A: Research Question

My research question for this project is, “Are customers with children (or more children) more likely to purchase higher speed internet packages?” This question is relevant to the business because it seeks to identify a trend among customers which could be useful in a variety of contexts. Promotional offers could be better directed to gain new customers using this data, and existing customers could be candidates for marketing of higher speed internet packages. This query could also be expanded to include other add-on services (online security, online backup, etc.) that might have similar value to the business.

A1: Identifying Data

This research question requires three separate columns from the available data, one of which is present in both the original data set and the add-on CSV file. From the customer table in the original ‘churn’ database, this question requires the “customer_id” and “children” fields. The customer_id field will be used to join this data to other data, while the children field is one of the two variables that we’re interested in researching. From the Services.csv add-on file, this question requires the “customer_id” field (for joining) and the “InternetService” field. By joining on customer_id, we can start to look at what relationship (if any) exists between number of children a customer has and the internet service package that they’ve chosen.

B: Logical Data Model

The add-on CSV file that I’ll be using for this research question is the services.csv file. While I only need two columns from this file, the entire file will be imported into a table, which I’ll name services_csv. That table will use the customer_id field as a primary key, as there cannot be two customers with the same customer id, and any time a customer would update their services, this should be reflected as an update to their existing record of services, rather than an entirely new record.

This customer_id field will also be a foreign key of the existing customer table, maintaining relational integrity between the two – a customer cannot be added to the services table without already having their information (address, phone number, etc.) recorded in the customer table. As customer_id is already stored as "text" datatype in other tables, the same datatype will be used here as well.

The other columns for this new table will simply reflect the existing column headers in the CSV, without changes or optimizations, as those are outside the scope of this project. Each column will have a NOT NULL constraint, as every column represents a service which the customer may choose not to purchase, and decisions not to purchase are recorded in the CSV file as being a "No" or "None", rather than a null.

B1: Creating Table for CSV Data

```
-- CREATION OF ADD-ON TABLE (SERVICES_CSV)

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;
```

The script above was generated from pgAdmin. I used the GUI tool to create the columns, select datatypes, etc. and then copied the SQL code that this process generated and executed. This code worked successfully to generate the table and allow me to insert all of the

data from the CSV file. There are optimizations available here, such as replacing the “text” datatype in most columns with other more efficient types for the data they contain. However, the text datatype was used to maintain consistency with the rest of the database, and such optimizations were outside the scope of this assignment.

B2: Loading CSV Data

```
-- ADDING THE DATA FROM THE SERVICES.CSV FILE TO THE SERVICES_CSV TABLE

COPY services_csv
FROM 'C:\LabFiles\Services.csv'
DELIMITER ','
CSV HEADER;
```

The above SQL code was used to add the data to the services_csv table from the services.csv file. I had originally attempted to drag and drop the file, moving it to the desktop, but this caused issues with regard to pgAdmin having permissions from the Desktop, rather than from this folder.

C: SQL Query

```
-- MY QUERY RELATING TO THE RESEARCH QUESTION

WITH cust_sums AS (
        -- Gather sum of customers for each number of children
        SELECT COUNT(customer_id) AS total,
                children
        FROM customer
        GROUP BY children
        ORDER BY children
),
cust_nums AS (
        -- Gather raw number of customers per # of children, for each service type
        SELECT cus.children AS num_children,
                serv."InternetService" AS service_type,
                COUNT(cus.customer_id) AS num_customers
        FROM customer AS cus
        INNER JOIN services_csv AS serv
        ON cus.customer_id = serv.customer_id
        GROUP BY cus.children, serv."InternetService"
        ORDER BY cus.children, serv."InternetService"
)

-- Main query takes data from CTE's and calculates percentage by service type amongst each # of children
SELECT cust_nums.num_children,
        cust_nums.service_type,
        cust_nums.num_customers,
        cust_sums.total,
        -- Cast both num_customers and total as numeric from string, divide and multiply by 100 for percentage
        -- Then round to 3 decimal places. Provides percentage of customers with each service type, for each # children
        ROUND((CAST(cust_nums.num_customers AS DECIMAL) / CAST(cust_sums.total AS DECIMAL)) * 100, 3) AS pct_service_type
FROM cust_nums
INNER JOIN cust_sums
-- The two CTE tables are both summaries, to join by # of children
ON cust_nums.num_children = cust_sums.children
```

The SQL query is much more involved than was required for the assignment. An initial SQL query that simply counted numbers of customers per number of children per internet service type was initially performed (visible in the cust_nums table in the above query). This query “informed” the research question, in that it provided numbers from which to begin working with, but I felt the raw numbers were inadequate and that a percentage breakdown of service type per number of children was more useful. This is because the raw numbers may not be comparable across different numbers of children (many more customers have 2 children than have 8 children), but those percentages of the whole would be more easily comparable.

