

Digital Marketing Analytics Group Report

April 2025

Bryan Cheong
Carlos Lembono
Leo Koay
Louis Pedrix

MSc Business Analytics
Imperial College London

Question 1

Create database

We used pgAdmin 4 to create the database because the platform provides easier steps in creating database. We created a database named DMA_Group_Assignment_1 through command Create > Database, then set the encoding to standard UTF8.

Upload dataset

There 5 tables provided in the zip file and only 4 will be uploaded into the database because 1 table (Demo Codes Reference) only contains code references for several columns which is not relevant in this assignment case. We explored different kinds of method in uploading and cleaning the data via pgAdmin and Python. We included the command to upload the other 4 datasets and some cleaning methods if needed.

1. Contacts table

a. Create table

```
CREATE TABLE contacts(cust_id INT,
                        contact_date DATE,
                        contact_type TEXT);
```

b. Copy data

```
COPY contacts(cust_id, contact_date, contact_type)
FROM
'/Users/carloslembono/Documents/Postgraduate/Imperial
College/Academic/Lecture/05 Summer Term/Digital
Marketing/Group Assignment/Group Assignment 1/Digital
Marketing HW1 data set/DMEFExtractContactsV01.csv'
DELIMITER ','
CSV HEADER;
```

2. Lines table

a. Create table

```
CREATE TABLE lines(cust_id INT,
                    order_num BIGINT,
                    order_date DATE,
                    line_dollars FLOAT,
                    gift TEXT,
                    receipt_number INT);
```

b. Copy data

```
COPY lines(cust_id, order_num, order_date, line_dollars,
           gift, receipt_number)
FROM
'/Users/carloslembono/Documents/Postgraduate/Imperial
```

```
College/Academic/Lecture/05 Summer Term/Digital
Marketing/Group Assignment/Group Assignment 1/Digital
Marketing HW1 data set/DMEFExtractLinesV01.csv'
DELIMITER ','
CSV HEADER;
```

c. Cleaning data

There are whitespaces in the table, and we replaced it with NULL values.

```
UPDATE lines
SET
    gift = NULLIF(gift, ' '),
    receipt_number = NULLIF(receipt_number, ' ')
WHERE gift = ' ' OR receipt_number = ' ';
```

3. Order table

a. Create table

```
CREATE TABLE orders(cust_id INT,
                     order_num BIGINT,
                     order_date DATE,
                     order_method TEXT,
                     payment_type TEXT);
```

b. Copy data

```
COPY orders(cust_id, order_num, order_date, order_method,
payment_type)
FROM
    '/Users/carloslembono/Documents/Postgraduate/Imperial
College/Academic/Lecture/05 Summer Term/Digital
Marketing/Group Assignment/Group Assignment 1/Digital
Marketing HW1 data set/DMEFExtractOrdersV01.csv' DELIMITER
','
CSV HEADER;
```

4. Summary table

We used *sqlalchemy* in Python (***submitted in notebook file: UploadSummaryTable.ipynb***) to upload this table because it contains many columns (170 columns) and this tool can read and upload the data much more efficient and faster.

The ids data is set to be in integer data type because it consumed less memory except for order_num where we use big integer as data types since the digits are higher. There are many missing data for demography data such as:

- 85,167 (~85%) records with no occupation code data

- 34,710 records with no income code data
- 25,410 records with no dwelling code data
- ... and so on

We need to fill the missing value since it comprises most of the data (removing it will not make any sense). This missing value might be due to the personal information which people might not be comfortable sharing it. Similarly, the missing value is filled with whitespaces hence we replace it with NULL. Because some of the data are not relevant in CLV calculation (demography such as occupation, income, dwellings, etc.), we will leave the missing value as it is.

Additionally, we decided to use snake case in naming conventions as it is PostgreSQL automatically converts unquoted identifiers to lowercase, so using uppercase letters can lead to unexpected behavior and the need for quoting.

