CSI 4142 - Fundamentals of Data Science

Project Phase 2: Data Staging and Data Mart Creation

Due Date: March 22, 2024, 11:59pm

Group 33: Tanner Guiddings, Coralie Ostertag, and Dana Shayakhmetova

I. Introduction

Project Overview:

In Phase 0, we found a dataset containing customer information and their responses to various advertisements. Then, in Phase 1, we did the conceptual design of our dimensional model. Now, in Phase 2, our goal is to dive deeper into this data, ensuring its quality and transforming it into a format that allows for effective analysis (for following phases).

Objectives:

1. To gain deeper insights into the influence of demographic factors on ad responses.
2. To employ classification algorithms, such as logistic regression, to predict ad effectiveness.
3. To establish a well-structured data mart for efficient data storage and retrieval.

Phase 2 Focus:

During this phase, we'll focus on the essential steps of data staging: extraction, transformation, and loading (ETL). We'll implement techniques such as data cleaning, normalization, and feature engineering to prepare the data for analysis. Furthermore, we'll utilize a Database Management System (DBMS) to construct a reliable data mart for streamlined access and analysis.

II. High-Level Data Staging Plan

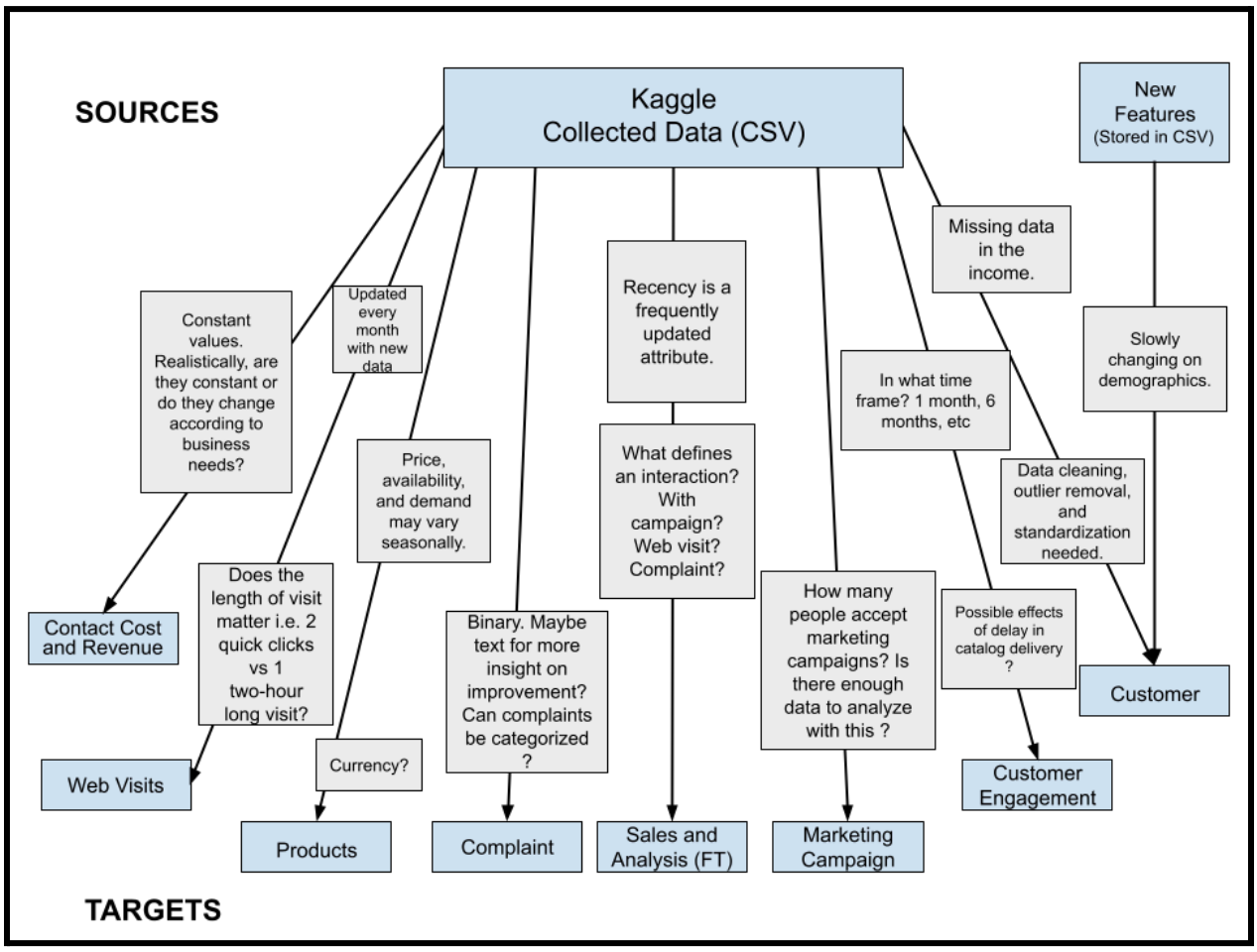

Figure 1: High Level Data Staging Schema

The high-level schematic presents a comprehensive overview of our data staging process, highlighting key steps and potential challenges that require extra attention during ETL. While our primary data source is the CSV file from Kaggle, we've incorporated additional features through feature engineering to enrich our dataset. The schematic also outlines various transformation techniques, including data cleaning, normalization, and questions related to them, underscoring their crucial role in refining raw data into a structured and analytically useful format for our project.

III. Implementation Details

Data Extraction:

- *What?* Finding a way to get our data to be able to use it in the following steps.
- *How?* We uploaded our data to github and used `pd.read_csv` in order to be able to use it in our colab environment.

Data Transformation:

- *What?* During this stage, we cleaned, processed, and transformed the raw data into a format suitable for analysis. This could involve handling missing values, removing duplicates, converting data types, aggregating or summarizing data, and performing any necessary feature engineering.
- *How?* We used pandas DataFrame methods and functions to perform data manipulation tasks. For example, we might use `data.duplicated()` to check for duplicate rows or `data.isnull()` and `data.fillna()` to check for and to handle missing values.

Data Loading:

- *What?* In data loading, we separate our transformed data into data frames and generate keys that represent each dimension and fact table of our project. Then, we load all of this onto our DBMS.
- *How?* We used the pandas and tqdm python libraries to generate primary keys and separate our data into dimension and fact table data frames. Then, we create dimension and fact tables with PostgreSQL on the supabase project website. Afterwards, in our colab, we create a supabase instance and connect to our DBMS with an API key and project link. To load the data into the tables, we transform each data frame into json format and upsert it into our DBMS.

