THOMAS MORE

Lead Pipeline Traceability

Realization document

Ravin Shalmashi
Student Bachelor Applied Computer Science

2025 - 2026

Table of Contents

1. Preface

This realization document presents the work carried out during my internship at KBC Bank & Insurance, within the Leads Capabilities (LeCa) team, a team responsible for managing and optimizing the marketing lead generation pipelines used across KBC's digital and direct marketing channels.

The focus of this internship was the analysis and improvement of data traceability within the LEDAS marketing pipelines. These pipelines are responsible for determining which clients receive communication in marketing campaigns, and for ensuring that all decisions made during this process are logged in a standardized and auditable way.

The central problem addressed during this internship was the existence of unexplained discrepancies between pipeline outputs and the Data Traceability (DT) dataset. These discrepancies made it difficult to understand why certain clients were included in or excluded from campaigns, reducing the reliability of reporting and increasing the complexity of debugging.

This document describes the practical execution of the project in full detail, including the technical context, the methodology developed, all findings uncovered during the investigation, and the solutions applied. Particular emphasis is placed on identifying the root causes of discrepancies and documenting a reusable debugging approach that can support the team in future investigations.

I would like to express my sincere gratitude to my internship coach and all team members of the LeCa team at KBC for their continuous guidance, patience, and support throughout this internship. Their willingness to explain the technical context and help interpret complex pipeline behaviors was essential to the success of this project.

2. Summary

During this internship, I investigated and resolved inconsistencies in the marketing pipelines at KBC Bank & Insurance. The pipelines are responsible for processing client leads and determining who receives marketing communications. Inconsistencies had been observed between the pipeline's output stages and the Data Traceability (DT) dataset, which is the authoritative record of which leads were processed and why they were filtered.

The main symptoms of the problem were:

- **Missing leads**: leads that existed in one stage were absent in the next without a traceable reason
- **Duplicate records**: the same lead appeared multiple times, inflating row counts
- **Unexplained discrepancies**: differences between stages could not be explained using existing documentation or dashboard data

To address these issues, I designed and implemented a structured data reconciliation methodology using PySpark notebooks. The methodology tracked data flow across the main pipeline stages:

- Notebook output (NB)
- Fixed Selection Retrieval (FSR) — going from targeted client to communicated client
- Offer Selection (OS) — initial lead generation.
- Offer Enrichment (OE) — adding business and client attributes.
- Buffer Update (BU) — manages leads that were not processed during the current execution and determines whether they remain eligible for communication. Buffer Selection (BS) makes sure client only receives the lead with the highest lead value

Leads selected for communication are written to the Lead Value dataset, while filtered leads are written to the Data Traceability dataset together with the corresponding filter-out reason.

The following key findings were identified during the investigation:

- Duplication in the Offer Enrichment stage caused by an incorrect union between parent and child records in the GDPR-related join logic within leads_common/transformations/filters.
- Missing buffer traceability: filter-outs applied during buffer processing were not included in the DT dataset, causing large unexplained discrepancies.
- Blocking table windowing issue: incorrect windowing logic led to reading outdated blocking entries, resulting in small but consistent inconsistencies.
- Duplication in the buffer update pipeline caused by dropping essential columns during the select_relevant_enrichments transformation, causing records to lose their uniqueness.

The outcome of this project is a fully documented debugging methodology, a clear technical understanding of pipeline behavior at each stage, and identification and resolution of multiple root causes contributing to data inconsistencies. The findings also provide a foundation for future improvements to the traceability framework.

3. Introduction

3.1. Context

KBC Bank & Insurance uses a sophisticated marketing pipeline system to determine which clients receive targeted communications. These pipelines process large volumes of customer data and apply a series of business rules, filters, and enrichments to identify eligible leads for various marketing campaigns.

At the heart of this system is the LEDAS (Lead Data Selection) platform, which orchestrates multiple pipeline stages including Fixed Selection Retrieval, Configurable Exclusions, Offer Selection, Offer Enrichment, Buffer Update, and Buffer Selection. Each stage transforms and filters data according to specific business and regulatory rules.

A critical component of this system is the Data Traceability (DT) dataset. This dataset is designed to record every lead that is filtered out at any point in the pipeline, along with a standardized reason for the exclusion. The DT dataset is used for reporting, compliance, and debugging purposes, and its accuracy is essential to the reliability of the entire pipeline.

The pipeline operates across two main frameworks: LEDAS and Level 5, each handling different types of marketing solutions. Pipelines run as either recurring scheduled executions or one-off ad-hoc runs and are orchestrated through Apache Airflow DAGs.

3.2. Problem Statement

In practice, significant inconsistencies were observed between the outputs of different pipeline stages. Specifically:

- The number of rows in the Offer Enrichment (OE) stage was higher than expected compared to the Offer Selection (OS) stage, indicating duplications.
- The sum of rows in the Lead Value (LV) stage and the Data Traceability (DT) stage did not equal the number of rows at the start of the pipeline, meaning some leads were unaccounted for.
- Certain filter-outs were not being captured in the DT dataset, making it impossible to explain where leads had disappeared.
- Small but consistent row count differences existed that could not be attributed to known business logic.

These inconsistencies reduced trust in reporting, complicated debugging efforts, and made it difficult for business teams to understand campaign outcomes. The core challenge was that the pipeline was complex, distributed across many components, and lacked end-to-end visibility.

3.3. Objective

The objective of this project was to:

- Systematically identify where leads are lost or duplicated across the pipeline stages.
- Understand the root causes of each discrepancy found.
- Develop a reusable debugging methodology for the team to apply in future investigations.
- Apply fixes where possible and validate those fixes against multiple marketing solutions.
- Document all findings in a way that improves the overall maintainability and transparency of the pipeline.
-

3.4. Scope

The analysis focused on the pipeline, specifically the following stages:

Metric	Description / Status
Fixed Selection Retrieval (FSR)	Looping through notebook output and going from targeted client (passed in notebook) to communicated client (who we actually send the communication to)
Offer Selection (OS)	Initial stage where leads are generated based on product eligibility rules. Defines the starting population for each marketing solution.
Offer Enrichment (OE)	Enriches leads with additional client and product attributes. Applies GDPR-related logic and joins with reference datasets.
Buffer Update (BU)	Buffer Update (BU) manages leads stored in the buffer and determines whether they remain eligible for future communication based on configurable retention rules.
Buffer Selection (BS)	applies channel limitations and ensures that only the highest-value lead is selected within a customer journey.

The work was performed using PySpark notebooks for data analysis, Airflow logs for pipeline monitoring, Qlik dashboards for validation, and Bitbucket for source code analysis. The scope excluded pipeline stages handled exclusively by Adobe Campaign and did not include one-off pipeline configurations unless specifically relevant to a finding.