Combine dataset

We combine the lines and orders because lines table contain detail information about the order value hence it is sensible to join these tables.

```
CREATE TABLE order_detail AS
(SELECT o.cust_id,
       o.order_num,
       o.order_date,
       o.order_method,
       o.payment_type,
       l.line_dollars,
       l.gift,
       l.receipt_number
FROM orders AS o
LEFT JOIN lines AS l
ON o.cust_id = l.cust_id)
```

Indexing

We also created indexes to optimise a database such as in SQL improving performance to ease the query. There are 4 indexes we created

1. Create index for contacts

```
CREATE INDEX idx_cust_cont_date ON contacts(cust_id,
contact_date)
```

2. Create index for lines. Create index for the receipt number just in case we want to see the receipt with gift.

```
CREATE INDEX idx_recipient ON lines(receipt_number)
```

3. Create index for orders

```
CREATE INDEX idx_cust_order_date ON orders(cust_id,  
order_date)
```

4. Create index for customer

```
CREATE INDEX idx_cust_id ON summary(cust_id)
```

Query Question

Identify the top 5 customer locations by average spend. The average spend is calculated by the total spending divided by the total number of purchases per customer. Then we calculate this number over a period of time. here are 2 approaches we did to calculate the average spend per location. First, the data contains purchases from 2001-2007, divide into two period before 2004 and 2004 to 2007. Thus, this approach gives an overview about top 5 location before 2004 and between 2004 to 2007. The second is by calculating it by yearly average spend from 2001 to 2007 and then take mean of the average per year.

Approach 1

```
-- Create summary table of order and value  
WITH t1 AS  
(SELECT o.cust_id,  
        o.order_num,  
        o.order_date,  
        o.order_method,  
        s.scf_code,  
        l.gift,  
        l.line_dollars  
FROM orders AS o  
LEFT JOIN lines AS l  
ON o.cust_id = l.cust_id AND o.order_num = l.order_num  
LEFT JOIN summary AS s ON o.cust_id = s.cust_id ),  
  
-- Query data for period before 2004  
b_2004 AS  
(SELECT scf_code,  
        ROUND(SUM(line_dollars)::NUMERIC, 2) AS spending,  
        COUNT(DISTINCT cust_id) AS purchase_num  
FROM t1  
WHERE EXTRACT(YEAR FROM order_date) < 2004  
GROUP BY scf_code ),  
  
-- Query data for 2004 to 2007 period  
f_2004_to_2007 AS
```

```
(SELECT scf_code,
        ROUND(SUM(line_dollars)::NUMERIC, 2) AS spending,
        COUNT(DISTINCT cust_id) AS purchase_num
FROM t1
WHERE EXTRACT(YEAR FROM order_date) >= 2004
GROUP BY scf_code)
```

```
(SELECT scf_code,
        ROUND(spending / purchase_num::NUMERIC, 2) AS
spending_per_purchase,
        'before_2004' AS period
FROM b_2004
ORDER BY spending_per_purchase DESC LIMIT 5)
UNION ALL
(SELECT scf_code,
        ROUND(spending / purchase_num::NUMERIC, 2) AS
spending_per_purchase,
        '2004_to_2007' AS period
FROM f_2004_to_2007
ORDER BY spending_per_purchase DESC LIMIT 5;
```

Approach 2

```
-- Calculate the yearly spending
WITH yearly_spend AS
(SELECT s.scf_code,
        EXTRACT(YEAR FROM o.order_date) AS year,
        o.cust_id,
        SUM(l.line_dollars) AS total_spend
FROM orders o
LEFT JOIN lines l
ON o.cust_id = l.cust_id AND o.order_num = l.order_num
LEFT JOIN summary s
ON o.cust_id = s.cust_id
GROUP BY s.scf_code, year, o.cust_id),

avg_by_year AS
(SELECT scf_code,
        year,
        AVG(total_spend) AS avg_spend_per_customer
FROM yearly_spend
GROUP BY scf_code, year),

-- Pivot the data with year as columns
pivoted AS
(SELECT scf_code,
```

```

        ROUND(AVG(CASE WHEN year = 2001 THEN
avg_spend_per_customer END)::NUMERIC, 2) AS "2001",
        ROUND(AVG(CASE WHEN year = 2002 THEN
avg_spend_per_customer END)::NUMERIC, 2) AS "2002",
        ROUND(AVG(CASE WHEN year = 2003 THEN
avg_spend_per_customer END)::NUMERIC, 2) AS "2003",
        ROUND(AVG(CASE WHEN year = 2004 THEN
avg_spend_per_customer END)::NUMERIC, 2) AS "2004",
        ROUND(AVG(CASE WHEN year = 2005 THEN
avg_spend_per_customer END)::NUMERIC, 2) AS "2005",
        ROUND(AVG(CASE WHEN year = 2006 THEN
avg_spend_per_customer END)::NUMERIC, 2) AS "2006",
        ROUND(AVG(CASE WHEN year = 2007 THEN
avg_spend_per_customer END)::NUMERIC, 2) AS "2007"
FROM avg_by_year GROUP BY scf_code)

-- Calculate the average spending for 7 years (2001-2007)
SELECT *,
ROUND((COALESCE("2001", 0) + COALESCE("2002", 0) +
        COALESCE("2003", 0) + COALESCE("2004", 0) +
        COALESCE("2005", 0) + COALESCE("2006", 0) +
        COALESCE("2007", 0))/7.0, 2) AS avg_spend_2001_to_2007
FROM pivoted
ORDER BY avg_spend_2001_to_2007 DESC
LIMIT 5;

