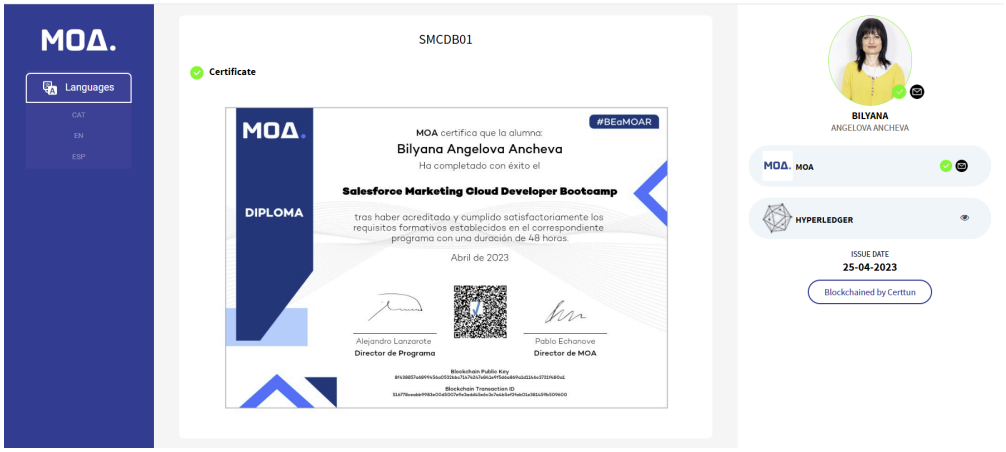
INSURANCE INDUSTRY USE CASE

Author: Bilyana Ancheva

Client: Insurance, Madrid

June 2024

About Bilyana Ancheva



Description

Salesforce Marketing Cloud Developer Bootcamp

- Horas: 48h
- Modalidad: Online.
- Objetivos: -Entender el funcionamiento de la plataforma Marketing Cloud a nivel técnico, - Conocer los principales lenguajes de programación en Marketing Cloud, - Aprender a personalizar a nivel avanzado emails y landing pages, así como SMS o Push notifications gracias a AMPscript, - Conseguir autonomía para la creación de la segmentación de los públicos objetivos en Marketing Cloud, - Aprovechar todas las capacidades que ofrece Marketing Cloud en una única plataforma, - Mejorar el perfil y las competencias profesionales de las personas que realicen esta formación.
- Temario:
  Introducción a Marketing Cloud: - Ecosistema de Salesforce, - Licenciamiento Marketing Cloud, - Marketing Cloud Overview, - Administración Marketing Cloud, - Modelo de datos en Marketing Cloud.
  Integraciones en Marketing Cloud: - Integraciones en tiempo real, - Integraciones en lotes (batch), - Acceso al modelo de datos, - SQL & Query Studio, - Análisis tracking envíos, - Gestión consentimientos Marketing Cloud.
  Personalización avanzada: - AMPscript y GTL, - CloudPages y SSJS, - Personalización avanzada de contenidos, - Construcción email avanzados, - Pruebas UAT y ejemplos reales.
  Certificación Developer y Workshops: - Configuración BU Marketing Cloud, - Integraciones Marketing Cloud, - Simulaciones llamadas POSTMAN, - Construcción Emails personalizados, - Consultas SQL en QueryStudio, - Certificación Developer
- El claustro de MOA son profesionales contrastados en el mercado con amplia experiencia profesional y académica como Alejandro Lanzarote, Álvaro Méndez de Vigo, Jorge Rashmawi, Juan Vicente López-Barrajón y Miguel Deza.

Project Objective:

The project is designed for a marketing campaign of Insurance company in Madrid, aiming to promote a new car insurance product. The campaign includes audience segmentation, process automation, and creating a multichannel communication strategy (emails and SMS).

Technologies and Tools:

1. SQL:
    - Used to create queries for customer segmentation based on specific criteria:
        - Age (25–40 years old).
        - Geographical location (Madrid and Barcelona).
        - Purchase history (car insurance).
        - Exclusion of active customers with insurance policies or open claims.
    - The code for the queries is organized in a GitHub repository for convenience.
2. Salesforce Marketing Cloud (SFMC):
    - Journey Builder to create a multi-step campaign with emails and SMS.

- Automation Studio to automate processes like segmentation, exporting data to FTP, and managing customer data.

3. Excel:
   - Used for data preparation and corrections before importing into SFMC.
4. SFTP:
   - Facilitates file transfers between systems for data integration.
5. Lucidchart:
   - Used for visualizing and planning customer journeys.

Project Phases:

1. Audience Segmentation:
   - Creating customized customer lists based on defined conditions.
   - Data processing and transformation using Excel and SQL.
2. Campaign Design:
   - Multi-wave strategy:
     - First wave: Introduction email about the new product.
     - Second wave: Reminder email for customers who haven't made a purchase.

- Third wave: Final attempt to reach out via SMS and calls.

3. Automation and Integration:
    - Exporting customer lists to FTP for call center outreach.
    - Automated data management using SFMC.
4. Reporting and Analytics:
    - Utilizing SFMC analytics tools to track engagement (email opens, clicks, and sales).

The project integrates automation, data processing, and personalization to deliver an effective customer acquisition campaign. The primary technologies employed include SQL for data processing, Salesforce Marketing Cloud for automation and marketing journeys, and integration with external systems via SFTP.

1 AUDIENCE SEGMENTATION

We are going to activate an email and SMS campaign to launch a new car insurance product. The database contains information about the customers, including details such as age, location, other active insurance policies, and previous insurance history.

The goal is to segment the audience to send an email and SMS to customers who meet certain criteria.

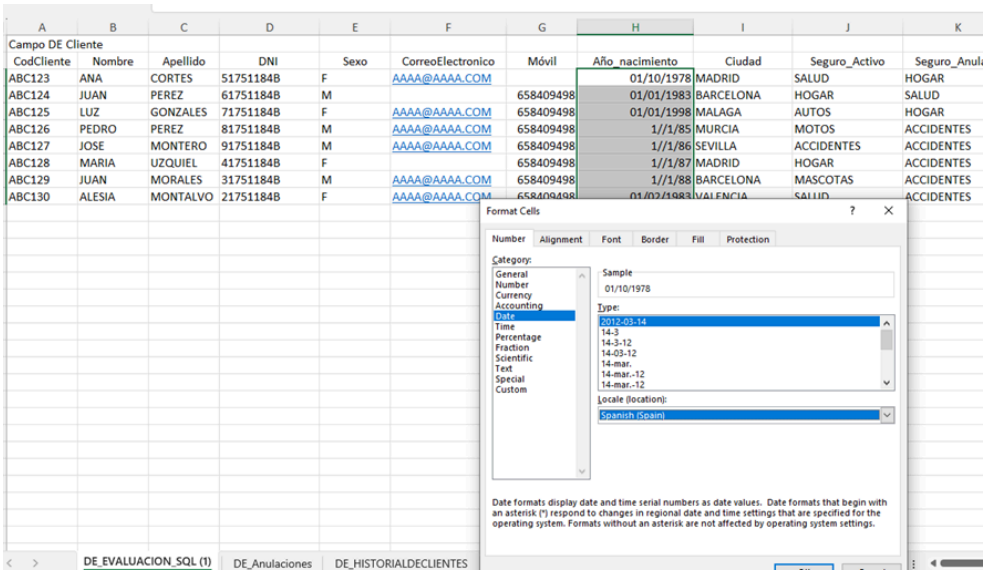
Create an SQL query to perform the segmentation. All the information is located in a table called `HISTORIAL_DE_CLIENTES`:

1. Customers aged between 25 and 40.
2. Customers living in Madrid and Barcelona.
3. Customers who have purchased car insurance in the past.
4. Exclude customers who currently have an active car insurance policy.
5. The SQL query should retrieve the relevant information from the table and generate a new table of emails and SMS for customers who meet the specified criteria.
6. Additionally, there is another table with customers' claims (`SINIESTROS`), and we must exclude from the campaign any customers who have an open claim.

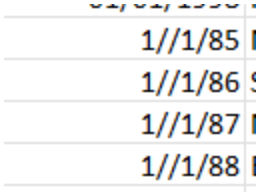
Implementation:

I begin by reviewing and analyzing the database provided by the client in an Excel file. This step is important to ensure that the data being imported into Marketing Cloud as a data extension is correctly mapped and does not cause data type errors. I make sure of the following:

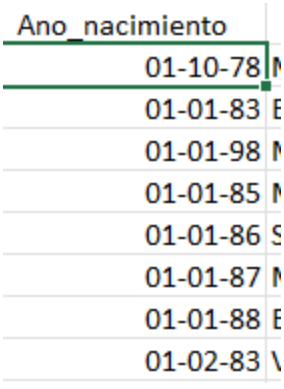
- The column field names do not contain spaces, accents, or "ñ" to guarantee a successful import.
- Before importing, I need to convert the file from Excel format to `.csv` format.
- Therefore, I rename the field `año_nacimiento` to `ano_nacimiento` and remove the accent from the field “Móvil.”
- I correct the data types for all columns in the file, paying particular attention to numbers and dates.
- I fix issues in the `ano_nacimiento` column. I notice that there are full dates. Since there are no additional requirements if only the year is needed without the full date, I leave it in the day-month-year format.