4. Approach

4.1. Overview

The approach was based on systematic data reconciliation across pipeline stages. Rather than investigating individual components in isolation, the methodology tracked the complete flow of records from entry to exit at each stage. This made it possible to pinpoint where discrepancies originated and to quantify their magnitude.
The guiding principle was that the following equation should always hold true:

$$OE = LV + DT$$

That is, every lead in Offer Enrichment must either continue to Lead Value or be recorded in Data Traceability with a filter reason. Any deviation from this equation indicates a data quality issue.

4.2. Methodology

The investigation follows a structured process:

This methodology was applied consistently across all investigated marketing solutions and formed the basis of the debugging framework developed during the internship.

4.2.1. Step 1: Count Comparison

The first step was to establish baseline row counts at each pipeline stage. Using PySpark notebooks, I extracted the number of records at each layer and file for the marketing solutions and multiple pipeline runs that were marked as problematic in the Qlik dashboard .This immediately surfaced the scale of discrepancies and helped prioritize pipeline stages to investigate first.

Example of the excels that I used to check the values from PySpark:

kbcMS_348293

					Number of Channels	6							
		Actual leads filtered out			nb_b	2	0	nb_e	96	0			
offerId	-7,14E+08	QLIK	Notebook		fsr_b	2	0	fsr_e	96	0			
PDATE	20260502	361	313	48	ce_b	12	0	ce_e	576	0			
LVL5	J0265	Filtered out			os_b	12	0	os_e	576	-48			
		QLIK	Notebook										
		397	397	0	oe_b	12	0	oe_e	528	84			
		DIFFERENCE											
		QLIK	Notebook		lv_b	5		lv_e	222		bu_u_o	141	-86
		-36	-84		dt_b	7		dt_e	390		bu_u_dt	397	0
					bu_u_t	151							

Example of Qlik dashboard view to see marketing solutions with correct Traceability or with problems

Marketing Solution Internal Name	Actual Leads Filtered Out	Filtered Out	Data Traceability Explainable	Difference
Totals	**59,617,304**	**58,514,688**	-	-
kbcMS_346781	[redacted]	[redacted]	✓	0
kbcMS_347665	[redacted]	[redacted]	✗	20411
kbcMS_349054	[redacted]	[redacted]	✗	1041
kbcMS_6836	[redacted]	[redacted]	✓	0
kbcMS_349532	[redacted]	[redacted]	✗	1
kbcMS_348131	[redacted]	[redacted]	✗	998
kbcMS_348503	[redacted]	[redacted]	✗	1031
kbcMS_346883	[redacted]	[redacted]	✓	0
kbcMS_347658	[redacted]	[redacted]	✗	19
kbcMS_347485	[redacted]	[redacted]	✗	890
kbcMS_348106	[redacted]	[redacted]	✗	52

The differences could be positive (Duplication) or negative (missing) and its checked based on what we have in Data traceability and the difference between all of the leads and the ones in Lead value so we see how many records should be present in Data Traceability and how many were actually recorded.

The count comparison also served as a regression test: after applying fixes, counts were re-measured to verify that the expected equation was satisfied.

4.2.2. Step 2: Data Splitting

Not all records flow through to LV. some are intentionally filtered. The data-splitting step separated records into two categories for each stage transition:

Leads that continue records that pass through the filter and appear in the Lead value.

Leads that are filtered: records removed at the current stage, which should appear in DT with a reason.

Once a discrepancy was identified, the data was split into two groups:

Leads that continued through the pipeline (present in LV, FLT_OUT_IC = 0)
Leads that were filtered out (present in DT, FLT_OUT_IC = 1)
The key fields inspected at each stage were:

```
FLT_OUT_IC  – filter-out indicator (0 = active, 1 = filtered)
FLT_OUT_RSN – filter-out reason (standardized string)
BUF_STS     – buffer status
BUF_STS_CMT – buffer status comment
MKT_SLT_ID  – marketing solution identifier
CPN_NO      – campaign number
TGT_PTY_GUID_NO – target party GUID
CMN_PTY_GUID_NO – communicated party GUID
```

By computing both groups, it became possible to verify whether the filtered leads were correctly captured in DT. Cases where filtered leads were absent from DT revealed traceability gaps.

4.2.3. Step 3: Difference Analysis

With counts and splits established, the difference analysis step identified the specific nature of discrepancies:

Missing records: leads present in an earlier stage but absent in both LV and DT

Duplicate records: leads appearing more than once at a given stage, inflating counts

Unexpected increases: situations where OE contained more records than OS, indicating unwanted record multiplication

Difference analysis was performed by joining stage datasets on unique business keys and examining unmatched records. This required careful understanding of what constitutes a unique key at each stage.

4.2.4. Step 4: Root Cause Investigation

For each identified discrepancy, a root cause investigation was conducted. This involved:

Tracing filter-out reasons through the pipeline source code on Bitbucket

Analyzing join logic, especially for GDPR-related parent-child relationships

Examining column selection and transformation steps that could affect record uniqueness

Running pipeline stages locally with controlled test datasets to reproduce and confirm the behavior

Reviewing Airflow execution logs to check for unexpected re-runs or partial failures

4.2.5. Step 5: Validation

All findings and fixes were validated through multiple independent methods:

Re-running the affected pipeline stages and verifying that count equations were satisfied

Comparing results across multiple marketing solutions to confirm the fix was general, not specific to one case

Cross-referencing with Qlik dashboards to verify that business-facing reports reflected the corrected data

Peer review with LeCa team members to confirm that the root cause interpretation and fix were correct

4.3. Tools Used

Tool	Purpose
PySpark (Jupyter Notebook)	Primary data analysis environment for reading, transforming, comparing pipeline datasets
Apache Airflow	Pipeline monitoring, log inspection, tracking read/write operations per stage
Qlik Dashboards	Validation of findings against business-facing reporting
Bitbucket	Source code analysis, tracing filter reasons, understanding pipeline logic
AWS Glue (PySpark)	Underlying execution environment for pipeline jobs
Adobe Campaign	Downstream consumer of pipeline outputs, source of campaign configuration

4.4. Debugging Strategy

Debugging distributed Spark pipelines requires a different approach compared to traditional application debugging. Key strategies employed during the investigation included:

Incremental tracing: following data record-by-record through transformations rather than comparing only aggregate counts.

Controlled reproduction: running pipeline stages locally with small, known datasets to reproduce and confirm bugs.

Assumption validation: explicitly testing every assumption about data structure, key fields, and uniqueness before using it as a basis for analysis.