```

Both methods have its own strengths and weaknesses, for example, the first approach it is suitable to spot top performers in each period to identify strong spenders between each period. Additionally, this approach can be utilised to spot trends to see if certain areas might be arising which is potentially can be a new market to tap in. However, SCF codes in top 5 may be completely different in each period which makes it hard to compare changes over time. Areas such as 823 have the biggest average spend before 2004 but vanished after 2004. Table 1 shows the result of the query.

Table 1. Table output of Approach 1 query

Area (SCF Code)	Spending per Purchase	Period
823	2,308.25	Before 2004
81	1,014.59	Before 2004
97	753.23	Before 2004
814	662.20	Before 2004
499	644.34	Before 2004
36	949.18	2004 to 2007
511	796.21	2004 to 2007

Area (SCF Code)	Spending per Purchase	Period
734	605.10	2004 to 2007
383	537.29	2004 to 2007
811	528.31	2004 to 2007

The second approach, on the other hand, is suitable to track records (historical performance) for the same area (SCF Codes). In addition, by measuring the growth over time, the company can compare how customer behavior shifted across time. But SCF codes that were only relevant in one period (i.e., they had no activity in the other) may get excluded or undervalued. Areas such as 823 still become the highest average spend due to high spending in year 2002, however the spending dropped significantly after that. This suggests a declining trend, which can inform targeted re-engagement efforts or resource reallocation. Table 2 summarises the output of the query.

Table 2. Table output of Approach 2 query

Area (SCF Code)	2001	2002	2003	2004	2005	2006	2007	Average spend (2001 to 2007)
823	0	4,589	26.95	0	0	0	67.83	669
81	2,041	1,161	109	72.95	81	119	102	527
36	205	203	215	497	42	329	1,134	375
811	0	135	211	198	1,016	418	177	308
511	24	194	343	36	235	704	575	302

Question 2A.

Question 1: How do different combinations of marketing channels (Catalog vs. E-mail) influence conversion, considering the multi-channel attribution problem?

Multi-Channel Attribution Problem

In real-world campaigns, customers often receive multiple types of communication before making a purchase. A single customer may receive a catalog, an email, or both - and only some of these contacts result in orders. This overlap poses a challenge: which channel gets credit?

We begin by analysing the distribution of marketing contacts and purchases across three groups:

- **C only:** Customer received a Catalog but not an Email in the 30 days before ordering.
- **E only:** Customer received an Email but not a Catalog in the 30 days before ordering.
- **Both C&E:** Customer received both channels within the same 30-day window.

To do this, we calculate both:

- Each **contact or touchpoint** (email and/or catalogue sent) is counted to reflect marketing effort or cost,
- Each **order** is labelled based on which channel(s) the customer was exposed to before purchasing, and this depends on the attribution method. We will cover first-click, last-click and linear attribution.

We then compute the key metric *response rate per contact*, defined as:

$$\text{Response \%} = \frac{\text{Orders}}{\text{Contacts}} \times 100$$

The following SQL query produces the response rate for each bucket:

```
-- PARAMETERS
WITH settings AS (
  SELECT INTERVAL '30 days' AS time_window
), order_buckets AS (
  -- ORDERS: bucket each order into C-only, E-only or Both
  SELECT o.order_num,
         BOOL_OR(c.contact_type = 'C') AS seen_c,
         BOOL_OR(c.contact_type = 'E') AS seen_e
  FROM orders o
  JOIN contacts c
    ON c.cust_id = o.cust_id
   AND o.order_date BETWEEN c.contact_date
                           AND c.contact_date + (SELECT time_window FROM settings)
  GROUP BY o.order_num
), orders_n AS (
  SELECT CASE
    WHEN seen_c AND seen_e THEN 'Both C&E'
    WHEN seen_c THEN 'C only'
    WHEN seen_e THEN 'E only'
  END AS bucket,
         COUNT(*) AS orders
  FROM order_buckets
  GROUP BY bucket
), contact_buckets AS (
```

```

-- CONTACTS: classify each contact based on overlapping exposure
SELECT CASE
    WHEN EXISTS (
        SELECT 1
        FROM contacts c2
        WHERE c2.cust_id = c.cust_id
            AND c2.contact_type <> c.contact_type
            AND c2.contact_date BETWEEN c.contact_date
                                AND c.contact_date + (SELECT time_window FROM settings)
    ) THEN 'Both C&E'
    ELSE c.contact_type || ' only'
END AS bucket
FROM contacts c
), contacts_n AS (
    SELECT bucket, COUNT(*) AS contacts
    FROM contact_buckets
    GROUP BY bucket
)
SELECT
    c.bucket,
    c.contacts,
    COALESCE(o.orders, 0) AS orders,
    ROUND(100.0 * COALESCE(o.orders, 0) / NULLIF(c.contacts, 0), 2) AS response_pct
FROM contacts_n c
LEFT JOIN orders_n o USING (bucket)
ORDER BY response_pct DESC;