- I apply **Format Cells** to the `año_nacimiento` column, but some rows cannot be formatted as dates. For these date fields, I need to use more advanced formatting in Excel, as it seems Excel does not recognize them as dates due to some issues.



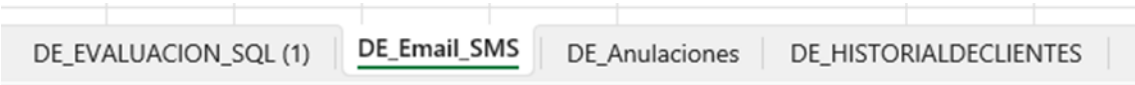
  - Therefore, I apply Excel's **Datevalue** and **Substitute** functions, and the result is a correctly formatted date.
  - In the **Data** tab, using **Text-to-Columns**, I assign the correct date format in the final step.



  - [x] Assign the correct date format based on the country when importing the CSV. For Spain, the UK date format can be used.

- ☑ When performing the import using the **Import Wizard**, select **Delimiter: Comma**.

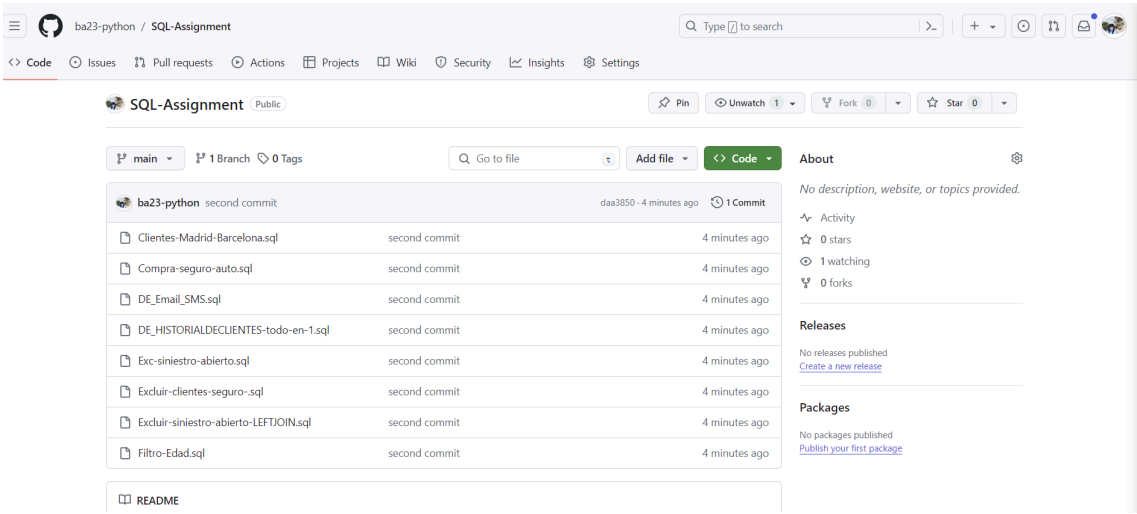
- ☑ I have created additional sheets, **DE_EMAIL_SMS** and **DE_HISTORIALCLIENTES**, in the CSV file.



SQLs in my GitHub repository

I am attaching a GitHub repository containing all the described SQL queries.

https://github.com/ba23-python/SQL-Assignment



I have created separate SQL segments for each condition described in each task point. Additionally, I have created a combined query for all tasks in one, which can be found in a new tab of the spreadsheet named **DE_HISTORIAL_DE_CLIENTES-todo-en-1.sql**.

- **Filtro-edad.sql** -Customers aged between 25 and 40 years.

SQL-Assignment / Filtro-Edad.sql

ba23-python second commit

Code Blame 36 lines (29 loc) · 885 Bytes

```
1     Clientes con edades entre 25 y 40 años.
2
3   SELECT
4   CodCliente,
5   Nombre,
6   Apellido,
7   DNI,
8   Sexo,
9   CorreoElectronico,
10  Móvil,
11  Año_nacimiento,
12  Ciudad,
13  Seguro_Activo,
14  eguro_Anulado,
15  Si_Publicidad
16  FROM DE_Clientes
17  WHERE (YEAR(GETDATE()) - YEAR(Año_nacimiento)) BETWEEN 25 AND 40;
```

- **Clientes-Madrid-Barcelona.sql**- 2. Clients who live in Madrid and Barcelona.

SQL-Assignment / Clientes-Madrid-Barcelona.sql

ba23-python second commit

Code Blame 13 lines (9 loc) · 289 Bytes

```
1   Clientes que viven en Madrid y Barcelona
2
3   SELECT CodCliente, Nombre, Apellido, CorreoElectronico, Móvil
4   FROM DE_Clientes
5   WHERE Ciudad IN ('MADRID', 'BARCELONA');
6
7
8   la segmentacion mostrara los siguientes suscriptores:
9
10  ABC123 (Madrid)
11  ABC124 (Barcelona)
12  ABC128 (Madrid)
13  ABC129 (Barcelona)
```

- **Compra-seguro-auto.sql** - 3. Customers who have purchased car insurance in the past.

SQL-Assignment / Compra-seguro-auto.sql

ba23-python second commit

Code Blame 9 lines (6 loc) · 215 Bytes

```
1   Clientes que han comprado Seguro Auto previamente
2
3   SELECT CodCliente, Nombre, Apellido, CorreoElectronico, Móvil
4   FROM DE_Clientes
5   WHERE Seguro_Anulado = 'AUTOS';
6
7   Debe de filtrar como resultado el cliente:
8
9   ABC125
```

- **Excluir-clientes-seguro.sql**- 4. Exclude customers who currently have an active car insurance policy.

SQL-Assignment / Excluir-clientes-seguro-.sql

ba23-python second commit

Code Blame 5 lines (4 loc) · 169 Bytes

```
1   excluir clientes que tienen actualmente seguro de coche
2
3   SELECT CodCliente, Nombre, Apellido, CorreoElectronico, Móvil
4   FROM DE_Clientes
5   WHERE Seguro_Activo != 'AUTOS';
```

- **DE_Email_SMS.sql** – 5. The SQL query must retrieve the relevant information from the DE and generate a new DE with the emails and SMS details of the customers who meet the established criteria.

ba23-python second commit

Code | Blame | 37 lines (33 loc) · 1.48 KB

```
1   SELECT CorreoElectronico, Móvil
2   FROM DE_Clientes
3   WHERE
4     (DATEPART(year, GETDATE()) - DATEPART(year, Año_nacimiento)) BETWEEN 25 AND 40
5     AND Ciudad IN ('MADRID', 'BARCELONA')
6     AND CodCliente IN (
7           SELECT CodCliente
8           FROM DE_Clientes
9           WHERE Seguro_Anulado = 'AUTOS'
10      )
11    AND Seguro_Activo != 'AUTOS'
12    AND CorreoElectronico IS NOT NULL
13    AND CorreoElectronico <> ''
14    AND Móvil IS NOT NULL
15    AND Móvil <> '';
16
17  Explicación:
18  Función DATEPART: aplico la función DATEPART, que extrae partes de una fecha.
19  Aquí, se utiliza para obtener el año de la fecha actual y el año de nacimiento para calcular la edad.
20  Filtro de Ciudad: Asegura que los clientes sean de Madrid o Barcelona.
21  Seguro de automóvil anterior: la subconsulta verifica si el cliente tuvo un seguro de automóvil en el pasado.
22  Excluir seguro de automóvil actual: Garantiza que el cliente no tenga actualmente un seguro de automóvil.
23  Correo electrónico y móvil no vacíos: garantiza que los campos de correo electrónico y móvil no sean nulos ni estén vacíos.
24
25
26  El resultado de esta segmentacion seria
27  CodCliente ABC129 (JUAN MORALES) como el unico suscriptor cumple la condicion de disponer de email y movil en el mismo tiempo
28  Edad: 36 (born in 1988)
29  Cuidad: Barcelona
30  Past car insurance (Seguro_Anulado = 'AUTOS')
31  No current car insurance (Seguro_Activo != 'AUTOS')
32  Non-empty email and mobile
33
34  Los demas suscriptores
35  ABC123 (sin movil)
36  ABC124 (sin email)
37  ABC128 (sin email)
```

- **DE_HISTORIAL DE CLIENTES-todo-en-1.sql** I have created a combined query for all the tasks in one, which can be found in a new tab of the spreadsheet named **DE_HISTORIAL_DE_CLIENTES-todo-en-1.sql**.

