Email Engagement Score

We send many emails. Often times, it means that we send many emails to the same person. Until recently, it was really difficult to have the visibility on this.

The goal of the Email Engagement Score is to represent two things related to a given person :

* the volume of emails that we send to someone
* how that person interacts with those emails (open, click)

How it is computed

**Step 1 : weighted sum**

We take the sum of the following actions :

* email marked as spam : -20
* email sent : -1
* email opened : +5
* email clicked : +5

This gives a total that can be either negative or positive, depending on how many emails were sent, and how the person interacted.

**Step 2 : normalized between -1 and 1**

In order to make it easier to grasp, that total sum is then normalized between -1 and 1. This is done by using a hyperbolic tangent Tanh : <https://reference.wolfram.com/language/ref/Tanh.html>

**Step 3 : scale the Tanh to avoid too many extremes**

In order to avoid having the majority of emails at -1 and 1, we scale down the sum (divided by 50).

**Complete formula :**tanh((-countSent-20\*countSpam+5\*countOpened+5\*countClicked)/50)

Implementation

[Vim · macOS Setup Guide (sourabhbajaj.com)](https://sourabhbajaj.com/mac-setup/Vim/)

What is a data domain?

A*domain*as “a sphere of knowledge, influence, or activity.” A domain has the *knowledge*of

* what events happen,
* what data is generated
* what the rules governing the data are

It*influences*exactly what data should be known and captured during the process and how. It has a set of *activities*to perform such processes

The domains have an organizational responsibility to share their data with product-like qualities inside and outside of their domain

Each domain becomes responsible for the data it is most familiar with: the domain that is the first-class user of this data or is in control of its point of origin

Domain types

we can define 3 main type of domains:

* *Source-aligned domain data*
  + Analytical data reflecting the business facts generated by the operational systems. This is also called a *native*data product.
* *Aggregate domain data*
  + Analytical data that is an aggregate of multiple upstream domains
* *Consumer-aligned domain data*
  + Analytical data transformed to fit the needs of one or multiple specific use cases. This is also called *fit-for-purpose*domain data.

Principle of Domain Ownership

The principle of domain ownership is a key aspect of the data mesh architecture.

In this approach, data is treated as a product and is owned by the domain that produces it. This means that the domain is responsible for the quality, governance, and availability of the data it produces, and that the data is managed in a way that is tailored to the specific needs of that domain.

This allows for more efficient and effective data management, as the domain teams are best equipped to understand the needs and requirements of their particular area of the business. In a data mesh architecture, domains are self-organizing and self-governing units that produce and consume data. Each domain has its own data team, which is responsible for the data products within that domain. These teams are typically cross-functional and consist of experts in data engineering, data science, and domain-specific knowledge.

In practice, the domain ownership principle means that each domain has autonomy over its own data, but also has an obligation to ensure that the data is accessible and usable by other domains within the organization. This is achieved through the use of standard data formats and shared technologies, which allow for seamless data integration across domains.

**What is a Data Platform?**

A data platform is an integrated set of processes, technologies and people that collectively meets an organization’s end-to-end data needs. It's like a toolbox for working with data–it provides a centralized and integrated environment for sharing, managing, and analyzing data.

Frontiers Data Platform (FDP) scope is to use different technologies to provide managed and governed data on the base of the company use case. It can satisfy streaming requests through Confluent Kafka, on demand queries via GraphQL, Analytics and reporting requests thanks to the DWH (SQL Server) and Frontiers Ocean (Google BigQuery).

Why do we need a Data Platform?

To allow for fast, scalable, reliable, and well-organized data production, ingestion, publishing, and consumption (collectively, "data management"). A Data Platform is a key component to unlock the value of the flow of data within Frontiers that brings together our technologies to form a Data Mesh Operating Model and our people in a new teams topology: the Data Platform tribe.

The evolution of our Data Platform

Frontiers' data storing, sharing, and analysis requirements have grown in complexity together with the business.

Past

Diagram

Description automatically generated

Diagram

Description automatically generated

### The monolith

A common industry practice at the time, the Frontiers approach to data was monolithic: a single operational application, an operational data store (ODS), and a data warehouse that synchronized and processed the operational data.

### Microservices

As the business evolved and new teams were introduced, Frontiers embraced new technologies and practices being developed at that time. From the monolith to microservices, each served the operational and data-storing requirements of a single business domain.

This new approach brought the benefits of segregation and team ownership but also the challenge of having to share, synchronize, and analyze data from a diverse set of sources.

### Frontiers Graph

To meet data-sharing requirements, Frontiers implemented an integration layer that aggregated data from the microservices, across the different tenants. and made this data available for querying through GraphQL, or as a stream of data changes, through Confluent Kafka.

### Frontiers Ocean

For efficient analytics, to enable better insight into internal and external data, and to allow processes to consume these insights, Frontiers introduced Ocean.