```

Then the results are shown below:

Table 3. Response Rate by Channel Exposure (Overlap Model)

Bucket	Contacts	Orders	Resp. %
C only	814,408	33,529	4.12
Both C&E	1,274,187	9,355	0.73
E only	1,300,734	7,070	0.54

Which leads to a Venn diagram of:

Channel Exposure (Proportional): Contact Share & Response Rate

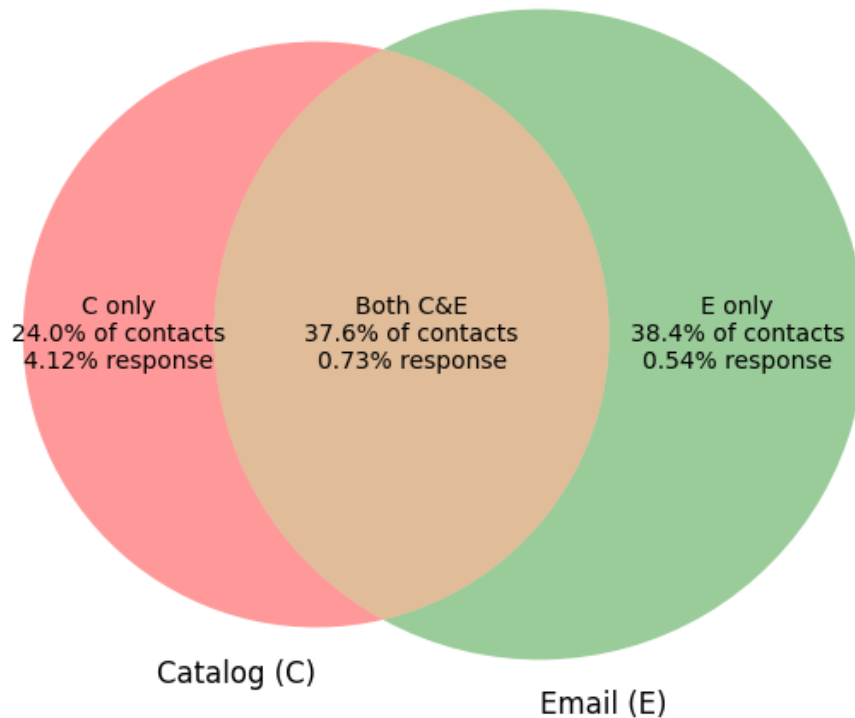


Figure 1. Proportional Venn diagram of contact share and response rate

Table 1 and Figure 1 show that catalog-only contacts performed best (4.12% response) even with the least number of contacts, while the Both C&E group had a diluted response (0.73%), and email-only contacts performed worst (0.54%). However, this does not address attribution - it simply observes the response rate without knowing the path's order.

To assign credit to understand the attributions more closely, we implement three standard attribution models:

- **First-Click Attribution:** Gives full credit to the first channel the customer interacted with.
- **Last-Click Attribution:** Gives full credit to the last channel before purchase (common in Google Analytics).
- **Linear Attribution:** Splits credit equally across all touchpoints within the attribution window.

The following SQL code creates the attribution models:

First-click:

```
-- PARAMETER
WITH settings AS (SELECT INTERVAL '30 days' AS win)
, contact_buckets AS (
  -- CONTACTS: total conatcts
  SELECT contact_type AS bucket,
         COUNT(*)      AS contacts
  FROM contacts
```

```

GROUP BY contact_type
)
, order_buckets AS (
-- ORDERS: assign each order to earliest contact
SELECT order_num,
        contact_type AS bucket
FROM (
    SELECT o.order_num,
           c.contact_type,
           ROW_NUMBER() OVER (PARTITION BY o.order_num
                              ORDER BY c.contact_date ASC) AS rnk
    FROM orders o
    JOIN contacts c
      ON c.cust_id = o.cust_id
      AND o.order_date BETWEEN c.contact_date
                              AND c.contact_date + (SELECT win FROM settings)
) sub
WHERE rnk = 1
)
, orders_n AS (
-- ORDERS: summarised by channel
SELECT bucket, COUNT(*) AS orders
FROM order_buckets
GROUP BY bucket
)
SELECT
    c.bucket,
    c.contacts,
    COALESCE(o.orders, 0) AS orders,
    ROUND(100.0 * COALESCE(o.orders, 0)::numeric / NULLIF(c.contacts, 0), 2) AS response_pct
FROM contact_buckets c
LEFT JOIN orders_n o USING (bucket)
ORDER BY response_pct DESC;

```

Last-click:

[illegible]

```

        FROM orders o
        JOIN contacts c
          ON c.cust_id = o.cust_id
         AND o.order_date BETWEEN c.contact_date
                                AND c.contact_date + (SELECT win FROM settings)
      ) sub
    WHERE rnk = 1
  )
, orders_n AS (
  -- ORDERS: summarised by channel
  SELECT bucket, COUNT(*) AS orders
  FROM order_buckets
  GROUP BY bucket
)
SELECT
  c.bucket,
  c.contacts,
  COALESCE(o.orders, 0) AS orders,
  ROUND(100.0 * COALESCE(o.orders, 0)::numeric / NULLIF(c.contacts, 0), 2) AS response_pct
FROM contact_buckets c
LEFT JOIN orders_n o USING (bucket)
ORDER BY response_pct DESC;

```

Linear:

```

-- PARAMETER
WITH settings AS (SELECT INTERVAL '30 days' AS win)
, contact_buckets AS (
  -- CONTACTS: total contacts
  SELECT contact_type AS bucket,
         COUNT(*)      AS contacts
  FROM contacts
  GROUP BY contact_type
)
, touch_paths AS (
  -- TOUCHES: identify all qualifying touches for each order
  SELECT
    o.order_num,
    c.contact_type AS bucket,
    COUNT(*) OVER (PARTITION BY o.order_num) AS touch_count
  FROM orders o
  JOIN contacts c
    ON c.cust_id = o.cust_id
   AND o.order_date BETWEEN c.contact_date
                           AND c.contact_date + (SELECT win FROM settings)
)
, order_weights AS (
  -- CREDIT: each touch gets 1 / #touches for that order
  SELECT bucket,
         SUM(1.0 / touch_count) AS orders_equiv
  FROM touch_paths
  GROUP BY bucket
)
SELECT

```

```

c.bucket,
c.contacts,
COALESCE(w.orders_equiv, 0)::numeric(10,2) AS orders_equiv,
ROUND(100.0 * COALESCE(w.orders_equiv, 0)::numeric / NULLIF(c.contacts, 0), 2) AS resp_pct
FROM   contact_buckets c
LEFT   JOIN order_weights w USING (bucket)
ORDER  BY resp_pct DESC;

```

The results of all three models are shown below:

Table 4. Comparison of Attribution Models: First-Click, Last-Click, and Linear

Channel	Contacts	First-Click		Last-Click		Linear	
		Orders	Resp.%	Orders	Resp.%	Orders	Resp.%
Catalog (C)	1,021,014	35,253	3.45	34,848	3.41	35,322.78	3.46
E-mail (E)	2,368,315	14,701	0.62	15,106	0.64	14,631.22	0.62

Interpretation

All three attribution models produce consistent findings:

- **Catalog (C)** outperforms Email (E) on a per-contact basis in every model. This aligns with the company's historical strength in catalog marketing, as highlighted in the dataset documentation.
- **Last-Click** gives slightly more credit to Email (0.64% vs. 0.62%) since it often delivers the final message before purchase.
- **Linear** distributes credit more evenly, but the overall ranking does not change - catalog remains the more effective driver of conversions per contact.

This suggests that regardless of attribution rule used, catalog performs better on a per-contact basis. However, when evaluating campaign strategy, cost and timing should also be considered - for example, email may still be more efficient due to its lower cost.

Question 2: What is the maximum cost per email contact such that its profit-per-contact matches that of catalog, assuming the normalised cost of \$1 per catalog mailing.

To ensure a fair comparison between catalog and email campaigns, we compute the *breakeven cost per email contact* - the maximum cost at which email remains as profitable per contact as catalog. A key advantage of email marketing lies in its relatively low cost and rapid deployment. As digital marketing continues to mature, channels like email offer an efficient alternative to physical mailings, contributing to higher ROI.

