Adetutu B.

Data Acquisition SQL Task

A: Research Question

Are customers who experience more than 10 seconds of power outages a week more likely to experience yearly equipment failure within a year, and how does it correlate with how often they replace their devices?

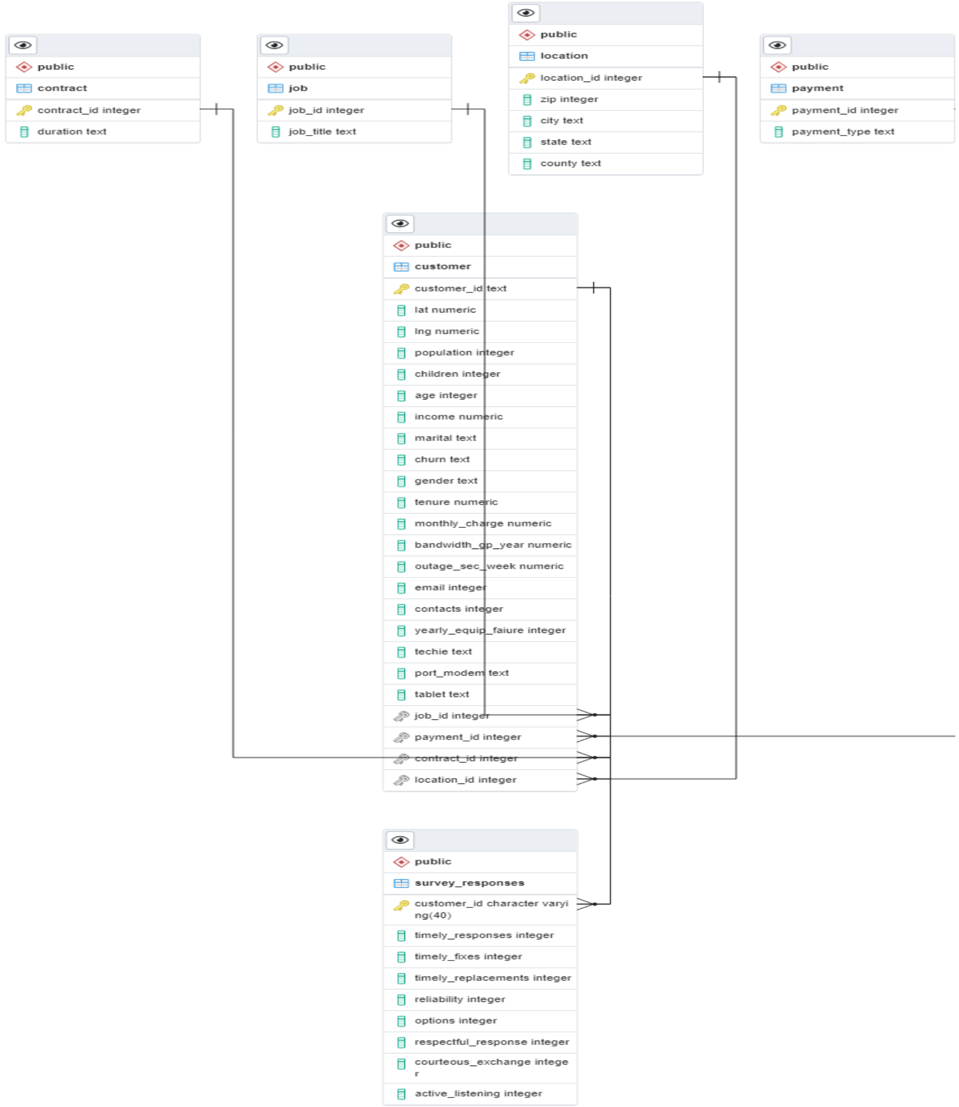
A1: Justifying the Question

Power outages have the ability to strain and damage devices which can cause device owners to lose important information and time trying to restore their data from a fixed or replaced device. My goal for this query is to gain insight on the possible relationship between outages longer than 10 seconds and yearly equipment failure for customers– and how it correlates with the frequency of device replacements. I will attempt to answer by only looking at customers who experience weekly outages for 10+ seconds and compare it to whether they replaced at least 1 device within a year. Data containing how many times the customers of interest replaced their devices will also be included to check for a correlation between needing to replace devices and experiencing certain lengths of outages and annual equipment failure. Businesses can use these insights to create bundled device protection and online backup service deals for customers who have dealt with outages and equipment failure, or may deal with it in the future. If more information was available, figuring out possible seasons where this issue may be more prevalent to consumers could enable businesses to offer seasonal service deals to prevent– and appease– affected customers.