Intermediate checkpointing persisting intermediate datasets to Parquet during investigation to avoid re-computation and enable comparison.

Multi-solution testing: verifying findings across multiple marketing solutions to distinguish general bugs from solution-specific behavior.

5. Technical Background

5.1. Pipeline Architecture

The LEDAS pipeline consists of several sequential processing stages. Understanding the purpose of each stage is essential for interpreting the findings.

5.1.1. Fixed Selection Retrieval

This is the entry point of the pipeline. The notebook reads business selections, customer data, marketing datasets, and campaign configuration from multiple sources. It applies data cleaning, attribute mapping, and filtering to produce an initial set of potential leads. The output is written to a dataset that feeds into subsequent stages.

In addition to data-driven leads generated by models (MSTD), fixed selections represent manually configured lead sets. The first processing step combines both sources and ensures that when a client appears in both a general subjourney and a specific subjourney, the specific marketing solution is retained and the general one is dropped.

5.1.2. Configurable Exclusions

This stage applies business-configurable filters to the lead set. These are exclusions chosen by business teams based on campaign requirements, targeting conditions, and eligibility criteria. They are applied before the pipeline considers offer-specific attributes such as flavor, tone, or taste.

The first major drop in lead counts typically occurs at this stage, driven primarily by duplicate messages and cases where too many channels have been selected for a single client. These exclusions are logged in the DT dataset with the appropriate FLT_OUT_RSN value.

5.1.3. Offer Selection

At this stage, the pipeline calculates which offer should be assigned to each eligible client. The selection is based on the client's marketing solution assignment, their product eligibility, and internal scoring models that determine the motive to buy. This stage is the primary reference point (OS) for all subsequent count comparisons.

5.1.4. Offer Enrichment

Offer Enrichment adds additional attributes to the selected offers. Specifically, it retrieves offerspace information and applies language-based filtering. The relevant logic is implemented in offer_enrichment.py and uses the enriched_ms_with_offerspace dataset.

Language handling at this stage is critically important. The pipeline maintains two separate language codes:

OFR_SPC_LGG (N, F, E, D): the language of the offerspace

PFR_LGG_CD (NL, FR, EN, DE): the client's preferred language

Only offerspaces matching the client's preferred language are retained. If a preferred language is missing, the offerspace is kept regardless. The filter function language_client_offerspace_filter assigns the value Offerspace_different_language to records that do not match, and these are logged as filter-outs.

At the end of each transform step, filter reasons are applied using the apply_filter_reasons function. Importantly, the total number of rows (leads) remains constant throughout enrichment; only the FLT_OUT_IC and FLT_OUT_RSN fields are updated.

5.1.5. Buffer Update and Buffer Selection

Leads that pass all previous filters are stored in a buffer. The buffer serves two purposes: it allows leads that could not be sent on a given day (e.g., due to a pipeline failure) to be retried on subsequent days, and it enables configurable time-based eligibility windows.

Buffer Update saves leads to the buffer and applies additional filtering. Buffer Selection then reads from the buffer, checks which leads are still eligible, applies channel limits, and ensures that within a client's full journey, only the lead with the highest value is selected.

The buffer stage introduces two specific filter-out types: one for replacing runs (which does not affect row counts because it replaces rows rather than removing them) and one for Other Journey Phase Selected, which is a genuine filter-out that should be logged in the buffer update traceability file.

5.2. Key Data Fields

Throughout the pipeline, each record carries a set of standardized fields that control its processing and logging:

Field	Description
FLT_OUT_IC	Filter-out indicator. 0 = lead continues; 1 = lead has been filtered out.
FLT_OUT_RSN	Filter-out reason. Standardized string identifying why the lead was excluded.
BUF_STS	Buffer status. Indicates the state of the lead within the buffer pipeline.
BUF_STS_CMT	Buffer status comment. Additional context for the buffer status.
MKT_SLT_ID	Marketing solution identifier. Links the lead to a specific campaign.
CPN_NO	Campaign number. Identifies the specific campaign instance.
TGT_PTY_GUID_NO	Target party GUID. Unique identifier for the targeted client.
CMN_PTY_GUID_NO	Communicated party GUID. Unique identifier for the client actually communicated to.

6. Results

GENERAL OBSERVATIONS

The initial count comparison across pipeline stages revealed systematic discrepancies across multiple marketing solutions. The key observations were:

- OE consistently produced more records than OS for certain marketing solutions, indicating unintended record multiplication
- The difference between OS and (LV + DT) was nonzero for all investigated solutions, confirming incomplete traceability
- The magnitude of discrepancies varied by marketing solution, suggesting the root causes were tied to specific data characteristics rather than universal pipeline bugs

Four distinct root causes were identified through the investigation, each contributing independently to the overall discrepancy.

6.1. Duplication in Offer Enrichment

Source

master ··· eab-leads-common / src / main / python / leads_common / transformations / **filters.py**