Based on the response rates from our last-click attribution analysis, we observe:

- Catalog yields a profit of **\$0.0239** per contact, assuming a cost of \$1.00 per piece and profit of \$30 per order.
- Email, under the same profit assumption, achieves the same profit-per-contact if its cost does not exceed **\$0.1674**.

We solve for this value by equating the profit per contact formulas for both channels:

$$\text{Breakeven Email Cost} = 1 - 30 \times (r_C - r_E)$$

Where r_C and r_E are the response rates for catalog and email respectively. This equation ensures that:

$$\text{Profit per Contact (Email)} = \text{Profit per Contact (Catalog)}$$

Substituting the observed values:

$$c_E = 1 - 30 \times (0.0341 - 0.0064) \approx \boxed{0.1674}$$

Thus, for email to be equally profitable per contact as catalog, its cost per message must not exceed \$0.1674, and this is a good guideline on how much to spend on emails.

Even when the cost per email is increased to the breakeven value of \$0.1674—ensuring equal profit per contact—the overall **ROI remains higher for email**, this is because ROI compares profit to the total cost base, and email still benefits from larger scale and lower absolute cost per contact compared to catalog.

As shown in Table 3, email delivers a return of **14%**, while catalog achieves just **2%**, despite both generating an identical \$0.02 profit per contact. This highlights email's superior *efficiency* in resource allocation, even when its profit contribution per contact is matched.

Table 5. ROI Comparison Using Breakeven Email Cost

Channel	Contacts	Orders	ROI	Profit per Contact
Catalog (C)	1,021,014	34,848	2%	\$0.02
Email (E)	2,368,315	15,106	14%	\$0.02

The table is calculated with the SQL code below:

```
-- PARAMETERS
WITH settings AS (SELECT INTERVAL '30 days' AS win)
, contact_buckets AS (
-- CONTACTS: total contacts
  SELECT contact_type AS bucket,
         COUNT(*)      AS contacts
  FROM contacts
  GROUP BY contact_type
),
-- ORDER ATTRIBUTION: assign each order to its last-click contact
order_buckets AS (
  SELECT order_num,
         contact_type AS bucket
  FROM (
    SELECT o.order_num,
           c.contact_type,
           ROW_NUMBER() OVER (PARTITION BY o.order_num
                              ORDER BY c.contact_date DESC) AS rnk
    FROM orders o
    JOIN contacts c
      ON c.cust_id = o.cust_id
      AND o.order_date BETWEEN c.contact_date
                             AND c.contact_date + (SELECT win FROM settings)
  ) sub
  WHERE rnk = 1
),
-- TOTAL ORDERS BY CHANNEL
orders_n AS (
```

```

SELECT bucket, COUNT(*) AS orders
FROM order_buckets
GROUP BY bucket
),
-- ROI CALCULATION
roi_calc AS (
  SELECT
    c.bucket,
    c.contacts,
    COALESCE(o.orders, 0) AS orders,
    CASE c.bucket
      WHEN 'C' THEN 1.00
      WHEN 'E' THEN 0.16742795867733120580
      ELSE 0.00
    END AS cost_per_contact,
    30.0 AS profit_per_order
  FROM contact_buckets c
  LEFT JOIN orders_n o USING (bucket)
)
-- FINAL METRICS
SELECT
  bucket,
  contacts,
  orders,
  ROUND(orders * profit_per_order, 2) AS total_profit,
  ROUND(contacts * cost_per_contact, 2) AS total_cost,
  ROUND((orders * profit_per_order - contacts * cost_per_contact) / NULLIF(contacts * cost_per_contact, 0), 2) AS roi,
  ROUND((orders * profit_per_order - contacts * cost_per_contact) / NULLIF(contacts, 0), 2) AS roi_scaled
FROM roi_calc
ORDER BY roi DESC;

```

As a further comparison, we substituted the observed first-click response rate to the breakeven calculation:

$$c_E = 1 - 30 \times (0.0345 - 0.0062) \approx 0.151$$

This lower breakeven value reflects the larger difference between the catalog and email response rates under first-click attribution. It suggests a safety margin when planning campaigns, rather than spending close to the maximum breakeven cost derived from last-click attribution. Email remains more cost-efficient, but overspending on it reduces the margin advantage.

Overall, these findings suggest that while the well-established catalog remains more effective in driving conversions, emails can offer greater scalability and ROI, particularly when cost efficiency is prioritised.

Question 2B

This problem is solved using Python file and is submitted in *notebook file (Question2B.ipynb)*.