SQL-Assignment / DE_HISTORIALDECLIENTES-todo-en-1.sql

ba23-python second commit

Code Blame 16 lines (13 loc) · 573 Bytes

```sql
-- Combinar todos los criterios en un query y guardar los datos en nueva data extension DE_HISTORIALDECLIENTES---


SELECT c.CodCliente, c.Nombre, c.Apellido, c.CorreoElectronico, c.Móvil
FROM DE_Clientes
WHERE (YEAR(GETDATE()) - YEAR(Año_nacimiento)) BETWEEN 25 AND 40
  AND Ciudad IN ('MADRID', 'BARCELONA')
  AND CodCliente IN (
        SELECT CodCliente
        FROM DE_Clientes
        WHERE Seguro_Anulado = 'AUTOS'
    )
  AND Seguro_Activo != 'AUTOS'
  AND (CorreoElectronico IS NOT NULL AND CorreoElectronico <> '')
  AND (Móvil IS NOT NULL AND Móvil <> '');
```

2. Automation & Journey

We are going to activate a campaign in SFMKT. You need to design a series of steps in an Automation and Journey for this campaign.

I divide the solution into two parts: **Automation** and **Journey Builder**.

Journey Builder-Journey Mapping

The campaign is aimed at promoting a new insurance product and consists of three waves:

- **First Wave:** Introduction of the insurance product.
- **Second Wave:** Reminder email, excluding those who have already purchased the insurance.
- **Third Wave:** Reminder for those who did not click on the second email, also excluding those who have purchased the insurance.

The email will include a CTA (Call-to-Action) button labeled “Call Us,” and we must provide the contacts of customers who click the button to the call center so they can call them. The file should be placed on an SFTP server.

The SMS will only be sent to those who did not click on the email from the second wave. The SMS is informational.

The primary channel is email, and SMS will be sent to customers who do not have an email address.

The reminder email will be sent 5 days after the first email, and the SMS will be sent 3 days after the reminder email.

I must create a 20% control group and generate a file for the SFTP.

It's important to identify the customers and the touchpoints they receive or the group they belong to, as the report must include:

Engagement:

- First wave
- Second wave
- Reminder

The databases: the customer database is in a **Data Extension** (DE) in SFMKT, but the daily updates of purchases are loaded onto the SFTP server.

Several options can be proposed, such as creating three separate journeys, one for each wave.

I'd like to discuss how this could be approached with the option of having three journeys—one journey per wave.
In this option, the goal is to clearly define the customers for each of the three waves, as required.

First Wave
The entry point for the journey would be a **Data Extension** of customers, likely coming from **Automation Studio**, refreshed and updated using a **Filter Activity** with nightly automated updates.

Second Wave
The entry point for the journey would be a **Data Extension** segmented via an **SQL Activity** in **Automation Studio**, ensuring customers who have already purchased are excluded, so only those who haven't purchased yet are targeted.

Third Wave
The same approach as the second wave. The entry point for the journey would be a **Data Extension** segmented via an **SQL Activity** in **Automation Studio**, ensuring customers who have already purchased are excluded, so only those who haven't purchased yet are targeted.

If there are three separate journeys for each wave, segmentation would need to happen in **Automation Studio**, creating a segment that only includes customers who haven't purchased, to be used as the entry **Data Extension** for **Journey 2** and **Journey 3**. It would be important to allow a 30-minute buffer between the time Automation ends and the Journey begins in **Journey Builder** to avoid potential conflicts.

Final Approach: Multi-Step Journey

Despite these considerations, I have decided to propose a **Multi-Step Journey** that incorporates all three emails and SMS into a single journey.

Solution 2:

Proposed Solution:

Multi-Step Journey with 3 Emails and SMS

Data Entry:

The data entry will be a sendable Data Extension (DE_Clientes). It is not explicitly stated where this data extension comes from, but it is assumed that it originates from Automation Studio, either via an Import Activity from the FTP or through a Filter Activity automation executed nightly.

The data extension must be mapped using the SubscriberKey in Contact Builder and Data Designer to utilize its dynamic attributes in the Contact Data for Decision Splits in Journey Builder.

Data Extensions Setup:

I will create the following empty sendable data extensions with the specified names:

- DE_Compras_1era_oleada
- DE_Compras_2a_oleada
- DE_Compras_3era_oleada

These will contain the attribute StatusCompra for each wave. They will be updated through the Update Contact activity within Journey Builder as the journey progresses.

- Update the sendable data extension DE_Compras_1era_oleada with the attribute "Status_Compra=SI" for completed purchases.

- The data extensions for each wave will be segmented using an SQL Query based on the StatusCompra attribute to ensure it equals SI. The resulting DE will then be exported to the FTP using Data Extract.

Additionally, another previously created data extension, DE_CallCenter, will store customers from the 3rd wave who clicked the CTA “Call Us.” This data extension will also be exported to the FTP using Data Extract.

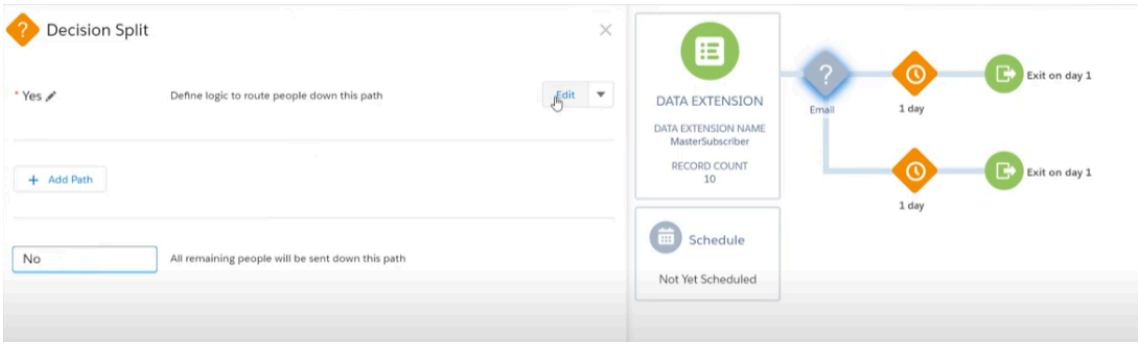
Automation Plan:

A scheduled automation will be set up to export the following data extensions to the FTP daily:

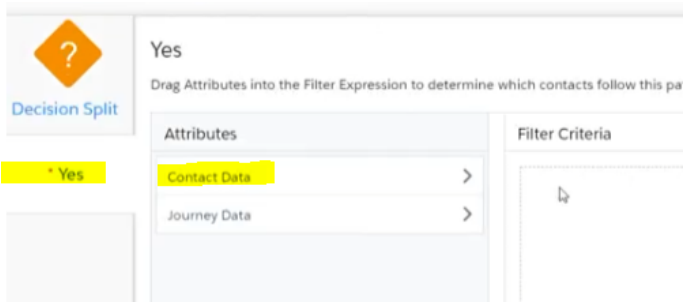
- DE_Compras_1era_oleada
- DE_Compras_2a_oleada
- DE_Compras_3era_oleada
- DE_CallCenter

Decision Splits in Journey Builder:

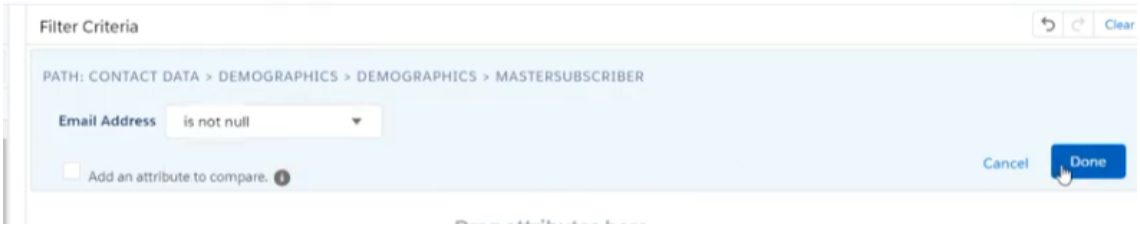
A Decision Split will filter whether an email address exists **(Yes or No).**



Since I have previously linked **Contact Data** in **Contact Builder**, the dynamic attributes from **Contact Data** are now available for use in the **Decision Split** within Journey Builder.



For the Yes branch, I click Edit, choose Contact Data for the corresponding DE, and drag Email—assigning is not null to ensure that only subscribers with an existing email address will enter the journey. (If we are confident that the DE is of good quality and fully populated with emails and mobile numbers, this step may not be necessary, and we can skip this Split.)



Engagement Split

This split will monitor clicks in **Email 2** and **Email 3**, redirecting the flow into two branches depending on whether a click has occurred or not.