Documentation: See the Colab comments for details on each step we completed. The link to our Google Colab has been placed in the Appendix Part B of this document.

IV. Data Quality Issues and Solutions

Identification of data quality issues encountered during the ETL process, such as:

- Missing values:
  - *How?* Found by checking all rows for missing values.
  - *Issue:* The income column has 24 missing values.
  - *Solution:* Replacing them with the median income.
- Typos or Outliers:
  - *How?* Using describe, we look at the ranges of all the values.
  - *Issue:* The age column seems to have some issues. We can see that the max value is 121, which is very very likely wrong.
  - *Solution:* Remove the entries over 95 years old (as the customers above are likely not the ones actually shopping and this won't be relevant to what we are looking at).
- Duplicates:

- *How?* Using data.describe()
- Checked to see if any rows were duplicated, but that is not the case here.

V. Data Mart Creation

After our data was transformed, we generated the corresponding surrogate keys for each table, according to the numeric system we have established in phase one. As a reminder, our starting surrogate keys for each dimension table is of the form X00000, which are values ranging from 100000 to 700000.

Consequently, we organized our data into separate data frames, each representing a dimension or fact table within our project. We made sure that the newly created data features and the surrogate keys are included in this step.

Next, to create our Data Mart, we used Supabase because of its easy-to-use features and powerful capabilities that align closely with our project requirements. In particular, this DBMS is an extension of PostgreSQL that integrates well with Python and offers a RESTful API that allows us to interact with our database directly from our google colab.

One of the convenient features of Supabase is its SQL Editor, which enables us to create our dimension and fact tables according to the schema outlined in phase 1. For the specific SQL commands used, please consult Appendix A at the end of this document.

It's important to mention that we opted not to auto-increment the primary keys for our dimension tables using default values. This decision was made because our data frames were initially generated with pre-assigned primary keys.

However, in the event that these primary keys were absent, we would have created a reusable sequence, denoted as primary_key_sequence, to handle the generation of surrogate keys. This sequence starts from 1 and increments by 1, as seen below:

```
-- Define primary key sequence that we will reuse for each dimension
CREATE SEQUENCE primary_key_sequence START WITH 1 INCREMENT BY 1;


-- Create and example table with the above sequence
CREATE TABLE Example Dimension (
   Primary_Key INTEGER DEFAULT (X000000 + nextval('primary_key_sequence'))
PRIMARY KEY,
   … other attributes
);
```

By doing the above, we successfully built our dimensions and fact tables within Supabase using PostgreSQL. These tables were designed in alignment with the attributes and relationships defined during Phase 1. To provide a clear overview of these connections and table characteristics, we've developed the following schema:

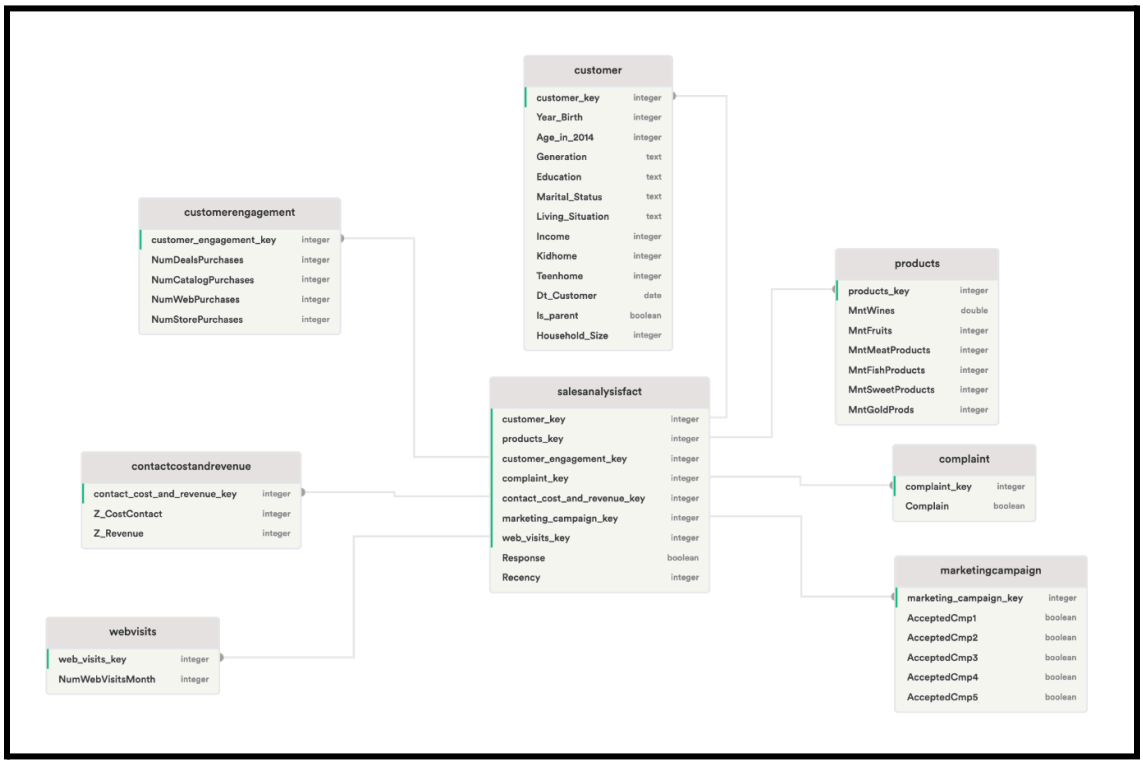

Figure 2: Schema Diagram of Database Table Relationships and Attributes

In order to load our data into DBMS, we needed to establish API connections between our Google Colab environment and Supabase. This involved utilizing the API authenticator key and project URL provided by Supabase. Once these connections were established, we proceeded to convert our data into JSON format. For a detailed demonstration of this process, please refer to the code execution in Google Colab.

Additionally, it's nice to note that despite our data frames containing integer values, they were represented in decimal form, such as numbers like 10 being displayed as 10.0. This discrepancy caused errors during the loading process because the attributes in SQL were specified as strictly integers. Thus, we resolved this issue by converting all decimal values to integers before proceeding with the JSON transformation, as explained in our code comments.

Moreover, instead of using insert, we utilized upsert in our data. As an example:

```
data, count =
supabase.table(table-name).upsert({json-data}).execute()
```

We decided to use upserts because they can insert new rows while avoiding duplicates, aligning perfectly with our project objectives. This approach ensures that our code can be executed multiple times without creating duplicate entries, thus maintaining data integrity and consistency within our database.

VI. Team Planning

Deliverable Checklist	Responsible Team Members	Completion Date
Data cleaning and transformation	Coralie	March 19, 2024
Transforming dataset into dimensions and linking to fact table	Tanner	March 20, 2024
Surrogate key generation	Tanner	March 20, 2024
Creating database	Dana	March 20, 2024
Linking Colab to Supabase	Dana	March 20, 2024
Document creation and organization	Coralie	March 15, 2024
Document writing	All team members	March 21, 2024
Team planning	Coralie	March 21, 2024

VII. Conclusion

In Phase 2 of our project, we successfully navigated through the extraction, transformation, and loading (ETL) operations of Data Staging. With many Python libraries and Supabase's RESTful API, we prepared the dataset for comprehensive analysis, ensuring its readiness for the next two phases.

