer\_id, so I could easily link it to the customer table. This relates approproiatel;y as there can be only one customer ID. This way, all attributes associated with a particular customer ID will be uniform and easy to track in the table.

As well as being a primary key, the customer\_id will function as a foreign key in the customer table . The relationship is then a one- to one relationship. This maintains the relational integrity of the two tables since a customer id is unique to an individual customer .

The rest of the columns in the table will be the columns that were included in the original services.csv file that I imported into the project. These columns have a not null constraint.

**B1. Creating Table for CSV Data**

**---**

**Create Table public.services\_csv**

**(customer\_id text NOT null,**

**“InternetService” text NOT null,**

**“Phone” text NOT null,**

**“Multiple” text NOT null,**

**“OnlineSecurity” text NOT null,**

**“OnlineBackup” text NOT null,**

**“DeviceProtection” text NOT null,**

**“TechSupport” text NOT null,**

**PRIMARY KEY (Customer\_id),**

**CONSTRAINT customer\_id\_fkey FOREIGN KEY (customer\_id)**

**REFERENCES public.customer (customer\_id) MATCH SIMPLE**

**ON UPDATE NO ACTION**

**ON DELETE NO ACTION**

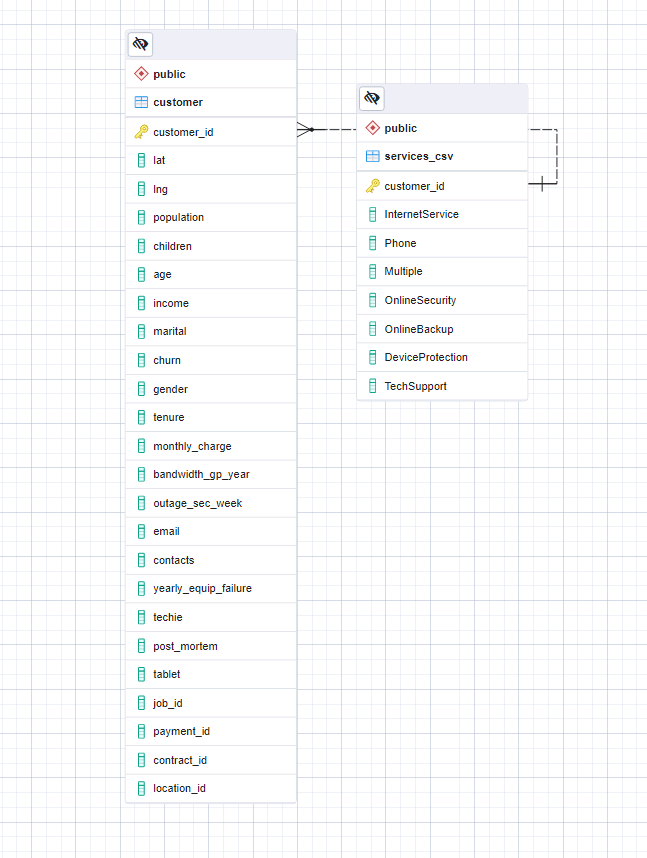
**NOT VALID**

**);**

**ALTER TABLE public.services\_csv**

**OWNER to postgres;**

Above is the code used to create the servicesa\_csv table. Using the gui tool to create the table, I then had the program create sql code to create the services\_csv table for me. Inside I define all the columns in the services file and give them NOT NULL constraints.



Additionally here is my ERD Diagram.

**B2: Loading CSV Data**

--

COPY services\_csv

FROM ‘CC:\LabFiles\services.csv’

Delimiter ‘,’

CSV HEADER;

To load the CSV data I imported the data using the PostGres Import functionality. However above is code that would load the services\_csv file manually.