Waits

1. **Before each Decision Split and Engagement Split**: Place a **Wait** to allow time for data analysis based on collected information.
2. **5-day Wait**: Before sending the second email.
3. **3-day Wait**: Before sending the SMS.
4. **1-day Wait**: Before each **Exit**.

Update Contact

The **Update Contact** feature allows us to update or modify a Marketing Cloud contact record directly within Journey Builder by using the **Update Contact** activity.

- It updates the contact to modify an attribute value when a contact reaches this point in the journey.
- Select a **sendable data extension** with a **Send Relationship** that has been previously created, and configure a **static value** to overwrite this value for all contacts reaching this activity.
- This type of **Update Contact** is known as updating with a **"Static Value"** and does not require the CRM to be connected to Marketing Cloud.

Other Update Contact Option

The alternative **Update Contact** method is more commonly used when the CRM is connected, but in our case, a connection is not necessary.

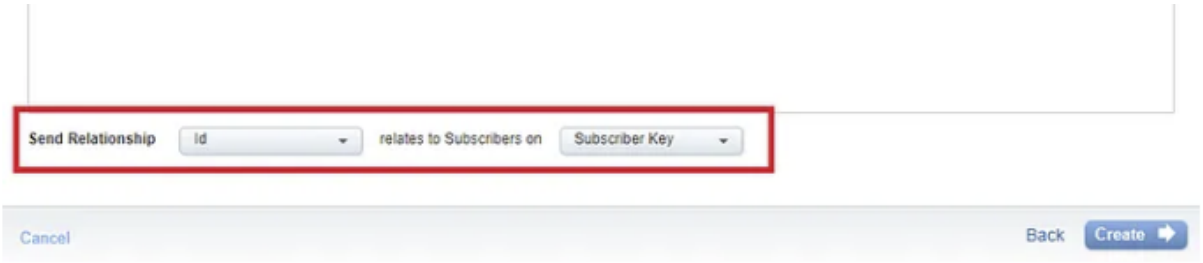
Documentation:

https://help.salesforce.com/s/articleView?id=sf.mc_jb_update_a_contact.htm&type=5

In Our Use Case: Update Contact

The **Update Contact** will update the attribute **Status_Compra** within the data extensions for each wave.

- The data extensions must be of type **"Sendable"** and must have a **"Send Relationship"** established.
- Each **Update Contact** activity will be linked to the corresponding data extension for the relevant wave:
  - **DE_Compras_1era_oleada**
  - **DE_Compras_2a_oleada**
  - **DE_Compras_3era_oleada**
  - **DE_CallCenter**



Attribute: Status_Compra

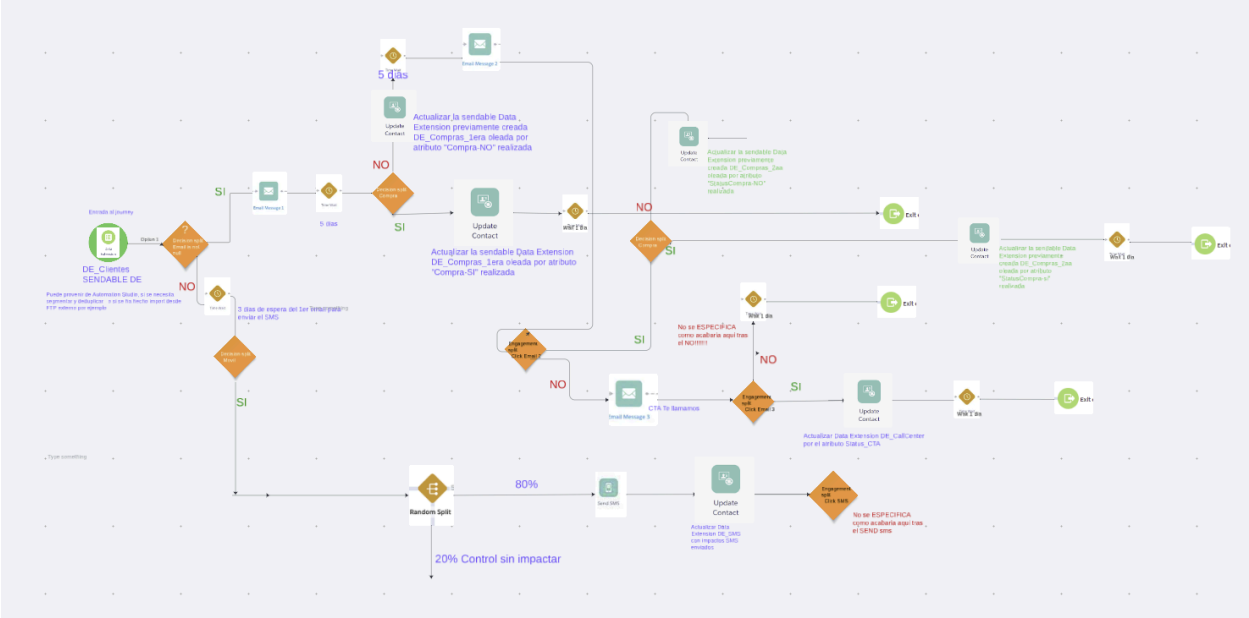
The attribute **Status_Compra** will be of type **Boolean**:

- **True**: Indicates that the purchase has been made.
- **False**: Indicates that the purchase has not been made.

Journey Diagram

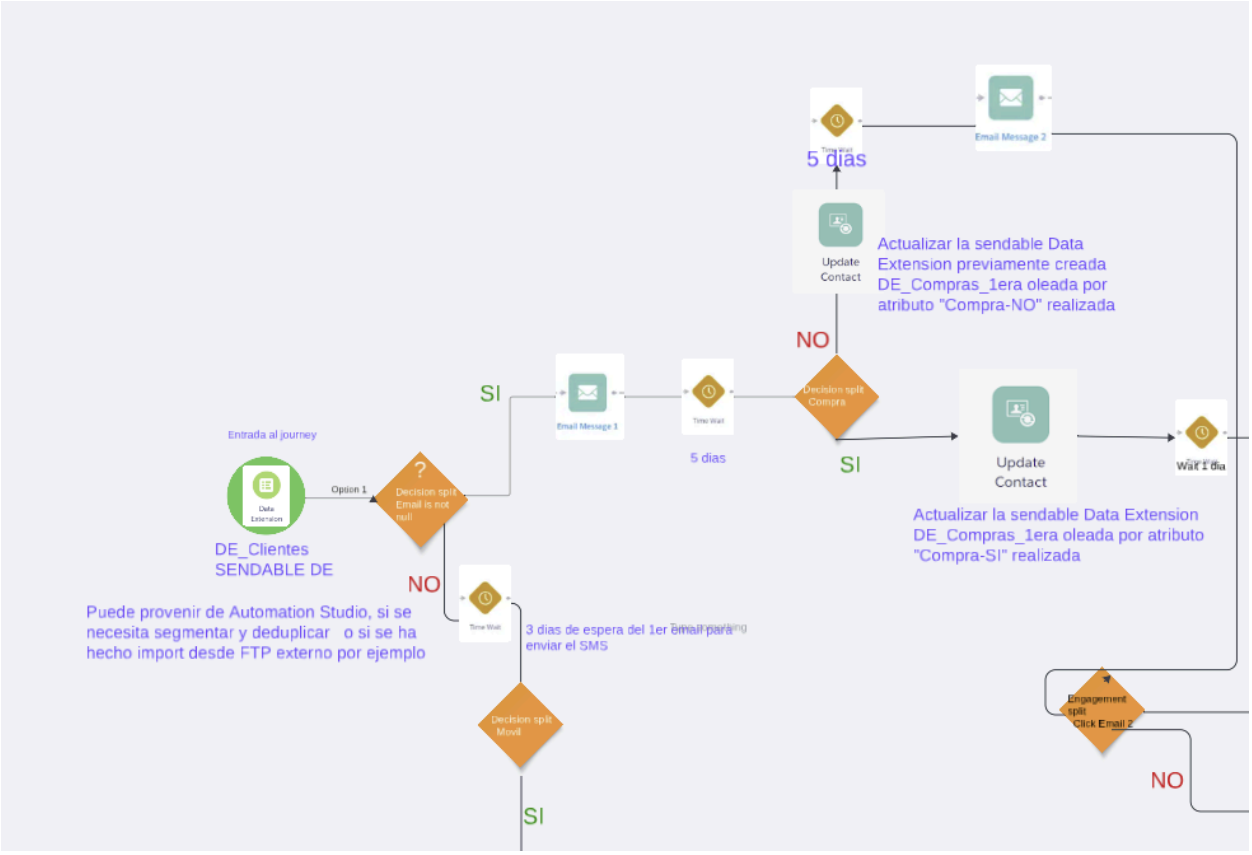
The entire journey is available on **Lucidchart**. I am sharing the link:

https://lucid.app/lucidspark/6a9e4550-1db0-434b-a865-1ed3a9f05f92/edit?invitationId=inv_afa24279-0a7d-4e04-8a18-4924cf86247a&page=0_0#



Partial description of the Journey

Email 1 First wave



Journey Entry: Sendable Data Extension - DE_Clientes

The journey entry point is the sendable **Data Extension DE_Clientes**.