Source view | Diff to previous | History

```
399 399             & (col("OFR_SPC_CHANNEL").isin(*OfferspaceChannel.mail.value))
400 400             & (col("CMC_APP_EML_IC") == 1)
401 401             & (col("PG_BAS_DRT_MKT_IC") == 1)
402 402             & (col("PG_ETD_DRT_MKT_IC") == 0)
403 403             & (~col("PDC_GRP_PSS").rlike("G3700|G3832"))
404 404             & (col("PDC_GRP_PSS").rlike("G0190"))
405 405         ),
406 406         FilterMsg.non_digital_commercial_approval_email.value,
407 407     )
408 408
409     -   result = (result_non_corp
410     -               .unionByName(result_corp, allowMissingColumns=True)
411     -               .unionByName(result_non_corp_non_digital, allowMissingColumns=True)).drop_duplicates()
    409 +   result = (
    410 +       result_non_corp_non_digital
    411 +       .unionByName(result_corp, allowMissingColumns=True)
    412 +       .drop_duplicates()
    413 +   )
412 414
413 415     return result
414 416
415 417 def commercial_approval_filter_phone(enriched_ms_with_offerspace, ignore_exclusion_msid_list=tuple()):
416 418     """
417 419     Filters out offerspaces which are sent in a channel where the client did not give his commercial approval for.
418 420     This filter is specific for commercial approval phone
419 421     :param enriched_ms_with_offerspace:
420 422     :param ignore_exclusion_msid_list: list of marketing solutions for which the exclusion does not need to be executed
421 423     :return:
```

```
+----------+-----------------------------------------------------------------+
|FLT_OUT_IC|FLT_OUT_RSN                                                      |
+----------+-----------------------------------------------------------------+
|0         |[]                                                               |
|1         |[Uncertain commercial approval for email for a non-digital client]|
+----------+-----------------------------------------------------------------+
```

```
os unique keys = 1095738

oe unique keys = 1095738

duplicates group in oe_b on main keys : 17292

duplicates group in oe_e on main keys : 0
[Stage 119:=====================================================>   (16 + 1) / 17]
+------------------------------+------------------------------+------+----------+----------+-------------+-----+
|TGT_PTY_GUID_NO               |CMN_PTY_GUID_NO               |CPN_NO|MKT_SLT_ID|MKT_OFR_ID|MKT_OFR_SPC_ID|count|
+------------------------------+------------------------------+------+----------+----------+-------------+-----+
|de21eb51e6db5468fe8bf0e42a4fe751|de21eb51e6db5468fe8bf0e42a4fe751|0001  |-709146693|-709013533|545052127     |2    |
|7191ae5ac9e2b15bce3f9ad95b9bdfcb|ed7816d84b4bd2a7a290ff34f815d835|0001  |-709146693|-709013533|545080682     |2    |
|9fa3215764abbfb9e5e2a511b6d25f07|9fa3215764abbfb9e5e2a511b6d25f07|0001  |-709146693|-709013533|545052127     |2    |
|f79c4d31a0112b6081a7591b344d8b96|f79c4d31a0112b6081a7591b344d8b96|0001  |-709146693|-709013533|545080682     |2    |
|f13b927a96a4000a1c9994ea0ab9d995|f13b927a96a4000a1c9994ea0ab9d995|0001  |-709146693|-709013533|545052079     |2    |
|165025bc003d8df3ba229f525c0e35ad|165025bc003d8df3ba229f525c0e35ad|0001  |-709146693|-709013533|545080652     |2    |
|b96e2b9b13162f61a812c57019cfe685|b96e2b9b13162f61a812c57019cfe685|0001  |-709146693|-709013533|545080652     |2    |
|f7c400861cb95959adce03662d5fb9f7|f7c400861cb95959adce03662d5fb9f7|0001  |-709146693|-709013533|545080682     |2    |
|3373a87d660d8efa44e9635c411f03d2|3373a87d660d8efa44e9635c411f03d2|0001  |-709146693|-709013533|545052079     |2    |
|4f3756291303c76067cc497a492d7757|4f3756291303c76067cc497a492d7757|0001  |-709146693|-709013533|545080682     |2    |
+------------------------------+------------------------------+------+----------+----------+-------------+-----+
only showing top 10 rows
```

kbcMS_349514

				Number of Channels	5				
		Actual leads filtered out			nb_b	270034	700	nb_e	222013	1
offerId	-709029226	QLIK	Notebook		fsr_b	270734	0	fsr_e	222014	0
PDATE		1371411	1,410,585	-39,174	ce_b	1353670	0	ce_e	1110070	0
LEDAS	E0774	Filtered out			os_b	1353670	39174	os_e	1110070	0
		QLIK	Notebook							
		1410585	1410585	0	oe_b	1392844	0	oe_e	1110070	0
		DIFFERENCE								
		QLIK	Notebook		lv_b	502134		lv_e	590195	
		-39174	0		dt_b	890710		dt_e	519875	
		-39174								

Root Cause

GDPR-related join logic between parent and child client records produced multiple output rows per input lead, inflating OE record counts.

CAUSE

In the Offer Enrichment stage, leads are joined with a reference dataset that contains GDPR consent information. This dataset has a parent-child structure: a parent client record can have multiple associated child records representing different consent scopes or household relationships. The join was implemented without adequately handling this one-to-many relationship, meaning that a single input lead could match multiple child records and produce multiple output rows.

EFFECT

The duplication caused inflated row counts in OE compared to OS. For some marketing solutions, the inflation factor was significant (multiple duplicate rows per original lead). This had downstream effects:

- LV inherited the duplicates, producing a larger-than-expected lead set
- Business reporting showed inflated lead counts, reducing trust in dashboard figures
- Deduplication at LV was inconsistent because the logic assumed uniqueness that was already violated in OE

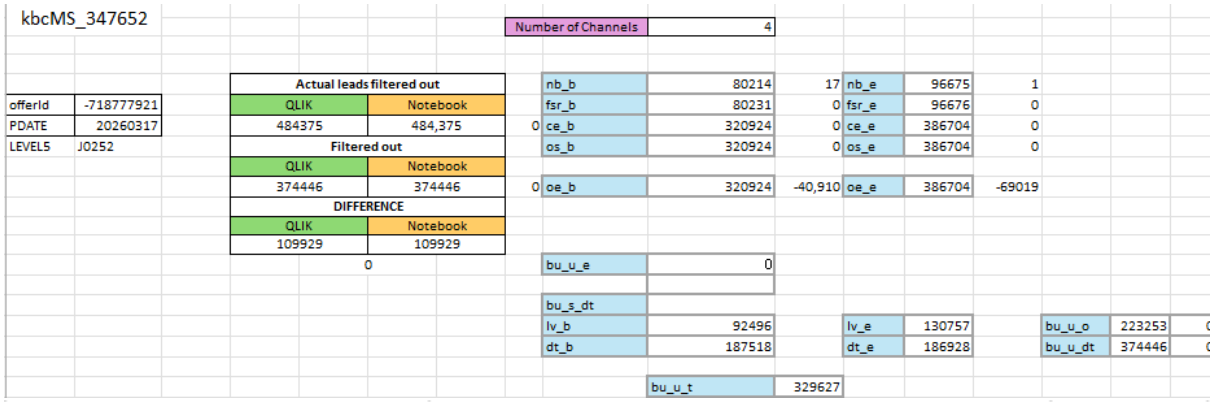
FIX APPLIED

The join logic in OE was corrected to enforce a one-to-one relationship between input leads and their GDPR consent records. Specifically, a deduplication step was introduced after the join, keyed on the correct business

identifier, ensuring that each input lead produces at most one output record in OE. The fix was validated by confirming that OE record counts matched OS counts for all affected marketing solutions.

In one investigated case, the duplication caused Offer Enrichment to contain more records than Offer Selection despite no additional leads being introduced. This immediately indicated that record multiplication rather than legitimate business logic was responsible for the discrepancy.

6.2. Missing Buffer Traceability



BUF_STS	BUF_STS_CMT	count
Replaced by new run	Replaced by new run 2026-03-15 00:00:00	219698
Other journey phase selected	Found higher lead_value in journey	109929

Root Cause

Leads filtered out during the buffer stage were not written to the DT dataset, making them invisible to business reporting and creating unexplained count gaps.