**C.SQL Query**

--

(SELECT \*

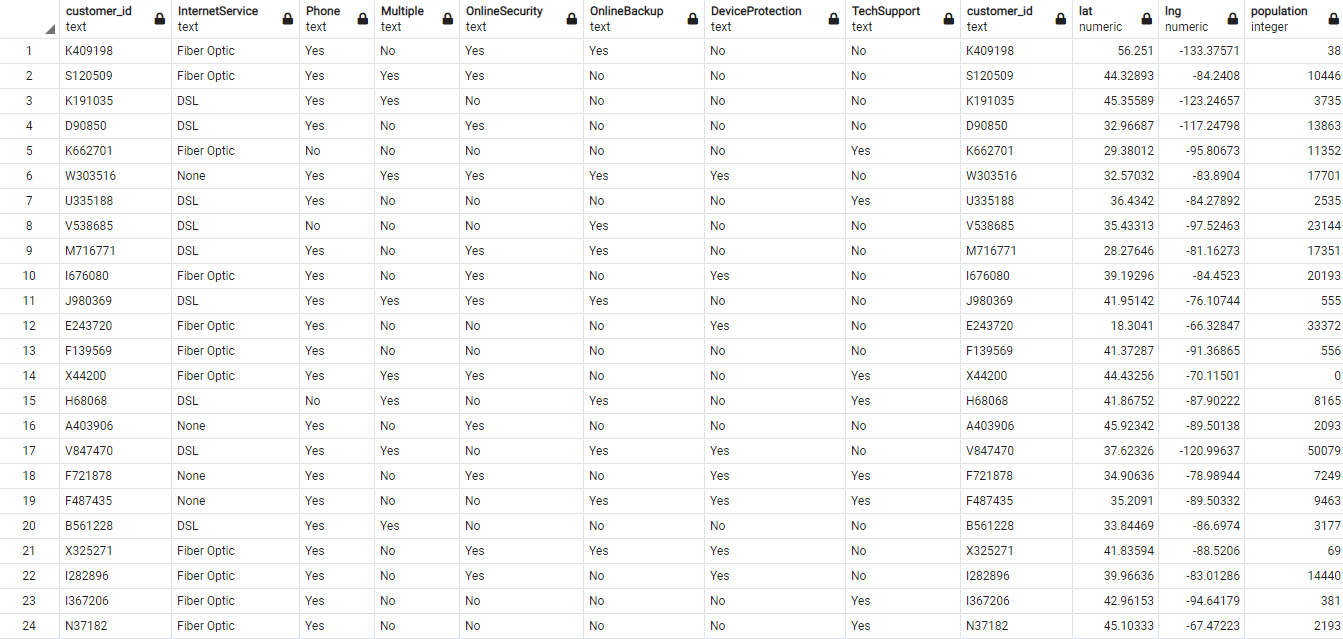
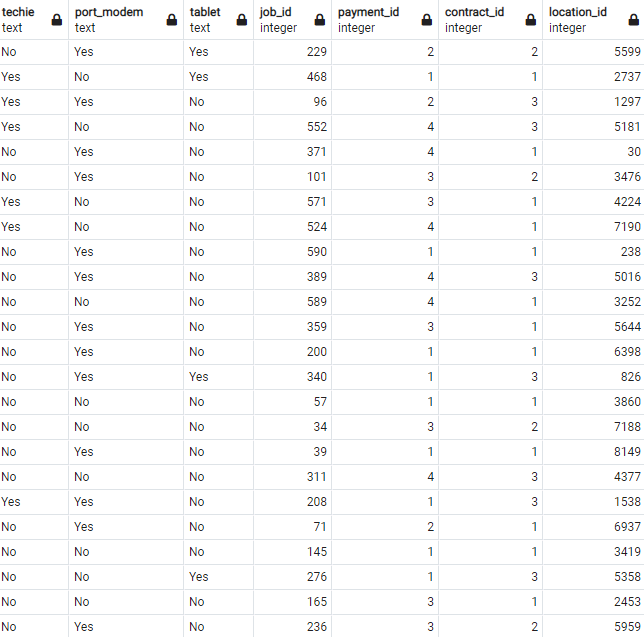
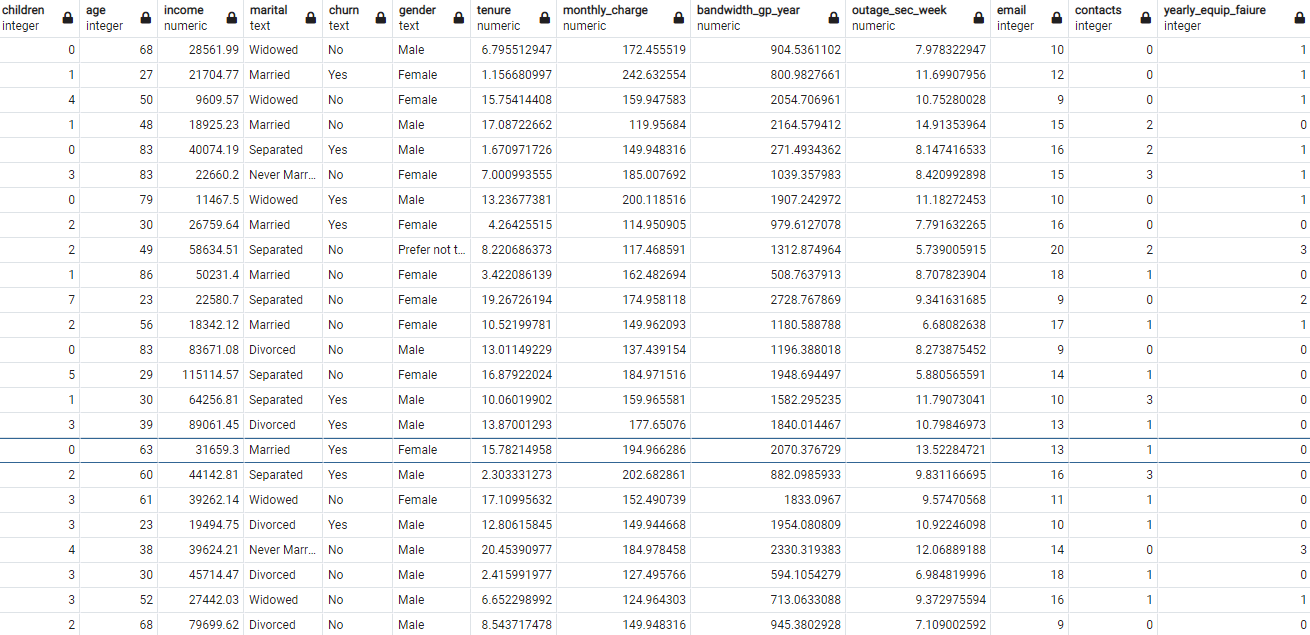
FROM services\_csv