- It is only specified that this data extension is located in **Marketing Cloud**, but not where exactly. It is possible that it originates from **Automation Studio**, where it has been refreshed and updated using a **Filter Activity**, or perhaps it has passed through a **subscriber deduplication automation** to prevent duplicate communications.

Decision Split: EMAIL IS NOT NULL

As the first filter, a **Decision Split** will check if the subscriber's email is not null.

- If the data extension entry comes directly from a **CloudPage** without passing through Automation Studio (which is a possible scenario), this Decision Split will ensure that the subscriber's email exists upon entering the journey.
- Subscribers with an email will be routed to receive **Email 1**.

Email 1

Email 1 will be sent to subscribers who have an email address.

No Email Branch

From the **Decision Split**, subscribers without an email will be routed to another branch to check via another **Decision Split** whether they have a mobile number.

Wait Times

1. **5-Day Wait**: A wait of 5 days will be added before each **Decision Split** and **Engagement Split** to allow time for analyzing the collected data.
2. **3-Day Wait**: A wait of 3 days will be placed between sending **Email 1** and the **Purchase Decision Split**, allowing enough time for the customer to make a purchase if they wish to do so from **Email 1**.

2. 5-Day Wait Before Sending the Second Email

A **5-day wait** will be added before sending the second email, as per the specified requirements.

This ensures sufficient time for customers to engage with the first email and potentially complete the desired action before receiving a reminder.

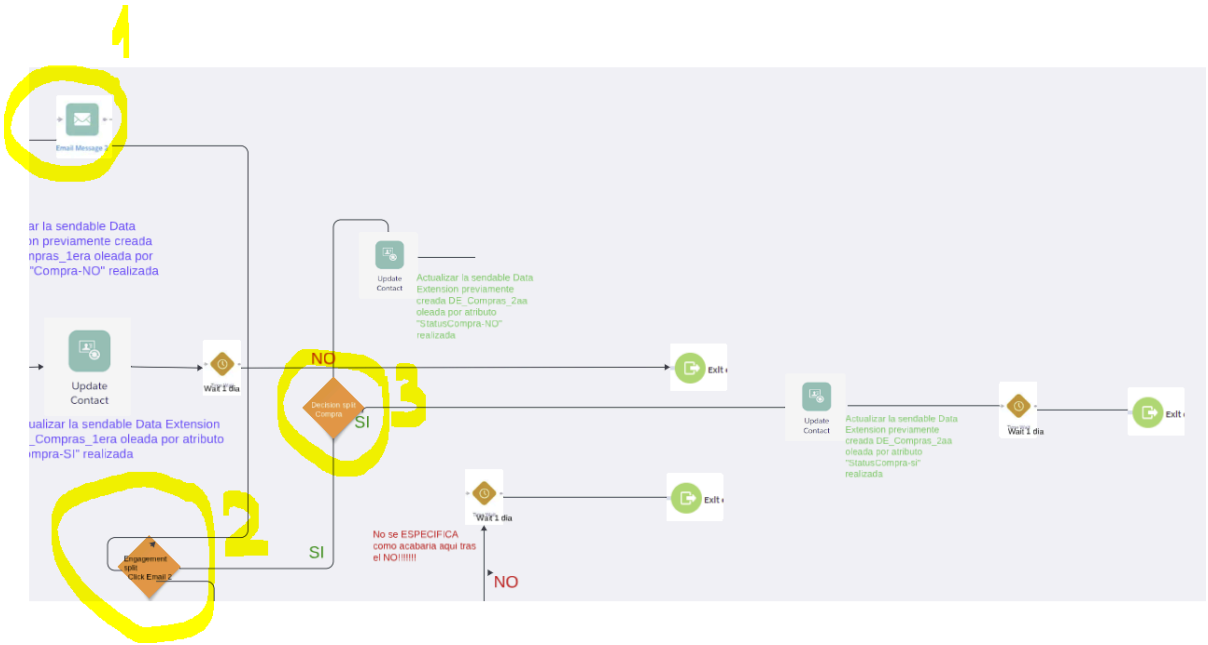
Update Contact: First Wave

In this first wave, the **Update Contact** activity will update the **DE_Compras_1eraoleada** data extension with the static attribute **Status_Compra** set to **true** or **false**:

- **True**: If a purchase has been made.
- **False**: If no purchase has been made.

This ensures that the purchase status is accurately recorded for each contact during the journey.

Second wave Email 2



Email 2

- **Email 2** will be sent **5 days** after the sending of **Email 1**, thanks to the **Wait** step of 5 days.

Engagement Split: Clicks on Email 2

- An **Engagement Split** will monitor clicks on **Email 2**.
- If subscribers have clicked, they will be routed to a **Decision Split** to verify whether they have completed a purchase, based on the **Status_Compra** attribute.

Wait Between Engagement Split and Decision Split

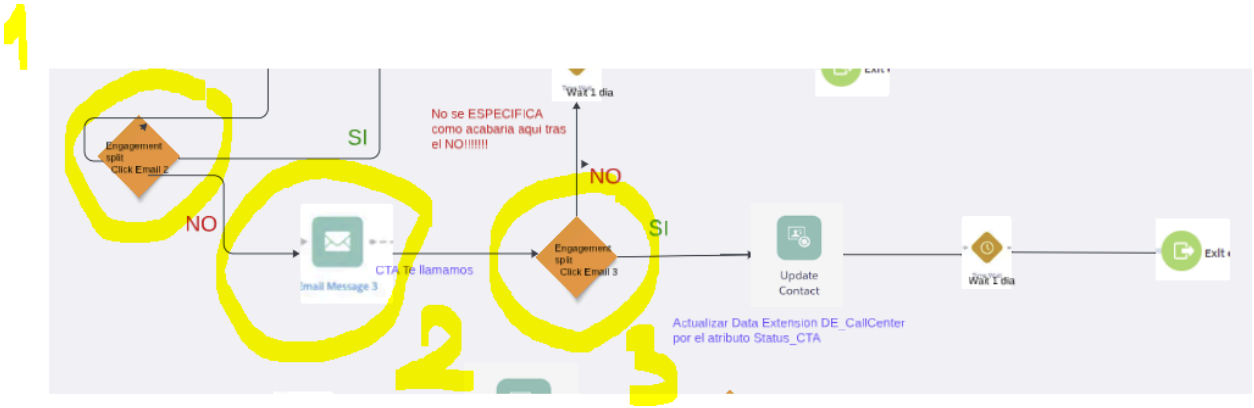
- A **Wait** will be placed between the **Engagement Split** and the **Decision Split** to give Marketing Cloud sufficient time to analyze the collected data and ensure accurate processing of clicks.

Update Contact

- Use **Update Contact** to update the previously created sendable data extension **DE_Compras_2a_oleada** with the attribute **Status_Compra**:
  - **True**: If a purchase has been made.
  - **False**: If no purchase has been made.
- This will ensure that the purchase status is recorded for all paths in the **Decision Split**.

Third Wave: Email 3 - CTA "Call Us"

- **Email 3** will include a CTA labeled **"Call Us"**.
- Clicks on this CTA will be recorded in the **DE_CallCenter** data extension for subsequent export and use by the call center team.



From the Engagement Split of Email 2

- For those who **did not click** on **Email 2**, the flow proceeds to send **Email 3**, which includes the **CTA "Call Us"**.

Email 3 with CTA "Call Us"

- **Email 3** is sent to subscribers who haven't clicked on Email 2.
- The email includes a CTA labeled **"Call Us."**

Engagement Split for Clicks on Email 3

- An **Engagement Split** will monitor clicks on **Email 3**.
- If subscribers click the CTA, an **Update Contact** activity will:
    - Update the **DE_CallCenter** data extension with a static attribute:
        - **Status_CTA** = **True** for those who clicked.