CAUSE

The pipeline includes a buffer stage that temporarily stores leads between processing cycles. During the buffer update process, some leads are filtered out based on business rules (e.g., the client has already received the communication, or the lead no longer meets eligibility criteria). These filter-outs were handled correctly from a processing perspective, but the corresponding records were never written to the DT dataset.

EFFECT

Because buffer filter-outs were absent from DT, the count equation OS = LV + DT was always violated for solutions that had active buffer filtering. The discrepancy could not be explained using the available DT data, making it appear as if leads were simply disappearing. This had caused confusion during previous manual investigations.

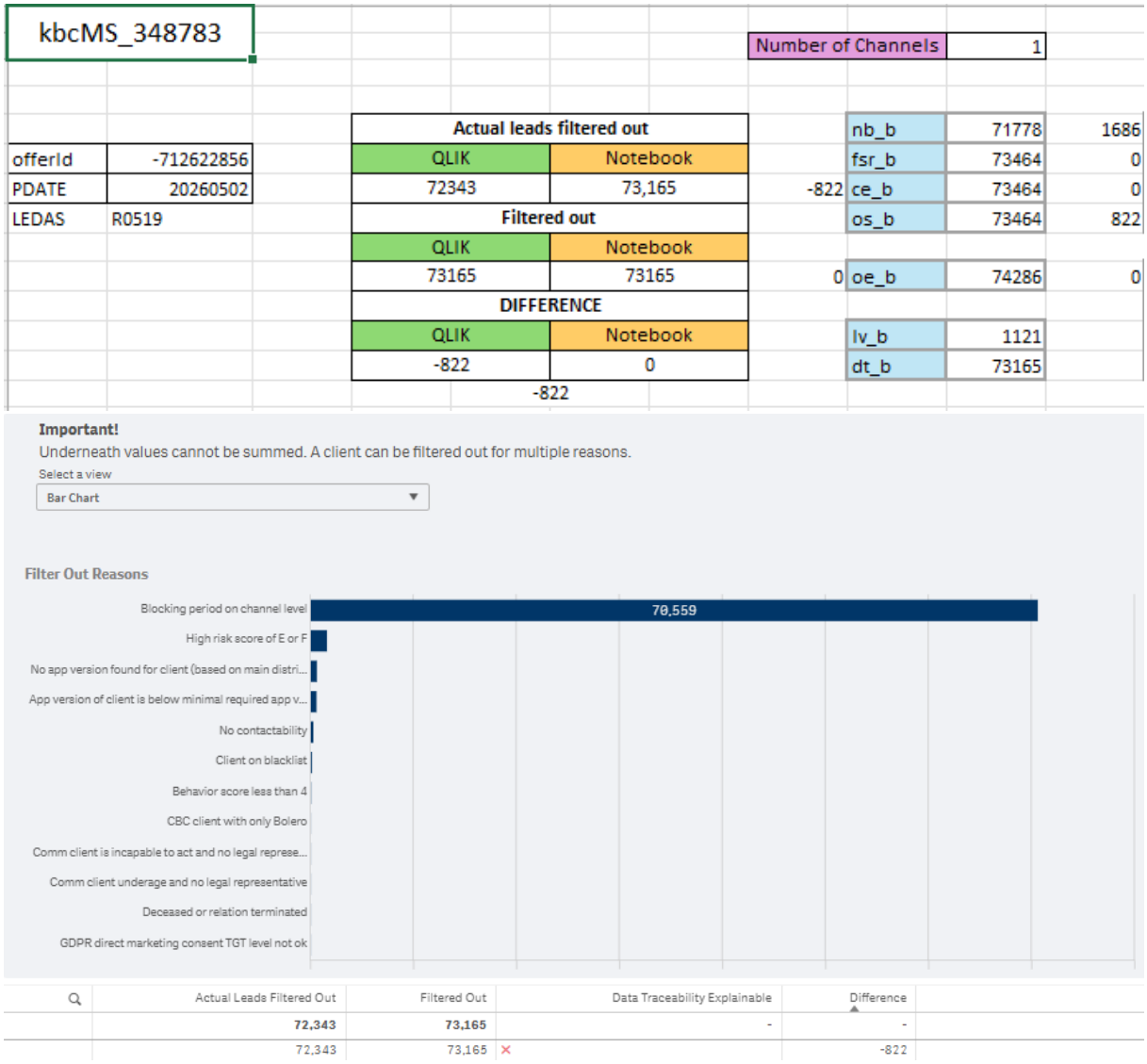
RECOMMENDED SOLUTION

The buffer stage should be extended to write filter-out records to the DT dataset with appropriate reason codes. As an intermediate measure, the Qlik reporting layer should be extended to also consume the buffer

traceability dataset (which does exist separately but is not currently included in reporting). This avoids high-risk changes to production pipeline logic while making the data visible to business stakeholders.

The discrepancy was confirmed by comparing pipeline outputs with Data Traceability records and identifying leads that were filtered during buffer processing but never appeared in the DT dataset.

6.3. Blocking Table Issue



kbcMS_348783

offerId	-712622856
PDATE	20260502
LEDAS	R0519

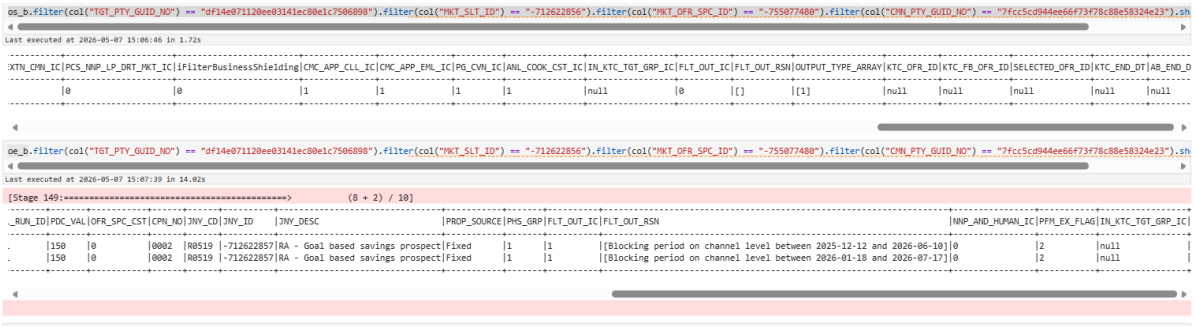
Actual leads filtered out	
QLIK	Notebook
72343	73,165
Filtered out	
QLIK	Notebook
73165	73165
DIFFERENCE	
QLIK	Notebook
-822	0
-822	

	Number of Channels	1	
	nb_b	71778	1686
-822	fsr_b	73464	0
	ce_b	73464	0
	os_b	73464	822
0	oe_b	74286	0
	lv_b	1121	
	dt_b	73165	