A2: Data Identification

I will need 4 columns to answer my research question. From the customer table in the original churn database, I will be utilizing the “outage_sec_week” and “yearly_equip_faiure” columns. I will create an add-on table using the “Survey_Responses.csv” file and need the “timely_replacements” column to answer my question. The “customer_id” from both the original database and the add-on CSV will be joined together to help identify each unique customer. While the unique identifier and all the needed variables are originally integer data types, I chose to alter the “yearly_equip_faiure” column to present as a VARCHAR data type to show that all zeros represented “No” (the customer did not experience equipment failure) and “Yes” (the customer did experience equipment failure).

B: Database ERD



public
contract
contract_id integer
duration text

public
job
job_id integer
job_title text

public
location
location_id integer
zip integer
city text
state text
county text

public
payment
payment_id integer
payment_type text

public
customer
customer_id text
lat numeric
lng numeric
population integer
children integer
age integer
income numeric
marital text
churn text
gender text
tenure numeric
monthly_charge numeric
bandwidth_gp_year numeric
outage_sec_week numeric
email integer
contacts integer
yearly_equip_faiure integer
techie text
port_modem text
tablet text
job_id integer
payment_id integer
contract_id integer
location_id integer

public
survey_responses
customer_id character varying(40)
timely_responses integer
timely_fixes integer
timely_replacements integer
reliability integer
options integer
respectful_response integer
courteous_exchange integer
active_listening integer

B1: Describing the Relationship

In order to include the survey responses from the add-on CSV I created a table for it to be included with the other tables in the original database. For my query I used columns from the customer table and survey_responses table. I made the customer_id column a primary key so each individual customer could be uniquely identified. However, I also made customer_id a foreign key so the relationship between the customer table and survey_responses were linked and being added to one table required addition of the necessary information in the other table as well.

B2: Add-on CSV Table

```
--Create Add-On Table for survey data
CREATE TABLE public.Survey_Responses
(
    customer_id TEXT NOT NULL,
    Timely_Responses INT NOT NULL,
    Timely_Fixes INT NOT NULL,
    Timely_Replacements INT NOT NULL,
    Reliability INT NOT NULL,
    Options INT NOT NULL,
    Respectful_Responses INT NOT NULL,
    Courteous_Exchange INT NOT NULL,
    Active_Listening INT NOT NULL,
    PRIMARY KEY (customer_id),
    CONSTRAINT fkey_id
        FOREIGN KEY (customer_id)
            REFERENCES public.customer (customer_id)
);

ALTER TABLE public.Survey_Responses
    OWNER TO postgres;
```

B3: SQL Loads CSV Data

```
--Loading survey.csv data
Copy Survey_Responses
FROM 'C:\LabFiles\Survey_Responses.csv'
Delimiter ','
CSV HEADER;
```

C: SQL Query Statement

```
--SQL Query
ALTER TABLE public.customer
ADD yrly_equip_failure VARCHAR;

UPDATE public.customer
SET yrly_equip_failure =
(CASE
        WHEN yearly_equip_faiure >='1' THEN 'Yes'
        ELSE 'No'
END
);

SELECT cu.customer_id, cu.outage_sec_week, cu.yrly_equip_failure, su.timely_replacements
FROM customer cu
LEFT JOIN public.Survey_Responses su on cu.customer_id=su.customer_id
WHERE cu.outage_sec_week >=10
ORDER BY cu.outage_sec_week;
```

C1: Query Results

My query results have been included as a separate CSV file.

D: Add-on File Refresh Period

For the particular data used for this research question it would be best to refresh the add-on file on a yearly basis to keep a well updated query.

D1: Refresh Period Explanation

My reasoning for choosing a yearly refresh period for the add-on file is because the only data I ultimately use in my query from the ‘survey_responses’ table is the ‘timely_responses’ column. While I do use the weekly outage data from the original customer table, it is unlikely for a customer to want to change their devices more than a few times a year. So, weekly/monthly refreshes are too frequent, and quarterly updates likely won’t paint as clear of a picture of the query results as a yearly refresh would since the equipment failure of customers is only taken into account on a yearly basis.

```
--Clear and Restore Add-On Table Data
DROP TABLE IF EXISTS Survey_Responses
CASCADE;

Copy Survey_Responses
FROM 'C:\LabFiles\Survey_Responses.csv'
Delimiter ','
CSV HEADER;
```

F: Data Sources

PostgreSQL Tutorial. (2024, January 22). PostgreSQL UPDATE. *https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-update/*

W3Schools. (n.d.). SQL CASE Expression. https://www.w3schools.com/sql/sql_case.asp