Export to FTP

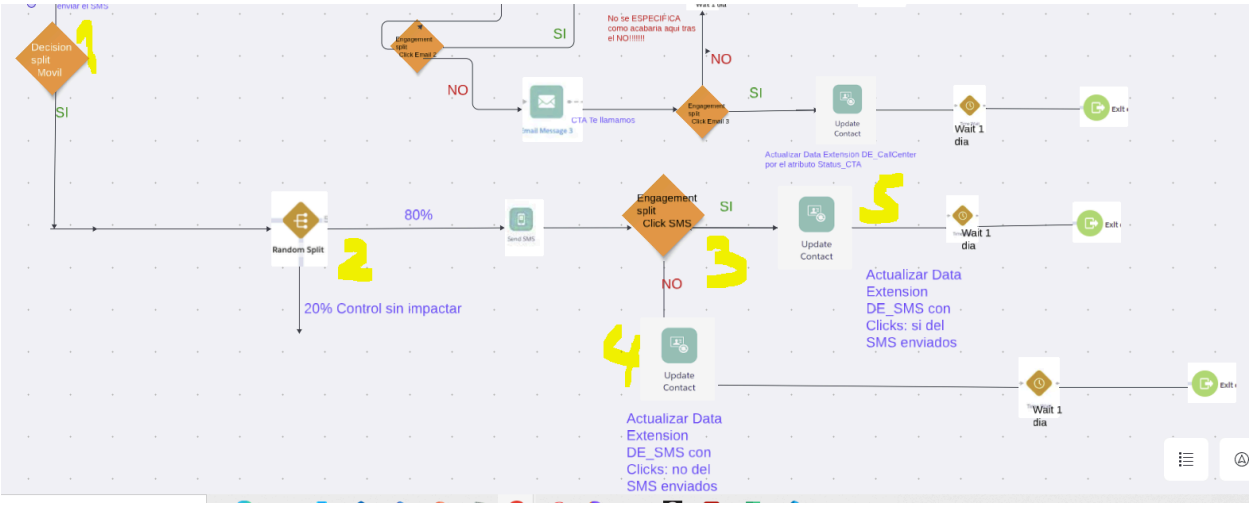
- The **DE_CallCenter** data extension, containing the customers who clicked the **CTA "Call Us"**, must be exported to the **FTP**.
- This file will be provided to the **Customer Service** team so they can contact the customers as soon as possible.

Alternative: CRM Integration

- If there is a CRM connected to **Marketing Cloud** (e.g., Service Cloud), it would even be possible to:
    - **Create a Task** in the CRM, triggering an alert for the Customer Service team.

Path for Subscribers Without Email but With Mobile Number

- Subscribers who do not have an email address but have a mobile number will be routed through a separate branch.
- This path will handle communication via **SMS**, ensuring these customers are still engaged with the campaign.



From the Engagement Split of Email 2

- For those who **did not click** on **Email 2**, the flow proceeds to send **Email 3**, which includes the **CTA "Call Us"**.

Email 3 with CTA "Call Us"

- **Email 3** is sent to subscribers who haven't clicked on Email 2.
- The email includes a CTA labeled **"Call Us."**

Engagement Split for Clicks on Email 3

- An **Engagement Split** will monitor clicks on **Email 3**.
- If subscribers click the CTA, an **Update Contact** activity will:
    - Update the **DE_CallCenter** data extension with a static attribute:
        - **Status_CTA** = **True** for those who clicked.

Export to FTP

- The **DE_CallCenter** data extension, containing the customers who clicked the **CTA "Call Us"**, must be exported to the **FTP**.
- This file will be provided to the **Customer Service** team so they can contact the customers as soon as possible.

Alternative: CRM Integration

- If there is a CRM connected to **Marketing Cloud** (e.g., Service Cloud), it would even be possible to:
    - **Create a Task** in the CRM, triggering an alert for the Customer Service team.

Path for Subscribers Without Email but With Mobile Number

- Subscribers who do not have an email address but have a mobile number will be routed through a separate branch.
- This path will handle communication via **SMS**, ensuring these customers are still engaged with the campaign.

From the Initial Decision Split: EMAIL IS NOT NULL

- For those who **do not have an email address**, the flow will branch off to check if they have a **mobile number** using another **Decision Split**.

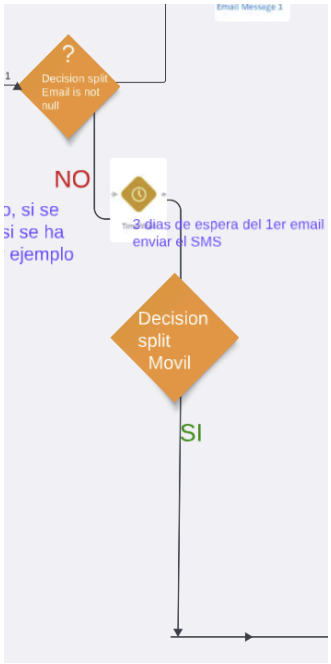
Decision Split: MOBILE IS NOT NULL

- Subscribers without an email will be filtered to determine if they have a valid mobile number.
- Those with a mobile number will proceed to receive an SMS communication.

Wait Before SMS Sending

- A **3-day Wait** will be added before sending the SMS. This allows sufficient time to analyze the data and ensure only the correct subscribers receive the SMS.

This ensures a smooth transition for subscribers who lack an email address while leveraging SMS as an alternative channel.



Wait Before Each Exit

- Add a **1-day Wait** before each **Exit** to allow for final data processing and ensure smooth termination of the journey for each subscriber.

Decision Split: MOBILE IS NOT NULL

- For subscribers with a mobile number, they will proceed to a **Random Split** for group segmentation.

Random Split: Control Group and SMS Sending

1. **20% Control Group**

- Assign 20% of the subscribers to a control group. These subscribers will not receive the SMS.

2. **80% Remaining Subscribers**
   - The remaining 80% will proceed to receive the SMS.

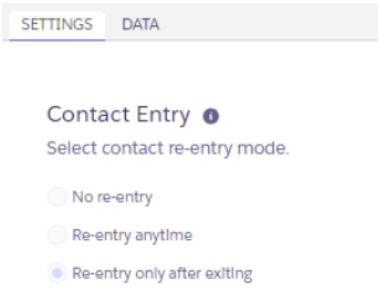
Engagement Split: Clicks on SMS

- An **Engagement Split** will monitor clicks on the SMS.
- Based on the result:
  - Update the **DE_SMS** data extension with the attribute **Clicks_SMS** set to:
    - **"Yes"** for those who clicked the SMS.
    - **"No"** for those who did not click.

Journey Settings

The task does not specify the required **Journey Settings**, but it is essential to configure them before activating the journey.

- **Suggested Setting**:
  - **Re-entry only after exiting**: This ensures that subscribers can only re-enter the journey once they have completed or exited it.



Automation Studio

Automation 1: Segmenting Completed Purchases per Wave

1. **SQL Activity**
   - Create an **SQL Activity** to segment customers who have completed purchases for each wave based on the attribute **Status_Compra = "SI"**.

Automation Workflow

- This automation can run **daily** to ensure the data is refreshed and synchronized with the journey.

```
SELECT
    CustomerID,
    SubscriberKey,
    status_Compra
FROM
    DE_Compras_1eraoleada
WHERE
    status_Compra = 'True'
```

Automation 2

Automation Studio: Data Extract and Export to FTP

To automate the export of purchases for each wave to the FTP server, follow these steps:

Step 1: Data Extract Activity

- Use the **Data Extensions Extract** type in Automation Studio to export data from the following data extensions:
    - **DE_Compras_1era_oleada**
    - **DE_Compras_2a_oleada**
    - **DE_Compras_3era_oleada**
    - **DE_CallCenter**
- Configure the Data Extract Activity to generate export files for each data extension and store them in the **Safehouse**.

Step 2: File Transfer Activity

- Add a **File Transfer Activity** immediately after the Data Extract.
- Configure the File Transfer with the option to **Move from Safehouse to FTP**.
    - Specify the target FTP directory for the files.

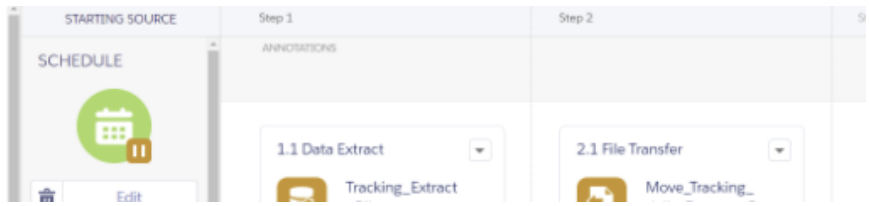
Automation Scheduling

- Schedule the automation to run **daily at 11:00 PM**. This ensures that updated purchase data from all waves is exported consistently and available on the FTP server.

Automation Workflow Summary

1. **Data Extract**: Extracts data from the data extensions for purchases by wave.
2. **File Transfer**: Moves the extracted files from Safehouse to the FTP server.
3. **Schedule**: Set to run daily at 23:00.

This setup ensures an efficient and automated process for keeping purchase data updated and accessible for further processing or analysis.



Automation 3

Automation Studio: Tracking Extract for Sends, Opens, and Clicks

This automation extracts tracking data (sends, opens, and clicks) for the last 30 days using a **Tracking Extract** and exports it to the FTP server daily.

Why Tracking Extract Instead of Data Views?