## The problems that we want to solve

### Faster resolution of business problems

Solving a business problem using software means creating an automatic process and a data model that formalize a real-world business procedure. Often, integration between heterogenous and distributed systems is a key factor in the solutions. Integration is enabled by shared and well-known read data models.

We want to be better at solving business problems with software in the fastest possible way.

We will reach this goal by designing and realizing a governed Data Platform that allows for an easy and fast way to create, publish, share and discover data, to enable processes and users to take data-driven actionable decisions.

### Tighter integration with the analytical platform

Since we are a data-driven company, we aim to make high-level business decisions based on data. For that reason, the Data Platform will empower Frontiers' analytical tools to seamlessly integrate with the operational platform by becoming consumers of the data the operational platform produces.

### A more cohesive group of people

We want to push boundaries around the concept of teams and create a cohesive group of software engineers and managers that collaborate to the development and success of the Data Platform, providing and sharing expertise, allowing for a cross-pollination of skills and diversifying competencies according to personal development goals.

### Provide capacity and knowledge to operational teams

We want to enable operational teams to easily and swiftly publish their data into the Data Platform, and data analysis teams to consume the published data, by providing guidance, tools, guidelines, secure roles and, where necessary, capacity.

# Data Platform examples

For more information, access our [Data Platform examples](https://confluence.frontiersin.net/pages/viewpage.action?pageId=151127731) page.

# Data Mesh Operating Model

The ever-evolving approaches to data management bring inherent complexities with them, above all the need for governance and scale. A modern paradigm, that tries to tackle these problems by embracing the ubiquity of the data in a complex business, is the Data Mesh.

DBT SET UP

**WSL and virtual environment**

WSL

For windows users, install WSL ([Windows subsystem for Linux](https://docs.microsoft.com/en-us/windows/wsl/install)). Else, go to next step

* Run the powershell as administrator and write

$wsl --install

* + Troubleshooting for WSL.1: In case you write sudo apt-get update and notice there is no connection to the internet
    - Modify hte resolv.conf file
* sudo nano/etc/resolv.conf
  + - Inside it will be written *nameserver x.x.x.x* It has to be renamed to *nameserver 8.8.8.8*
    - Save changes
    - *block* that file

sudo chattr -f +i /etc/resolv.conf

* Troubleshooting for WSL.2:
  + - Run powershell as administrator
      * wsl.exe --update
      * wsl –shutdown (it forces reboot)
    - Reboot the system
    - Once rebooted, the ubuntu shell will prompt asking for a new user and password. If not prompted => <https://docs.microsoft.com/en-us/windows/wsl/setup/environment#set-up-your-linux-username-and-password>
* *Configure Linux terminal to work with VSCode*
  + [*https://docs.microsoft.com/en-us/windows/wsl/tutorials/wsl-vscode*](https://docs.microsoft.com/en-us/windows/wsl/tutorials/wsl-vscode)
  + <https://docs.microsoft.com/en-us/windows/wsl/setup/environment>
* Now that the linux terminal and VS code are installed, open VSCode and click on the green icon

A screenshot of a computer

Description automatically generated with medium confidence

## Virtual Environment & Google Cloud

### Venv

* For Windows users:
  + Open powershell
  + write

wsl

Now we are able to use bash

* Make sure you have a Python version 3.8 / higher
* Navigate to your main linux folder

cd /home/<your user>/

* create a folder for repositories, grant the right permissions and navigate to it

mkdir ubuntu\_data

chown –R <your user>:ubuntu\_data/

cd ubuntu\_data

We will create a virtual environment for our DBT project with **venv**using the following command:

pythonX -m venv <name of your env>

X = 3.8 or higher. The name of your environment without spaces. Ex: python3.8 -m venv dbt\_env.

With this we will create a virtual environment with Python 3.8.

* Venv:
  + to activate it, we have to navigate to the folder in which it was created

cd /home/<your user>/<name of your env>

source <name of your env>/bin/activate

* + to deactivate it:

bash

IF you did it, great. Else:

* [Official documentation](https://docs.python.org/3/library/venv.html#venv-def)
* [Nice youtube video](https://www.youtube.com/watch?v=-TZfH7r33CQ)

### GCloud CLI

We need it to work with Google Cloud products (BigQuery, bucket...) locally

* Double-check with your manager/workmates if you have access for the BigQuery projects you'll have to work with.
* Configure gcloud CLI (in this case, using WSL terminal)
  + The ubuntu SO supports all you need for the link below, so install the first option always of this link => <https://cloud.google.com/sdk/docs/install#deb>
  + Following the steps it will take you to this link => <https://cloud.google.com/sdk/auth_success>

# ****The repository****

* Clone [the respository](https://devops-server.frontiersin.net/Madrid/Architecture/_git/Frontiers.Ocean.Researcher.Profiles)

Graphical user interface, application, Teams

Description automatically generated

To maintain coherence it is really important you locate all your repositories in the same place. Probably the best one is the ubuntu\_data folder previously created

* Navigate to that folder

git clone <the https url>

* + Troubleshooting: If you can't maybe is that you need writing permisions on that folder. That can be changed as follows:
    - Navigate above the folder. In the terminal:

ls

Chown –R <linux\_user>:<linux\_user data\_folder/> (example: vargas:vargas ubuntu\_data/)

* + Introduce your .net user and password
  + To don't be asked every time for this user and password:
    - Navigate to the repository folder

git config credential.helper store

**Working with Visual Studio Code**

* Check you are using the right Python interpreter

Now that you have the repository locally:

* + Activate virtual environment
  + Navigate to the repository

code .