Important!
Underneath values cannot be summed. A client can be filtered out for multiple reasons.
Select a view
Bar Chart

Filter Out Reasons

	Actual Leads Filtered Out	Filtered Out	Data Traceability Explainable	Difference
	72,343	**73,165**	-	-
	72,343	73,165	✕	-822



os_b.filter(col("TGT_PTY_GUID_NO") == "df14e071120ee03141ec80e1c7506898").filter(col("MKT_SLT_ID") == "-712622856").filter(col("MKT_OFR_SPC_ID") == "-755077480").filter(col("CMN_PTY_GUID_NO") == "7fcc5cd944ee66f73f78c88e58324e23").sh

Last executed at 2026-05-07 15:06:46 in 1.72s

XTN_CMN_IC	PCS_NNP_LP_DRT_MKT_IC	iFilterBusinessShielding	CMC_APP_CLL_IC	CMC_APP_EML_IC	PG_CVN_IC	ANL_COOK_CST_IC	IN_KTC_TGT_GRP_IC	FLT_OUT_IC	FLT_OUT_RSN	OUTPUT_TYPE_ARRAY	KTC_OFR_ID	KTC_FB_OFR_ID	SELECTED_OFR_ID	KTC_END_DT	AB_END_D
|0	|0	|1	|1	|1	|1	|null	|0	|[]	|[1]	|null	|null	|null	|null	|null	

oe_b.filter(col("TGT_PTY_GUID_NO") == "df14e071120ee03141ec80e1c7506898").filter(col("MKT_SLT_ID") == "-712622856").filter(col("MKT_OFR_SPC_ID") == "-755077480").filter(col("CMN_PTY_GUID_NO") == "7fcc5cd944ee66f73f78c88e58324e23").sh

Last executed at 2026-05-07 15:07:39 in 14.02s

[Stage 149:==> (8 + 2) / 10]

_RUN_ID	PDC_VAL	OFR_SPC_CST	CPN_NO	JNY_CD	JNY_ID	JNY_DESC	PROP_SOURCE	PHS_GRP	FLT_OUT_IC	FLT_OUT_RSN	NNP_AND_HUMAN_IC	PFM_EX_FLAG	IN_KTC_TGT_GRP_IC
|150	|0	|0002	|R0519	|-712622857	|RA - Goal based savings prospect	|Fixed	|1	|1	|[Blocking period on channel level between 2025-12-12 and 2026-06-10]	|0	|2	|null	
|150	|0	|0002	|R0519	|-712622857	|RA - Goal based savings prospect	|Fixed	|1	|1	|[Blocking period on channel level between 2026-01-18 and 2026-07-17]	|0	|2	|null	

Root Cause

Incorrect windowing logic when reading the blocking table caused some leads to be double-counted or incorrectly blocked, contributing to minor count discrepancies.

CAUSE

The blocking table is a reference dataset that records which clients should be excluded from communications during a specified blocking period. When reading this table, the pipeline used a window function to select the relevant blocking record for each client. The windowing logic did not correctly handle cases where a client had multiple blocking records with overlapping or adjacent date ranges.

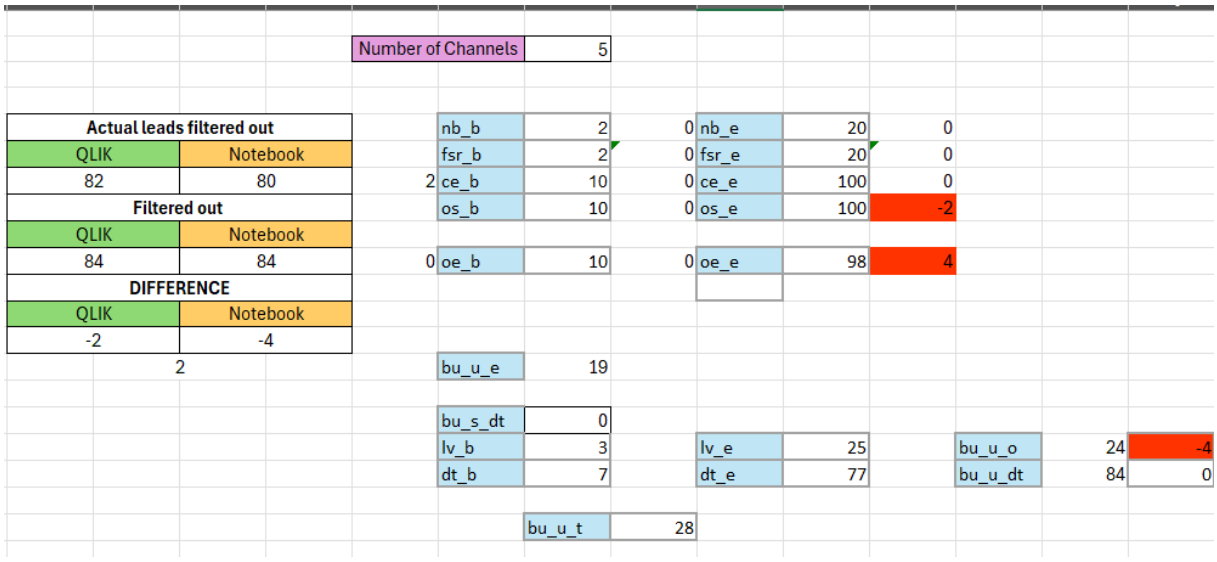
EFFECT

In these cases, the window function could select an outdated blocking record (with an earlier end date) instead of the most recent one, causing leads to be incorrectly blocked or unblocked. The magnitude of this issue was smaller than the other findings, but it contributed measurably to count discrepancies.

FIX APPLIED

The windowing logic was corrected to always select the record with the latest blocking end date for each client. This ensures that the most current blocking status is used when evaluating lead eligibility. The fix was validated by confirming that affected leads were correctly classified as blocked or unblocked across historical pipeline runs.