- Since the journey is new and the campaign has just started, using **Tracking Extract** ensures you can retrieve sufficient data for analysis, even when the **Data Views** have limited initial records.
- The focus is on **SENT/ENVIADAS**, **Opens**, and **Clicks** from the **previous day**.

Automation Workflow

Step 1: Data Extract Activity (Tracking Extract)

- Use the **Tracking Extract** type in Automation Studio to extract tracking data for:
  - Sends
  - Opens
  - Clicks
- Configure the Data Extract Activity:
  - Set the **Time Period** to the **last 30 days** for the initial extraction.
  - For daily exports, configure it to extract data from **the previous day**.

Step 2: File Transfer Activity

- Add a **File Transfer Activity** after the Data Extract.
- Configure the File Transfer to **Move from Safehouse to FTP**, specifying the target FTP directory.

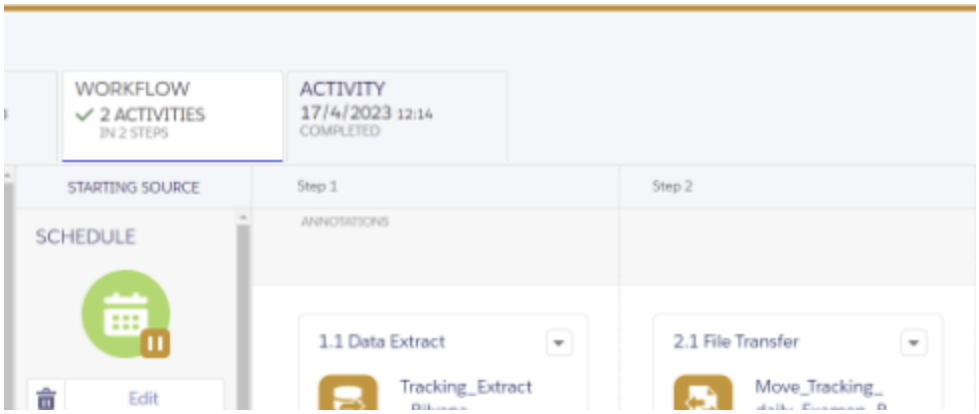
Scheduling

- **Schedule the automation** to run daily at **8:00 AM**, ensuring that tracking data for the previous day is consistently exported.

Automation Workflow Summary

1. **Data Extract (Tracking Extract)**: Retrieves tracking data (sends, opens, clicks) for the required period.
2. **File Transfer**: Moves the extracted files from Safehouse to the FTP server.
3. **Schedule**: Runs daily at **8:00 AM**.

This setup ensures that tracking data is available daily for analysis and reporting.



Using Data Views with SQL Activity

If using **Data Views**, create an **SQL Activity** to perform a **LEFT JOIN** between the data extension and relevant Data Views for tracking (_Bounce, _Sent, _Click, _Open), joined by the **SubscriberKey**.

Steps for SQL Activity

1. **Data Views to Include**:
    - **_Sent**: Records of emails sent.
    - **_Open**: Records of email opens.
    - **_Click**: Records of link clicks.
    - **_Bounce**: Records of bounces

```sql
SELECT
    de.SubscriberKey,
    de.EmailAddress,
    de.Status_Compra,
    sent.EventDate AS SentDate,
    open.EventDate AS OpenDate,
    click.EventDate AS ClickDate,
    bounce.BounceCategory AS BounceReason
FROM DE_Clientes AS de
LEFT JOIN _Sent AS sent
    ON de.SubscriberKey = sent.SubscriberKey
LEFT JOIN _Open AS open
    ON de.SubscriberKey = open.SubscriberKey
LEFT JOIN _Click AS click
    ON de.SubscriberKey = click.SubscriberKey
LEFT JOIN _Bounce AS bounce
    ON de.SubscriberKey = bounce.SubscriberKey
WHERE sent.EventDate >= DATEADD(DAY, -30, GETDATE()) -- Last 30 days
```

Automation 4

Importing from FTP to SFMC Using Automation Studio

If data needs to be imported from an FTP server into Salesforce Marketing Cloud (SFMC), the process will be handled using an **Import Activity** with a **FileDrop** as the trigger. For compressed files in `.zip` format, this will require two steps:

Step 1: File Transfer Activity

1. **Purpose**: Move the file from FTP to the **Safehouse** and handle decompression and decryption if necessary.
2. **Configuration**:
    - Use the **Manage File** feature in the **File Transfer Activity**.
    - Configure the activity to:
        - **Unzip** the file.
        - **Decrypt** the file (if applicable).
    - Ensure the file is stored in the **Safehouse** directory for further processing.

Step 2: Import Activity

1. **Purpose**: Import the decompressed file from the Safehouse into a specified Data Extension in SFMC.

2. **Configuration**:
    - Set the **Import Activity** to:
        - Source the file from the **Safehouse** directory.
        - Map the file columns to the corresponding fields in the Data Extension.
    - Use **FileDrop Automation** as the entry event to trigger the Import Activity whenever a new file arrives in the specified FTP folder.

Automation Workflow

1. **File Transfer Activity**: Handles the file transfer, decompression, and decryption from FTP to Safehouse.
2. **Import Activity**: Processes the file from Safehouse into SFMC.
3. **Automation Schedule**: Configure FileDrop Automation to monitor the FTP folder for new files and trigger the workflow automatically.

Notes

- Make sure the file naming convention matches the FileDrop configuration to avoid errors.
- Validate the mapping between the file and the Data Extension fields before activation.
- Test the process with a sample file to confirm functionality.

Import Activity - From Safehouse to SFMC: Assign the target data extension and locate the file from the ExactTarget Safehouse.

If the file is in **.csv format**, only one step is required: the **Import Activity**.

Task 3: Use Case - Automation for Exclusion

Scenario:
There are 4 planned email campaigns for the day. The **priority campaign** is **DE_MASTER_CLIENTES**.
Since a single client may appear in multiple Data Extensions (DEs), as a policy, no client should receive more than one email per day.

Primary Key:

- `CO_CLIENTE` (Client ID)

Data Extensions:

- **DE_MASTER_CLIENTES** (priority campaign)
- **DE_HOGAR**
- **DE_SALUD**
- **DE_AUTO**

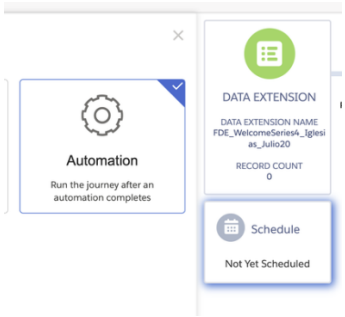
Option 1 - SQL Query in Automation Studio

Segment the emails to ensure that the criteria of not sending more than one email per day to a customer are respected, using the **SubscriberKeys** from **DE_MASTER_CLIENTES**.

```
    SELECT
    m.CorreoElectronico,
    m.Móvil,
    m.CO_CLIENT,
    m.Nombre,
    m.Apellido
FROM
    DE_MASTER_CLIENTES m
LEFT JOIN
    DE_HOME h ON m.SubscriberKey = h.SubscriberKey
LEFT JOIN
    DE_SALUD he ON m.SubscriberKey = he.SubscriberKey
LEFT JOIN
    DE_AUTO a ON m.SubscriberKey = a.SubscriberKey
LEFT JOIN
    DE_Exclusions_Today e ON m.SubscriberKey = e.SubscriberKey AND e.SendDate = CONVERT(DATE, GETDATE())
WHERE
    h.SubscriberKey IS NULL
    AND he.SubscriberKey IS NULL
    AND a.SubscriberKey IS NULL
    AND e.SubscriberKey IS NULL
    AND m.CorreoElectronico IS NOT NULL
    AND m.Móvil IS NOT NULL

Esta consulta sirve para seleccionar los clientes de la tabla DE_MASTER_CLIENTES que no están presentes
 en ninguna de las tablas DE_HOME, DE_SALUD, DE_AUTO y DE_Exclusions_Today con la fecha actual.
Esto asegura que los clientes no reciban más de un correo electrónico por día
y solo selecciona aquellos con direcciones de correo electrónico y números de móvil válidos.
La tabla la tengo que crear adicionalmente DE_Exclusions_Today.
Contiene clientes que ya recibieron un correo electrónico hoy CON 2 filas de atributos:
 CO_CLIENT text(255),
    SendDate DATE.
Si el cliente está en esta tabla con la fecha actual, no será incluido en el resultado.
```

The result of this query would be a new Data Extension generated in Automation Studio, which will serve as the **Entry Source** for the email send.
In Automation Studio, in **Step 1**, this SQL Query would be executed, and in **Step 2**, either an Email Activity would be triggered directly, or the Data Extension would be scheduled to enter Journey Builder as an entry source via a **Scheduled Automation**.