Question 3A

The calculation of CLV starts with summarising the order per customer which includes number of purchases, first time purchase, last purchase, duration from the last purchase, total and average value generated. We assumed the value generated by a customer is the monetary value without considering acquisition costs or churn rate. Additionally, we did not make any assumptions about future orders, the calculation is based purely on past transactions with a monthly discount rate assumption $i = 0.01$ (which corresponds to an annual rate of approximately 12%), appropriate for a consumer business

We observed that there are two types of customers which include: one-time buyer and repeated customers. The customer lifetime value is divided based these types of customers because one-time buyer one purchased once in the past which can be calculated using the discounted formula to adjust past revenue to present value.

$$CLV_{one-time\ buyer} = \frac{M}{(1+i)^n}$$

Where:

- M = monetary value
- n = number of periods (e.g., months or transaction gaps) since the purchase
- i = discount rate per period

Meanwhile, the repeated customer's lifetime value is calculated using the infinite time value that accounts for recurring transactions

$$CLV_{repeat\ customer} = \frac{(1+i)M}{(1-r+i)}$$

Where:

- M = monetary value
- r = retention rate
- i = discount rate per period

We need to obtain the retention rate for the repeat customer, we computed retention rates over the first 50 purchases showing how many customers who made at least n orders also made at least n+1 orders.using the following SQL.

```

WITH customer_order_counts AS (
    SELECT
        cust_id,
        COUNT(DISTINCT order_num) AS num_orders
    FROM orders
    GROUP BY cust_id
),
series AS (
    SELECT generate_series(1, 50) AS n
),
r_table AS (
    SELECT
        s.n,
        COUNT(CASE WHEN c.num_orders >= s.n THEN 1 END) AS
base_n,
        COUNT(CASE WHEN c.num_orders >= s.n + 1 THEN 1 END) AS
base_n_plus_1
    FROM series s
    CROSS JOIN customer_order_counts c
    GROUP BY s.n
)
SELECT
    n,
    base_n,
    base_n_plus_1,
    ROUND(base_n_plus_1::numeric / NULLIF(base_n, 0), 4) AS
retention_rate
FROM r_table
ORDER BY n;

```

The computed retention rates stabilized around 0.80 after the early purchases, indicating high loyalty among recurring customers. Then, since it is in e-commerce context, the t is accounted for number of transactions hence the discounted factor needs to be adjusted based on the time of each customer. The adjusted discounted rate is calculated by this formula

$$i = (1 + r_{\text{annual}})^{\left(\frac{TBT}{12}\right)} - 1$$

Where:

- r_{annual} = assumed to be 12%
- TBT = Time between transaction
- i = discount rate per period

The time between transaction per customer is calculated by using the following query (while creating new table)

```
CREATE TABLE avg_time_between_order AS
(WITH distinct_orders AS (
    -- Step 1: Get distinct order dates (ignore item-level
    details)
    SELECT
        cust_id,
        order_num,
        MIN(order_date) AS order_date -- Take the first date
    for each distinct order
    FROM lines
    GROUP BY cust_id, order_num
),
transaction_times AS (
    -- Step 2: Use LEAD to get the next order date for each
    order
    SELECT
        cust_id,
        order_num,
        order_date,
        LEAD(order_date) OVER (PARTITION BY cust_id ORDER BY
    order_date) AS next_order_date
    FROM distinct_orders
),
time_differences AS (
    -- Step 3: Calculate the time difference between
    consecutive orders (in months)
    SELECT
        cust_id,
        -- Subtract the dates directly to get the number of
        days, then divide by 30 to convert to months
        (next_order_date - order_date) / 30.0 AS
    months_between_orders
    FROM transaction_times
    WHERE next_order_date IS NOT NULL -- Exclude the last
    order for each customer
),
customer_avg_time AS (
    -- Step 4: Calculate average time between purchases for
    each customer
    SELECT
        cust_id,
        AVG(months_between_orders) AS avg_time_between_orders
```

```
        FROM time_differences
        GROUP BY cust_id
    )
-- Final output: Get the results with average time between
purchases
SELECT
    cust_id,
    avg_time_between_orders
FROM customer_avg_time;
)
```