This command will open Visual Studio Code at the level of the repository. Make sure the Python interpreter in VScode is the same than in your Virtual Environment.

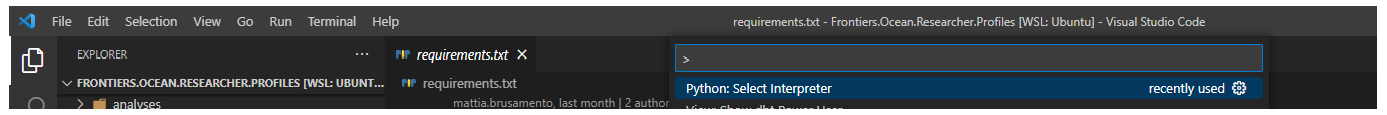
* + Open WSL, activate your venv, write in the terminal

which python

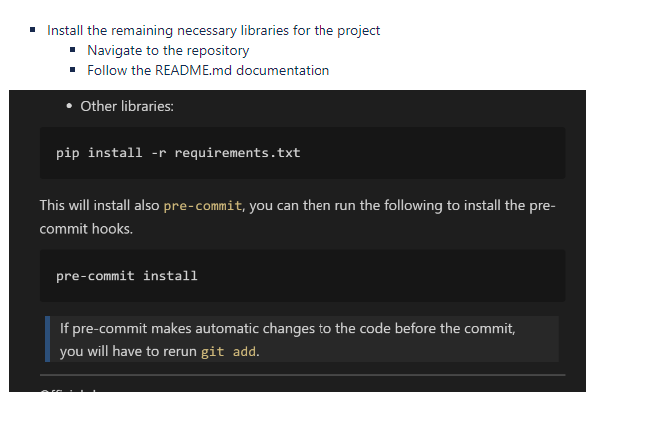
It will gives you the path of the Python used in that venv



* Go to VSCode and press cntrl + shift + P. Then select Python interpreter.



* Enter the pah of your Venv



Graphical user interface, text

Description automatically generated

Graphical user interface, application

Description automatically generated



POC

oVERVIEW

Objective

Poc Data scope

Poc Steps

##### ETL Tool Evaluation

AirByte as a data transfer tool was evaluated using a number of criteria (See)

| **Pros** | **Cons** | **Notes** |
| --- | --- | --- |
| Tool currently used for BQ transfer on Data Platform (In-House) | Tool has limitations in the ability to extract very wide data tables, since data transfer is limited to 10,000 rows and xxx MB |  |
| Simple, Easy to use and Configure | Lack of scheduling capabilities | Airflow is being used |
| Parallelisation can be configured |  |  |
| Open-Source (No Cost) | Open-Source not usually a robust ETL tool \*\*\* |  |

Given current data vault data storage strategy facilitates parallelisation, and parallelisation is what is used in the current SQL data transfer

* The use of parallelisation was evaluated with AirByte, and this minimised the core limitations of the tool

To maintain the use of the data platform tool architecture(in-house), given other tools may equally struggle with extremely wide tables.

* AirByte is deemed to be sufficient as a data ingestion tool for this requirement

##### ETL Strategy Evaluation

Based on the initial scope of the PoC, an initial ETL assessment was carried out, and the load duration where highly inadequate due to the limitations of the tool and the unusual wide nature of teh source tables

As a result a change was made to evaluate the ETL of the data vault model since this model provides for the load of vertical partitions of data which can be handled by AirByte tool.

Option 1: Full load of Denormalised Reporting Schema

* Performance limitations on Airbyte for transfer size

**Option 2:  Load of DataVault Model (Vertical Partitions of Data Domains)**

* + **All dimensional entities are loaded in parallel batches to optimise performance**
    - **These are full load due to the nature of aggregated attributes in the model**
  + **Incremental load of fact table**
  + **Full load of fact tables applied outside business hours**

Considerations

* RowVersion datatype not accepted by AirByte, a workaround needs to be implemented to enable Incremental load

Risks

* AirByte is not as performant at the load of the full fact dataset of 40million rows ~ 01:37:00 (SQL = 00:20:00)
* Current business data issues, updates are applied on historic data from time to time and the structure of the table changes.
* Full load may be required as a result, and scalability concerns may arise since the batch window for weekends need to be planned

##### Data Transformation

Considerations

* DataType must match expected Tableau data types
* Attribute naming on Tableau may need to change to avoid special characters