6.4. Duplication in Buffer Update Pipeline



Actual leads filtered out	
QLIK	Notebook
82	80
Filtered out	
QLIK	Notebook
84	84
DIFFERENCE	
QLIK	Notebook
-2	-4

Number of Channels 5

		nb_b	2	0	nb_e	20	0				
		fsr_b	2	0	fsr_e	20	0				
2		ce_b	10	0	ce_e	100	0				
		os_b	10	0	os_e	100	-2				
0		oe_b	10	0	oe_e	98	4				
2		bu_u_e	19								
		bu_s_dt	0								
		lv_b	3		lv_e	25		bu_u_o	24	-4	
		dt_b	7		dt_e	77		bu_u_dt	84	0	
			bu_u_t	28							

```
# Step 4 calculate enrichments
logger.info(".... Buffer Enrichments")
buffer_enrichments = (
    buffer_enrichments
    .transform(self.remove_updated_enrichments, enrichments)
    .transform(self.union_all_enrichments, enrichments)
    .transform(self.select_relevant_enrichments, self.select_unique_buffer_values(buffer_result.Out))
)

return OutTraEnr(
    Out=buffer_result.Out.select(*self.select_buffer_columns),
    Tra=buffer_result.Tra.select(*self.select_buffer_columns),
    Enr=buffer_enrichments.select(*self.buffer_enrichment_schema.fieldNames()),
)
```

```
missing_from_oe = os_df.join(
    oe_df,
    on=keys,
    how="left_anti"
)
missing_from_os = oe_df.join(
    os_df,
    on=keys,
    how="left_anti"
)
```

```
Last executed at 2026-04-23 15:20:02 in 12ms
```

```
missing_from_oe.count()
```

```
Last executed at 2026-04-23 15:20:20 in 17.13s
```

```
26
```

```
missing_from_os.count()
```

```
Last executed at 2026-04-23 15:20:34 in 14.59s
```

```
24
```

ROOT CAUSE

During the investigation of discrepancies between Offer Enrichment and Buffer Update outputs, duplicate records were identified inside the buffer enrichment process. Detailed step-by-step analysis showed that the duplication originated in the SELECT_RELEVANT_ENRICHMENTS transformation.

CAUSE

The buffer enrichment process consists of three main stages:

- remove_updated_enrichments
- union_all_enrichments
- select_relevant_enrichments

The first two stages produced the expected number of records. However, during select_relevant_enrichments, key columns used to distinguish records were removed from the dataset.

Specifically, columns such as:

buf.TGT_PSN_CLT_NO

buf.CMN_PSN_CLT_NO

were dropped during the transformation.

Before these columns were removed, two records were technically different because they referred to different communicated-client combinations. After the columns were removed, the remaining attributes became identical, causing previously distinct records to collapse into duplicates.

EFFECT

This resulted in duplicate enrichment records being written to the buffer enrichment dataset, causing inflated row counts and inconsistencies between pipeline stages.

PROPOSED SOLUTION

Two potential solutions were to

Retain the distinguishing columns throughout the enrichment process so that uniqueness is preserved.

or

Apply a dropDuplicates() operation after the transformation if business rules confirm that only a single record should remain.

The preferred solution is to preserve the distinguishing columns, as this maintains data integrity and avoids removing potentially valid business records.

VALIDATION

The issue was reproduced locally by executing the transformation step-by-step and comparing row counts before and after the column removal. Validation confirmed that the duplicates were introduced only after the distinguishing columns were droppe

6.5. Validation

To validate the findings, we used:

Multi-solution testing: findings were reproduced, and fixes were verified across at least three different marketing solutions to confirm generalizability.

Pre/post count comparison: record counts at each pipeline stage were measured before and after fixes to confirm that the OS = LV + DT equation was satisfied

Qlik dashboard cross-check: business-facing reports were reviewed to confirm that corrected counts aligned with dashboard figures.

Peer validation: root cause explanations and proposed fixes were reviewed by LeCa team members with domain expertise.

6.6. Qlik Dashboard Analysis and Improvement Proposal

During the investigation, the Qlik dashboard was used extensively to identify marketing solutions with traceability discrepancies and to validate findings against business-facing reporting. While the dashboard provides useful visibility into lead counts and traceability metrics, several limitations were identified.

First, the dashboard does not expose the complete pipeline flow. Users can observe final lead counts and traceability values, but it is difficult to determine at which pipeline stage records are lost or duplicated. This often requires manual analysis using PySpark notebooks.

Second, buffer traceability information is not directly visible in the dashboard. Although buffer update processes generate traceability datasets containing status information and comments, these datasets are not integrated into the reporting layer. As a result, filter-outs occurring in the buffer stage can appear as unexplained discrepancies.

Third, expected lead calculations are currently based on Offer Selection outputs. Since additional filtering and enrichment logic is applied in Offer Enrichment, this can create misleading differences between expected leads and final communicated leads.

Based on these findings, several improvements were proposed:

1. Integration of buffer traceability datasets into reporting.
2. Addition of pipeline-layer reconciliation views showing counts at each stage of the pipeline.
3. Standardized grouping of traceability reasons.
4. Using Offer Enrichment rather than Offer Selection as the baseline for expected lead calculations.

These improvements were documented and discussed with stakeholders as future enhancements that would significantly improve operational visibility and troubleshooting capabilities.

7. Proposed dashboard enhancements

- Pipeline-layer reconciliation view
- Buffer traceability visibility
- Standardized traceability reason grouping
- Offer Enrichment based expected lead calculations
- Lead flow waterfall visualizations
- Traceability root-cause analysis dashboards

These improvements were analyzed during the internship but were not implemented due to dependencies on shared ETL and DataModel components. The proposals have been documented to support future development.

8. Evaluation

ACHIEVEMENTS

Developed a reusable, structured debugging methodology for investigating pipeline discrepancies, which can be applied by the LeCa team in future investigations without requiring extensive prior knowledge of the codebase.
Identified and documented four distinct root causes contributing to data inconsistencies in the LEDAS pipeline.
Fixed the major duplication issue in Offer Enrichment caused by the incorrect union between parent and child records, directly improving the accuracy of pipeline outputs.
Fixed the blocking table windowing issue, eliminating a consistent source of small discrepancies.
Fixed the duplication in the buffer update pipeline caused by incorrect column dropping in select_relevant_enrichments.
Identified the missing buffer traceability issue and proposed and validated the solution (PRPRS change in Qlik).
Produced comprehensive documentation of the pipeline architecture and filter logic that improves the team's understanding of system behavior.

RISKS

Misinterpreting filtering logic: Given the complexity of the pipeline, there was a risk of incorrectly attributing a discrepancy to the wrong cause. This risk was mitigated by validating each finding across multiple marketing solutions and reviewing with team members.
Hidden transformations: Some transformations that affected row counts were not immediately visible in the code or logs. These required careful step-by-step inspection to uncover.
Incomplete traceability after fix: There remained a risk that not all filter-out paths were fully covered by the fixes applied. The structured methodology helps address this but cannot guarantee completeness without exhaustive testing across all pipeline variants.
Framework changes breaking traceability: As noted in the technical documentation, changes to the pipeline framework (such as write-basic vs write(df, 'basic') syntax changes or dataset path component changes) can break traceability writes in ways that are not immediately obvious. This represents an ongoing risk for the team.

METRICS

Difference in row counts between OS, OE, LV, and DT: the primary metric for identifying and validating discrepancies.