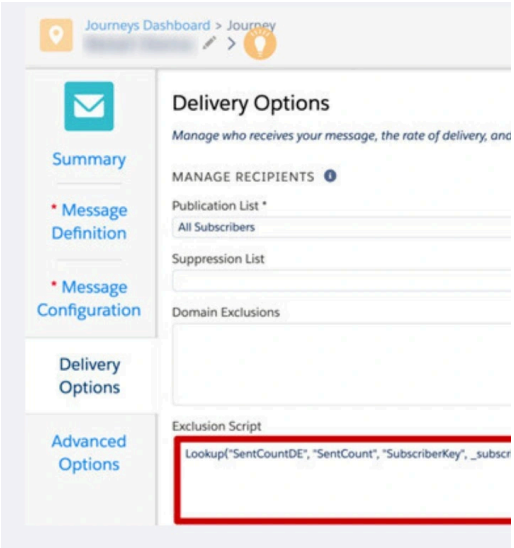
Option 2 as a Solution

In **Journey Builder**, I go to the journey for the priority campaign using the **DE_MASTER_CLIENTES** data extension.

I open the **Email Send Activity** in this journey, navigate to the **Delivery Options** tab on the left, and locate the **Exclusion Script** section.

Steps:

1. **Open the Email Send Activity**
   - Navigate to the **Email Send Activity** in the journey.
2. **Go to the Delivery Options Tab**
   - On the left-hand side, select the **Delivery Options** tab.
3. **Locate the Exclusion Script Section**
   - Within the **Delivery Options**, find the **Exclusion Script** field.
4. **Configure the Exclusion Script**
   - Add a script to ensure subscribers who have already received an email that day are excluded from this send.



RowCount(LookupRows("DE_HOGAR", "Email_Field", emailaddr)) > 0)
OR
RowCount(LookupRows("DE_SALUD", "EmailAddress", emailaddr)) > 0)

RowCount(LookupRows (“Nombre de la DE” , “nombre columna de la DE”, AttributeValue(“nombre de la columna del DE del journey”)))> 0

RowCount(LookupRows (“Nombre de la DE” , “nombre columna de la DE”, AttributeValue(“CO_CLIENTE”)))> 0

Rowcount(LookupRows**("_Sent**", "SubscriberKey", SubscriberKey, "EventDate", Format(SystemDate(), "yyyy-MM-dd"))) > 0

Explanation:

- This script checks the **_Sent Data View** to see if the subscriber (`SubscriberKey`) has already received an email on the current date (`SystemDate()`).
- If the count is greater than 0, the subscriber will be excluded from this send.

This Approach

- Allows the priority campaign to enforce the one-email-per-day policy directly in Journey Builder.
- Reduces dependency on additional automation or SQL scripts.

Sources of documentation used:

https://ampscript.com/using-an-exclusion-script-for-email-sends/#google_vignette

Option 3 as a Solution for Testing

To test before implementing it in **Delivery Options**, add a block of **AMPscript** in a copy of the same email before sending it.

```
%%[
VAR @clientId, @sendDate, @isExcluded

/* Retrieve the Client ID */
SET @clientId = AttributeValue("CO_CLIENTE")

/* Format the current date to match your data format */
SET @sendDate = FormatDate(Now(), "YYYY-MM-DD")

/* Check if the client exists in the DE_Sent_Today data extension */
SET @isExcluded = RowCount(LookupRows("DE_Sent_Today", "CO_CLIENTE", @clientId,
"SendDate", @sendDate))
]%%

<!-- Display exclusion status for debugging -->
<p>Exclusion Status: %%=TreatAsContent(v(@isExcluded))=%%</p>

%%[ IF @isExcluded > 0 THEN ]%%
    <p style="color: red;">This client has already received an email today.</p>
%%[ ELSE ]%%
    <p style="color: green;">This client has not received an email today.</p>
%%[ ENDIF ]%%
```

Variable Initialization:
Ensure VAR is used for declaring variables (`@clientId`, `@sendDate`, and `@isExcluded`).

`FormatDate` Function:
Confirm that `FormatDate(Now(), "YYYY-MM-DD")` matches the format of the SendDate field in your `DE_Sent_Today` data extension.

TreatAsContent:
The line `TreatAsContent(%%=v(@isExcluded)=%%) > 0` works but lacks context. It's better to use it for debugging output or validation and ensure proper conditional logic with `IF`.

Debug Output:
Add an output like `<p>` to visually confirm the status during testing, particularly for the value of `@isExcluded`.

IT works in the following way:

`LookupRows`: Retrieves rows from `DE_Sent_Today` where `CO_CLIENTE` matches `@clientId` and `SendDate` matches `@sendDate`.

`RowCount`: Counts the rows returned. If greater than 0, the client is excluded.

Conditional Logic: Renders appropriate messages (`red` or `green`) based on the exclusion status.

Task 4: Reporting

Generate the report from **SFMKT** for **Journey 4CESSVC101_RETARGETING_DELEGACIONES**.

There are several ways to extract analytical data from a journey, such as:

---

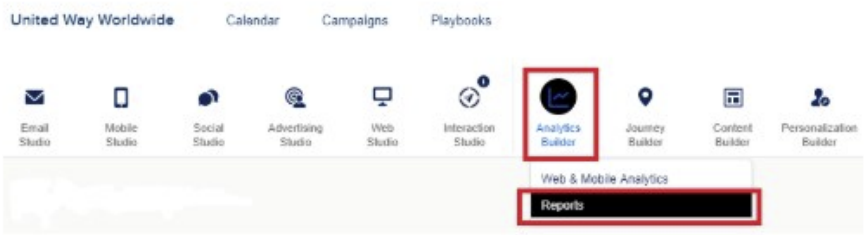
1. **Journey Analytics:**
   - Within the journey, you can access the **Analytics panel** to see how the emails and SMS sent have impacted the audience.
2. **Journey Tracking:**
   - In **Email Studio > Tracking**, you can access the tracking section, where you'll find a folder for each journey. Inside, there is a report for every email activity within the journey.
3. **Journey Report:**
   - In the **Reports Catalog**, there is a dedicated report for journeys. It provides information about the overall performance of the journey and detailed insights for each branch.
4. **Analytics Builder > Reports > Journey Builder Email Send Summary:**
   - Use this option to get a summary of the email sends in the journey.
5. **Email Studio Overview > My Tracking > Journey Builder Sends:**
   - View a summary of email performance for sends within Journey Builder.
6. **Intelligence Reports > Journey Performance by Email:**

- You can input the **Journey activity name** and receive metrics such as:
  - Sends
  - Opens
  - Bounces
  - Unique clicks
  - Click-to-open rate

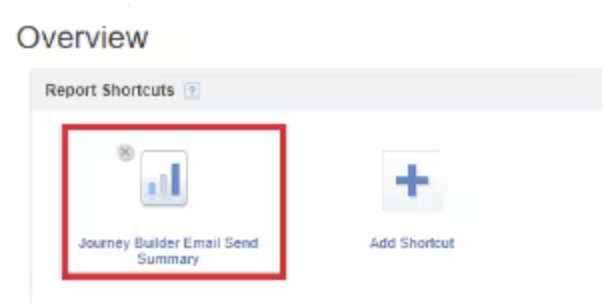
Note: This option 6 may not be available depending on the company's Marketing Cloud license.

Solution Task 4

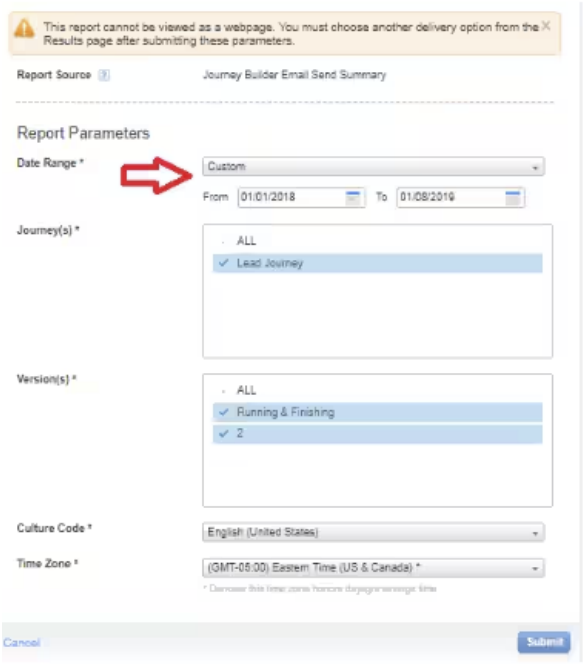
To obtain the reporting required for the task, I need to access **Analytics Builder > Reports > Journey Builder Email Send Summary**.



Mark Journey Builder Email Send Summary



In **Report Parameters**, assign the dates.
Select the **Journey 4CESSVC101_RETARGETING_DELEGACIONES**.
Click **Submit**.



I choose to have the report sent via email.

If I need this report on a recurring basis, I can schedule it to be sent automatically. However, I must first save the report to enable this option.