Some of our key achievements include the implementation of data cleaning, normalization, and feature engineering techniques, which enhanced our data quality and suitability for analysis. Additionally, we generated surrogate keys for each dimension and fact table, according to the

dimension modelling diagram that we built in phase 1. Furthermore, we established a robust and well-structured data mart using Supabase as our chosen Database Management System (DBMS), enabling efficient storage and retrieval of data with PostgreSQL.

We also faced a couple of technical challenges, including data representation discrepancies and loading complexities which were overcome through collaborative efforts and Supabase documentation.

Looking ahead, there are several ways we can ameliorate our project for future phases: Firstly, we think implementing automated data quality checks and validation processes to detect and resolve issues more efficiently could be useful. Additionally, we can consider incorporating real-time data streaming and analytics capabilities that can empower dynamic updates in response to evolving customer behaviour.

VIII. Appendix

Appendix A: PostgreSQL

The following code blocks create the dimension tables in our Data Mart:

```
CREATE TABLE Complaint (
   Complaint_Key INTEGER PRIMARY KEY,
   Complain BOOLEAN
);
```

```
CREATE TABLE ContactCostAndRevenue (
   Contact_Cost_and_Revenue_Key INTEGER PRIMARY KEY DEFAULT 600000,
   Z_CostContact INTEGER DEFAULT 3,
   Z_Revenue INTEGER DEFAULT 11
);
```

```
CREATE TABLE CustomerEngagement (
   Customer_Engagement_Key INTEGER PRIMARY KEY,
   NumDealsPurchased INTEGER,
   NumCatalogPurchases INTEGER,
   NumWebPurchases INTEGER,
   NumStorePurchases INTEGER
);
```

```sql
CREATE TABLE Customer (
   Customer_Key INTEGER PRIMARY KEY,
   Year_Birth INTEGER,
   Age_in_2014 INTEGER,
   Generation TEXT,
   Education TEXT,
   Marital_Status TEXT,
   Living_Situation TEXT,
   Income INTEGER,
   Kidhome INTEGER,
   Teenhome INTEGER,
   Dt_Customer DATE,
   Is_parent BOOLEAN,
   Household_Size INTEGER
);
```

```sql
CREATE TABLE MarketingCampaign (
   Marketing_Campaign_Key INTEGER PRIMARY KEY,
   AcceptedCmp1 BOOLEAN,
   AcceptedCmp2 BOOLEAN,
   AcceptedCmp3 BOOLEAN,
   AcceptedCmp4 BOOLEAN,
   AcceptedCmp5 BOOLEAN
);
```

```sql
CREATE TABLE Products (
   Products_Key INTEGER PRIMARY KEY,
   MntWines FLOAT,
   MntFruits INTEGER,
   MntMeatProducts INTEGER,
   MntFishProducts INTEGER,
   MntSweetProducts INTEGER,
   MntGoldProducts INTEGER
);
```

```sql
CREATE TABLE WebVisits (
```

```
   Web_Visits_Key INTEGER PRIMARY KEY,
   NumWebVisitsMonth INTEGER
);
```

The following code block creates the Fact Table in our Data Mart:

```
CREATE TABLE SalesAnalysisFact (
   Customer_Key INTEGER REFERENCES Customer(Customer_Key),
   Products_Key INTEGER REFERENCES Products(Products_Key),
   Customer_Engagement_Key INTEGER REFERENCES
CustomerEngagement(Customer_Engagement_Key),
   Complaint_Key INTEGER REFERENCES Complaint(Complaint_Key),
   Contact_Cost_and_Revenue_Key INTEGER REFERENCES
ContactCostAndRevenue(Contact_Cost_and_Revenue_Key),
   Marketing_Campaign_Key INTEGER REFERENCES
MarketingCampaign(Marketing_Campaign_Key),
   Web_Visits_Key INTEGER REFERENCES WebVisits(Web_Visits_Key),
   Response BOOLEAN,
   Recency INTEGER,
   PRIMARY KEY (Customer_Key, Products_Key, Customer_Engagement_Key,
Marketing_Campaign_Key, Complaint_Key, Web_Visits_Key,
Contact_Cost_and_Revenue_Key)
);
```

Appendix B: Code

The code for the data transformation and cleaning can be found in this Colab link: https://colab.research.google.com/drive/1k-6hJCWFq4Scx8NG7BxmrIsXl6lov24D?usp=sharing