For example, there are 2570 customers with 0 children. Of these 2570, 526 have no internet service, 922 have DSL, and 1122 have fiber optic service. Those numbers may not easily compare with the customers with 4 children, so the total number of customers at that number of children is also provided and used to break these down into percentages. Carrying on with the above example, of the 2570 customers with 0 children, 20.467% have no internet service, 35.875% have DSL, and 43.658% have fiber optic service. This provides much more context to the data for the purposes of making decisions about promotional or marketing efforts.

C1: CSV File(s)

Data Output | Explain | Messages | Notifications

	num_children integer	service_type text	num_customers bigint	total bigint	pct_service_type numeric
1	0	DSL	922	2570	35.875
2	0	Fiber Optic	1122	2570	43.658
3	0	None	526	2570	20.467
4	1	DSL	832	2472	33.657
5	1	Fiber Optic	1114	2472	45.065
6	1	None	526	2472	21.278
7	2	DSL	510	1495	34.114
8	2	Fiber Optic	663	1495	44.348
9	2	None	322	1495	21.538
10	3	DSL	507	1472	34.443
11	3	Fiber Optic	644	1472	43.750
12	3	None	321	1472	21.807
13	4	DSL	355	1006	35.288
14	4	Fiber Optic	425	1006	42.247
15	4	None	226	1006	22.465
16	5	DSL	61	212	28.774
17	5	Fiber Optic	105	212	49.528
18	5	None	46	212	21.698
19	6	DSL	67	187	35.829
20	6	Fiber Optic	80	187	42.781
21	6	None	40	187	21.390
22	7	DSL	68	185	36.757
23	7	Fiber Optic	84	185	45.405
24	7	None	33	185	17.838
25	8	DSL	77	210	36.667
26	8	Fiber Optic	85	210	40.476
27	8	None	48	210	22.857
28	9	DSL	33	92	35.870
29	9	Fiber Optic	36	92	39.130
30	9	None	23	92	25.000
31	10	DSL	31	99	31.313
32	10	Fiber Optic	50	99	50.505
33	10	None	18	99	18.182

A picture of the query's results is provided here, and the results of the query are also saved in the CSV file submitted alongside this report.

D: Add-On File Refreshment/Update Interval

The primary use case for this research question, at least to my thinking, is for marketing efforts for new and existing customers. In this regard, new customer marketing campaigns require only generalized information about trends rather than detailed and specific data about particular customers, while we would not want to harass existing customers with our marketing efforts. Additionally, the data isn't likely to change frequently, as customers don't change their services often, nor call up and inform their telecom provider when they've had (or lost) a child.

The services table will get updated daily with changes to specific customers' products, and that data should exist within the database rather than in separate flat files, for a variety of reasons. However, this query and the research question that it pertains to only needs to be refreshed on an occasional basis. A monthly refresh of this query's results seems adequate to me, as this would provide the trends for new customer marketing. This timing could also highlight existing customers who might consider new services, especially if this data were combined with information surrounding service contracts to allow for contacting a customer and setting up a new service contract in advance of the prior one's expiration.

E: SQL Script

```
-- REFRESH DATA ON ADD-ON TABLE

-- Clear all rows from the table
DELETE FROM services_csv;
-- Regenerate all rows from CSV flatfile
COPY services_csv
FROM 'C:\LabFiles\Services.csv'
DELIMITER ','
CSV HEADER;
```

The data from the CSV file was already loaded in section B2. This same code will work to refresh the services_csv table if the contents of the services.csv file have changed, though the existing rows in the table will need to be removed to allow this code to work. I would recommend moving away from storing all data pertaining to customer services in a flat file for a number of reasons, and there are optimizations that could be implemented to make this

update process more efficient. Both issues are outside the scope of this assignment, however, and this script will work within the existing business framework to update the table from the services.csv file as needed. The file path can be modified to reflect any location where the services.csv file may be stored, such as a network shared drive or other folder.

F: Panopto Video

The Panopto video for this project can be found in the D205 Student Creators folder. I would include a link, but I'm told that sessions in assignment folders cannot be shared.

G: Sources

No sources, web or otherwise, were used in the production of this project and its report.