Number of unexplained records: records present in OS but absent from both LV and DT, which should approach zero after all fixes are applied.

Percentage of traced leads: the proportion of all leads that have a valid, documented outcome (either communicated or filtered out with a reason), which should approach 100%.

Number of root causes identified and resolved: four root causes were identified, three of which were directly fixed in the pipeline code, and one of which required a reporting layer change.

9. Technical Reflection

One of the biggest challenges at the start of the internship was understanding the scale of the LEDAS ecosystem. The pipelines process thousands of records daily and consist of numerous interconnected stages, datasets, and business rules. Initially it was difficult to understand how data moved through the system and which components were responsible for specific transformations. To overcome this challenge, I created personal notes, diagrams, and data-flow overviews while regularly validating my understanding with team members. This approach gradually transformed a complex and unfamiliar environment into a system that could be analyzed systematically and confidently.

One of the most important lessons learned during the internship was that debugging distributed data pipelines requires a fundamentally different mindset compared to traditional application debugging. In Spark pipelines, transformations are executed lazily, intermediate datasets are hidden behind chained operations, and a single incorrect join or dropped column can silently multiply or remove thousands of records without triggering any error.

Initially, the investigation relied primarily on row-count comparison as a diagnostic tool. While effective for locating discrepancies, count comparison alone proved insufficient when the definition of a unique key was not fully understood. Several early assumptions proved incorrect because duplicate records were technically distinct before certain transformations removed the identifying columns.

A particularly important distinction emerged between data loss and reporting loss. During the buffer traceability investigation, filtered leads initially appeared to be disappearing from the system entirely. Later investigation revealed that these records existed in a separate traceability dataset but were simply not included in the Qlik reporting layer. This distinction significantly changed both the interpretation of the issue and the appropriate solution — rather than modifying production pipeline logic, the fix was to extend the reporting layer.

10. Rejected Alternative Solutions

Several alternative approaches were considered and ultimately rejected during the investigation:

Adding a generic distinct() operation after the Offer Enrichment stage would have technically removed duplicate records, but it would also have hidden the underlying join issue and potentially removed valid business records that happened to share the same business key. This approach was rejected because it prioritized symptom suppression over root cause resolution.

Rewriting large portions of the traceability logic to centralize all filter-outs into a single dataset was considered as a long-term architectural improvement. While technically cleaner, this would have introduced significant operational risk into production pipelines and was beyond the scope of the internship. The safer approach was to extend the reporting layer to consume existing traceability datasets.

Applying broad data quality checks after every stage (e.g., automated duplicate detection on every pipeline run) was discussed but deferred. While valuable, implementing such checks requires careful design to avoid performance degradation on large-scale Spark jobs.

11. Challenges

The internship involved several engineering challenges that extended beyond the functional debugging work itself.

One significant challenge was understanding how the same business concept was implemented across multiple pipeline stages and frameworks. The pipeline and related systems share similar architectural patterns but contain framework-specific variations. A behavior that was correct in one context could be incorrect in another, making it difficult to immediately determine whether a specific implementation was intentional or a defect. Resolving this ambiguity required reading source code across multiple repositories and consulting with team members who had institutional knowledge of specific design decisions.

A second challenge was performance-related debugging. Re-running Spark transformations during notebook analysis triggered large distributed computations involving joins across datasets with tens of millions of rows. This made exploratory debugging expensive in both time and compute resources. The response to this challenge was to develop a more disciplined, hypothesis-driven debugging approach: forming a specific hypothesis about the root cause, designing a targeted test to validate it, and only re-running the minimum pipeline steps needed to evaluate that test.

A third challenge was developing sufficient domain knowledge to correctly interpret business logic. Marketing pipeline rules are driven by business requirements that are often not fully documented in code. Understanding why a particular filter was applied, or why a specific join was designed a certain way, required ongoing dialogue with business and technical stakeholders. This highlighted the importance of embedding in the team and participating in knowledge-sharing sessions rather than working purely independently.

12. Conclusion

This internship demonstrated that data traceability in distributed marketing pipelines is not only a technical implementation detail but a critical operational requirement. Without reliable traceability, business reporting becomes difficult to trust, debugging becomes reactive rather than proactive, and small inconsistencies can propagate across multiple downstream systems in ways that are difficult to reverse.

The project confirmed that the observed discrepancies in the pipeline were not caused by a single isolated bug, but by the interaction of multiple independent issues: a GDPR join multiplying records, a buffer stage silently swallowing filtered leads, a windowing function selecting stale blocking records, and a column-dropping transformation destroying record uniqueness. Each of these issues was technically subtle and would have been difficult to identify without a systematic reconciliation methodology.

Beyond the individual fixes, the most durable outcome of the project is the reusable debugging methodology. This methodology provides a structured, evidence-based way to investigate future pipeline inconsistencies and reduces the dependency on implicit or tribal system knowledge. It can be applied by any engineer on the LeCa team, regardless of their familiarity with specific pipeline components.

From a personal perspective, the internship significantly deepened my understanding of enterprise-scale data engineering, distributed processing with Apache Spark, and the practical realities of maintaining production data pipelines. It also demonstrated the importance of combining technical analysis with clear communication: identifying a root cause is only valuable if it can be explained clearly to both technical peers and business stakeholders.

Beyond improving technical correctness, the findings also improve business confidence in reporting and campaign monitoring. By reducing unexplained discrepancies and improving traceability visibility, future investigations can be performed more efficiently and with greater certainty. This ultimately reduces operational effort and increases trust in the data used for marketing decisions.

The internship was a valuable experience that bridged academic knowledge and professional practice, and I am grateful to the LeCa team at KBC for the opportunity.

REFERENCE LIST

The following sources were consulted during the internship project. Internal KBC documentation and source code repositories are referenced in general terms to respect confidentiality.

- Apache Software Foundation. (2024). Apache Spark Documentation. https://spark.apache.org/docs/latest/
- Apache Software Foundation. (2024). Apache Airflow Documentation. https://airflow.apache.org/docs/
- KBC Bank & Insurance. (2024). Internal LEDAS pipeline documentation and architecture diagrams. [Internal, confidential]
- KBC Bank & Insurance. (2024). Internal Level5 pipeline documentation and architecture diagrams. [Internal, confidential]
- KBC Bank & Insurance. (2024). LeCa team Bitbucket repositories — LEDAS pipeline source code. [Internal, confidential]
- KBC Bank & Insurance. (2024). LeCa team Bitbucket repositories — LEVEL5 pipeline source code. [Internal, confidential]
- Qlik Technologies. (2024). Qlik Sense documentation. https://help.qlik.com/