DIGITAL MARKETING ANALYTICS

GROUP ASSIGNMENT 1

In your groups, analyse the DMEF data sets to gain insights into the effectiveness of the different direct marketing channels (catalog mailing vs. email) and propose strategies the company could explore. This is an open-ended assignment so there is no one "right" answer, but you are expected to do substantial work. At the very least, you need to consider the following (for top marks, you need to consider more):

1. Which channel has better response rates, catalog mailing or email?
2. Segment the customers using the RFM dimensions and estimate the response rates for each RFM cell and make a decision on how many to mail based on an estimate of ROI of the campaign (assume it costs a normalized $1 per mailing and an average profit of $30 per purchase).
3. Calculate the Average CLV of a customer (Note that this is an e-commerce setting).
4. Management believes different types of customers have different CLVs. How would you would explore this belief so the results are useful for the business? (There is no need to implement it, but write a plan on how you would go about doing it, based on the data you have in this case.)

You can consult the internet and other open resources freely but all work has to be done within the group. You will have to make a lot of design decisions yourself in the analysis- always include a brief explanation of why you chose what you chose to do.

The grade depends on your effort, correctness of analysis, breadth, depth and clarity of presentation.

Submission: Submit a *technical report* of maximum 5 A4 pages (including figures and tables) using 11pt Arial or Calibri font in pdf. Do not submit anything else. (A technical report is a document that presents the result of a technical investigation in a way that an educated but non-expert can understand it.)

Suggested Structure of your report: (i) one paragraph Executive Summary or Abstract; (ii) Introduction (iii) Problem Description (iv) Methodology (v) Results & Discussion (vi) Conclusion. It's okay to vary the structure a bit from this but note this is standard.

Before you get started:

You need to create a database in Postgresql and load the four DMEF data files given to your database. You can find a detailed description of the data sets in the document "Data Set 9 DMEF MultiChannel Gift Dataset.pdf".

You may want to follow the following instructions:

1. Download and install the Postgres server.

- The postgres server should be downloaded from official website depending on your OS Mac or Windows. There are many online tutorials that can help you with the downloading and installation.
- Please note the default port for Postgres server is 5432.
- After installation, please configure the environment on your machine to enable you to run psql on your terminal directly.
    - Windows: see here.
    - Mac: Add following command line to .bash_profile under the home directory. This is an example of PostgreSQL version 15:

```
export PATH=/Library/PostgreSQL/15/bin:$PATH
```

2. Connect Posgtres Server by typing psql in the command line. You should see something like the below:

```
(base) dyn3205-139:~ lialiang$ psql
psql (15.1, server 15.2)
Type "help" for help.
lialiang=#
```

Note: you may need to specify the host, username, password, port, and database. See here.

3. Now, create database using the command CREATE DATABASE:

```
# HW1 is the name of the database, you could choose whatever you want.
lialiang=# CREATE DATABASE HW1;
CREATE DATABASE
```

4. Loading data – example using DMEFExtractOrdersV01.csv.

    a. Go to the correct database

```
lialiang=# \c hw1
psql (15.1, server 15.2)
You are now connected to database "hw1" as user "lialiang".
hw1=#
```

    b. Create tables

```
lialiang=# CREATE TABLE Orders
lialiang-# (Cust_ID bigint,
lialiang(# OrderNum bigint,
lialiang(# OrderDate date,
lialiang(# OrderMethod text,
lialiang(# PaymentType text);
CREATE TABLE
```

Note: using varchar(2) for OrderMethod and PaymentType could be a bad idea nowadays. Use text instead for flexibility.

c. COPY csv file into database (note: you need to change the path from the below)

```
lialiang=# COPY orders FROM '/Users/lialiang/Desktop/DMA assignment 1/Digital Marketing HW1 data set/DMEFExtractOrdersV01.csv' DELIMITER ',' CSV HEADER;
COPY 241366
```

5. Using Python to clean up the data – example using customers

a. Import libraries

```
import pandas as pd
from sqlalchemy import create_engine
```

b. Connect to the database server

```
# Create an engine to connect to the PostgreSQL database
engine = create_engine('postgresql://[username]:[password]@localhost:5432/[database]')
```

Note: replace [username], [password] and [database]

c. Load and clean data

```
customers = pd.read_csv(pathtofile + '/DMEFExtractSummaryV01.csv')

# Replace NaN values with NULL to prevent errors during the SQL insert df
# Some cells in this file contain spaces instead of being empty.
customers = customers.where(pd.notnull(customers), None)
customers = customers.replace(r'^\s*$', None, regex=True)
```

Note: pathtofile should be defined.

d. Load to psql

```
customers.to_sql('customers', engine, if_exists='replace', index=False)
```

Note: By using this method, the columns name could be case sensitive, meaning that "Cust_ID" and "cust_id" are treated as two different columns.

6. Sanity check- identifying the top 5 customer locations by average spend

- Location can be identified by SCF Code in Customers table
- LineDollars in Lines table represents the selling price of the line item in dollars.
- Therefore, those two tables need to be joined.
- Then apply aggregation on the location.
- Rank by the spend.

6.2. Using Python

```
# Execute a SQL query
with engine.connect() as connection:
        result = connection.execute(' SELECT "SCF_Code", SUM("LineDollars")/COUNT(DISTINCT
"OrderNum") AS "spend" \
        FROM Customers as c INNER JOIN Lines as
l \
        ON c."Cust_ID" = l."Cust_ID"\
        GROUP BY "SCF_Code"\
        ORDER BY "spend" DESC\
        LIMIT 5;')
```

```
for row in result:
        print(row)

#('823', 950.4299999999997)
#('202', 268.72499999999997)
#('36', 265.5342105263154)
#('788', 260.67499999999995)
#('586', 234.19999999999996)
```

6.3. Using Postgres

```
SELECT "SCF_Code", SUM("LineDollars")/COUNT(DISTINCT "OrderNum") AS "spend"
FROM Customers as c INNER JOIN Lines as l
ON c."Cust_ID" = l."Cust_ID"
GROUP BY "SCF_Code"
ORDER BY "spend" DESC
LIMIT 5;

hw1=# SELECT "SCF_Code", SUM("LineDollars")/COUNT(DISTINCT "OrderNum") AS
"spend"
hw1-# FROM Customers as c INNER JOIN Lines as l
hw1-# ON c."Cust_ID" = l."Cust_ID"
hw1-# GROUP BY "SCF_Code"
hw1-# ORDER BY "spend" DESC
hw1-# LIMIT 5;

823 | 950.4299999999997
202 | 268.72499999999997
36 | 265.5342105263154
788 | 260.67499999999995
586 | 234.19999999999996
```