After gaining the values needed for the calculation, we calculate the lifetime value with the following query

```
WITH order_summary AS
(SELECT cust_id,
    COUNT(DISTINCT order_num) AS num_of_purch,
    MIN(order_date) AS first_purch,
    MAX(order_date) AS last_purch,
    ((SELECT MAX(order_date) FROM lines) -
MAX(order_date))/30 AS dur_since_last_purch,
    SUM(line_dollars) AS total_value,
    SUM(line_dollars)/COUNT(DISTINCT order_num) AS
avg_value
FROM lines
GROUP BY 1
ORDER BY 2 DESC),

-- For 1 time customer, we assume the discounted rate to be 1%
(12% annually)
one_time_cust AS(
SELECT *,
    avg_value/POWER(1+0.01, dur_since_last_purch) AS
clv_1_time
FROM order_summary
WHERE num_of_purch = 1),

-- For repeat customers, we assume the discounted rate to be
 $i = (1 + r_{\text{annual}})^{(TBT/12)} - 1$ 
-- where TBT = Time between transaction (in months)
repeat_cust_rate AS(
SELECT o.cust_id,
    o.num_of_purch,
    o.last_purch,
    o.dur_since_last_purch,
```

```

        o.total_value,
        o.avg_value,
        POWER(1+0.12, a.avg_time_between_orders/12)-1 AS
discount_rate
FROM order_summary AS o
INNER JOIN avg_time_between_order AS a
ON o.cust_id = a.cust_id
WHERE o.num_of_purch > 1),

repeat_cust AS(
SELECT *,
        avg_value*(1+discount_rate)/(1-0.8+discount_rate) AS
clv_repeat
FROM repeat_cust_rate
WHERE num_of_purch > 1),

combine AS
((SELECT cust_id, clv_1_time FROM one_time_cust)
UNION ALL
(SELECT cust_id, clv_repeat FROM repeat_cust))

(SELECT ROUND(avg(clv_repeat)::integer,2) AS clv,
        'Repeat customer' AS category
FROM repeat_cust)
UNION
(SELECT ROUND(avg(clv_1_time)::integer,2) AS clv,
        'One-time buyer' AS category
FROM one_time_cust)
UNION
(SELECT ROUND(avg(clv_1_time)::integer,2) AS clv,
        'Overall' AS category
FROM combine)

```

The estimated value of M (Average contribution margin) for recurring customers is \$273 and for one-time buyer is \$47. The repeated customer has higher value than the one-time buyers. Based on this numbers, it can be implied that the purchase frequency plays a pivotal role in determining the value of a customer. Customers who make frequent purchases is likely to have a higher value compared to one-time buyer. If we combine all the customers, the CLV is \$149.

Question 3B

To assist management better understand the customer value and tailoring marketing strategies, we propose a structured project to segment customers by Customer Lifetime

Value (CLV). The objective is to uncover which customer types drive the most value and to guide differentiated acquisition, retention, and engagement strategies based on those insights.

Our analysis will begin by leveraging data from both the transactional tables (*lines*, *orders*, *summary*) and the contact file (*contacts*). Segmentation will be based on behavioral, monetary, and channel interaction characteristics. From the transactional data, we will classify customers by their purchase frequency and recency using the variables `order_num` and `order_date`. This will allow us to distinguish one-time buyers from occasional and high frequency purchasers and assess how recently their last transaction took place. We will compute the average monetary value per customer using the `line_dollars` variable, enabling us to segment customers into low, medium, and high spenders based on historical order value. Acquisition timing, drawn from the `acq_date` variable, will allow us to differentiate between recently acquired and long-tenured customers, recognizing that tenure can influence retention patterns.

Crucially, we will incorporate channel preferences directly using the *contacts* file. By linking this contact data to each customer's purchasing history (via `cust_id`), we can segment customers based on their dominant communication channel or multichannel engagement profile. This is particularly useful, as prior contact method may influence both purchasing behavior and responsiveness to future marketing efforts. Additionally, we can combine this customer segment with some demographic attributes provided by the summary table. This can be included age, income, occupation, family size, and interest to get an in-depth characteristic of each customer segment (for example, a high spender segment have income more than \$75,000, lawyers, aged 35-44 years old).

After defining these customer segments, we will compute historical retention rates within each group, using a methodology similar to our earlier retention analysis. Specifically, we will track customer orders over time and estimate repeat purchase probabilities conditional on prior purchase counts. This allows us to capture behavioral differences in retention across segments such as one-time buyers, frequent buyers, and channel-based cohorts.

With these differentiated retention rates and customer characteristics, we will calculate segment-specific CLVs using both historical data and steady-state models. Then, we could test marketing strategies targeted to specific segments via A/B experiments, measuring the impact on retention, order value, and engagement.

The final output of this project will be insights into which customer segments are most valuable and why. Management will be prepared to prioritize high-CLV segments for loyalty initiatives, tailor communication strategies by channel preference, and deploy re-engagement campaigns where they can most effectively lift customer lifetime value.