**Using SQL to Answer a Simple Question: How Many Customers Still Use DSL?**

Michelle Nelson

Department of Information Technology, Western Governor’s University

D205: Data Acquisition

Dr. David Gagner

June 30, 2023

**A. Research Question**

For this project, I selected the question “How many customers in each pre-determined age group still use DSL for their internet service?” This question is business-relevant because DSL is becoming quickly outdated, and it would be helpful to get a better understanding of how many customers may need to be persuaded into subscribing to newer services. Filtering this question into age groups can help tailor targeted marketing campaigns.

**A1. Identifying Data**

This question requires several columns. From the original dataset in the churn database, I will need to use the customer\_id and age columns. The customer\_id column is also present in the add-on CSV file titled “services.csv,” which I will use to join the CSV data to the customer table in the original dataset. Once the join is complete, I will also need the internet\_service column from the add-on CSV file to answer the research question. In addition, this question requires that I pre-define age groups. For these, I will use age groups commonly found in both online and paper surveys, starting from a minimum age I will determine by querying the dataset.

**B. Logical Data Model**

I will be using the services.csv file for this research question. I will import the entire file into a table which I will name “services\_csv.” The header names in the CSV file will be used for the column names in the new table. For each column, I will use the text datatype. The customer\_id column will eventually serve as a foreign key, as noted below. Since the customer\_id column of the customer table is also stored as text, choosing text for this column will allow this constraint. Since the negative answers in this file are always recorded as “No” or “None” instead of null, I will also add NOT NULL constraints to every column.

In the new table, the customer\_id column will serve as both the primary key for the table and also as a foreign key referencing the customer\_id column in the customer table in the original dataset. The customer\_id column makes a good primary key for the new table since two customers cannot share a customer ID, and any updates to the services a customer receives should be reflected by altering an existing record for that customer, not via the creation of an entirely new record. By making the customer\_id column a foreign key, we also maintain relational integrity between the customer table and the new table: a customer should not and cannot be added to the services\_csv table without already being present in the customer table.

Below is the ERD generated by pgAdmin using the “Generate ERD” functionality. This diagram illustrates the relationships described above for the new table, as well as the original customer table that will be involved in my SQL query used to answer the research question. Please note that while the diagram below shows a one-to-many relationship between the customer table and the services\_csv table, it is in fact one-to-one. PgAdmin does not currently support showing a one-to-one relationship in ERD diagrams, only one-to-many and many-to-many (PgAdmin Development Team, 2023).

A screenshot of a computer

Description automatically generated with medium confidence

**B1. Code for the Physical Data Model**

A screen shot of a computer code

Description automatically generated with low confidence

The above code generates the new table services\_csv and assigns the constraints described in section B.

**B2. Loading CSV Data**

A picture containing text, font, screenshot, white

Description automatically generated

The code above loads the services.csv data into the newly created services\_csv table.

**C. SQL Query**

Before writing the query that will answer the research question, I first need to know the minimum and maximum ages of all customers in the customer table. While I could skip this step and simply create age groups ranging from 0 to an unbounded maximum, I felt it would be better to limit the number of age groups so as not to have an unwieldy CASE statement in my query. In addition, completing this step will ensure there are no unneeded rows in the results once the query is run. The short query below was used to find this information. In section C1, I have pasted the results of this query and all queries to follow in section C.

A picture containing text, font, screenshot, white

Description automatically generated

This query returns a minimum age of 18 and a maximum age of 89. Thus, I decided to start the age groups at 18, rather than zero. The age groups can be configured in any way that proves useful for analysis, but I chose to use roughly 10-year increments with the exception of the first age group and the last. Since the first age group begins with 18, this group will be 18-24 or a 6-year bracket since I cannot continue the pattern and use 15-24. This bracket is followed by 10-year brackets: 25-34, 35-44, and so on. For the last age bracket, I chose to use an ELSE clause and create a group for ages 75 or older, but this, again, was simply a choice I made. The pattern could be continued if it made sense for the data.

The query below creates these age brackets using a CASE statement aliased as age\_group and informs the research question by calculating a COUNT of all rows that, after a left join between the left table customer and the right table services\_csv on customer\_id, have DSL as their internet service as listed in the column internet\_service. The query groups by age\_group so that it returns a count of customers in each age group. To make this easier to read, I also ordered the results by age\_group so the groups would list in numerical order.

A screenshot of a computer program

Description automatically generated with low confidence

**C1. CSV File**

Below are the results of the query requesting the minimum and maximum age of all customers in the customer table.

A screenshot of a computer

Description automatically generated with low confidence

The next image shows the table that is displayed after running my main query, which will answer the research question.

A screenshot of a computer

Description automatically generated with low confidence

A CSV file with the main query results shown above is also submitted alongside this essay.

**D. Add-On File**

The information contained in the add-on file, services.csv, appears to be highly transactional. A customer could call in at any time to cancel, change, or add new services. Thus, this file could be updated daily, even if marketing campaigns are only run monthly or quarterly. It is valuable that this information is kept up to date so as to avoid frustrating customers who may have already changed their service from DSL to a more modern offering but continue to receive outdated marketing emails.

However, if there is an issue with updating that frequently, then the add-on file must, at minimum, be updated as often as there are marketing campaigns. For example, if marketing campaigns are quarterly, then the add-on file should be updated quarterly, but prior to the campaign run date.

**E. SQL Script**

A picture containing text, screenshot, font, number

Description automatically generated

In order to refresh the data in the services\_csv table, I must first clear the table of records. The first line of code under the comment does just this. If this step is skipped, the code starting on line 4 will try to add new records to the bottom of the table still containing the old rows. This will result in an error since many of the rows the code will attempt to insert are only updates to previously existing rows. Attempting to add a new row with the same customer\_id as an existing row will violate the primary key constraint on customer\_id.

After the table is cleared, the same SQL commands used in section B2 are used to load the refreshed services.csv file, so long as the file name and location have not changed. Should the file name or location change, the new path can be pasted in single quotes on line 5 of the above code.

**F. Panopto Video**

The Panopto video can be found in the appropriate D205 folder on Panopto.com, but here is a link to it as well: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=35b51bce-cd53-4283-9a66-b03200197c7e>.

**G. Web Sources**

PgAdmin Development Team. (2023, June 29). *ERD Tool*. ERD Tool - pgAdmin 4 7.4 documentation. <https://www.pgadmin.org/docs/pgadmin4/development/erd_tool.html>

**H. Sources**

None were used.