Full JOIN customer

ON services\_csv.customer\_id = customer.customer\_id)

The above code fully joins the customer table and the services\_csv table. This puts all the columns together in a new table that we can view to examine the age of customers that purchased the phone services plan. Below are the results generated from merging the two tables on the customer\_id field. With this new table, you can use the customer id column to identify a customer, then proceed to examine their age and if they are using phone service in order to draw conclusions based on customer’s age and if they are purchasing the service.

**C1: CSV File(s)**

****

Above are the results from running the query to join the customer and services\_csv table.

The file is included in the submission.

**D: Add-On File Refreshment/Update Interval**

The main focus of my research question is for identifying trends of customers which a marketing team could use different strategies to influence certain customers to purchase the phone service.

To keep the information up to date, the services\_csv table should be updated daily as customers can purchase services fairly easily. It is fair to note that the customer's age should only change once a year and the customer\_id should never change, so the daily update is just with the phone service in mind. This timing will keep the data relevant and correct

**E: SQL Script**

**DELETE FROM services\_csv**

**COPY services\_csv**

**FROM ‘C:\LabFiles\services.csv’**

**Delimiter ‘,’**

**CSV HEADER;**

The process to refresh the add-on table is simple and straightforward. First the services\_csv table and its contents are deleted from the database. Then the services\_csv file is reloaded into the database with the code used earlier. If this services file is up to date and updated daily as I described in the previous section, then deleting the file and reloading it will include any updates or changes made.This process would be repeated daily to keep accurate results for any inquiries based off the customer and services\_csv tables.

**F: Panoptoo Video**

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3f22d127-90db-481a-bed2-b162010be5a7

**G: Sources**

*SQL full outer join*. Programiz. (n.d.). https://www.programiz.com/sql/full-outer-join

w3 schools , w3 schools. (2018). *SQL FULL OUTER JOIN*. SQL full outer join keyword. https://www.w3schools.com/sql/sql\_join\_full.asp