



What is CDP Identity Resolution Missing?

WHITEPAPER

Table of Contents

01	Introduction	03
<hr/>		
02	What CDP Identity Resolution is Missing: Most of your Customer Data	05
<hr/>		
03	How Identity Resolution Works	07
<hr/>		
	The Problem	
	Types of Matching	
04	Choosing a Matching Algorithm	16
<hr/>		
	False Positives and False Negatives	
	Cost	
	In-Place vs In-Memory Processing	
	Putting it All Together - Choosing a Matching Algorithm	
05	Bottom Line: Handle Identity Resolution in the Data Warehouse	21
<hr/>		
06	Ready to get started?	22
<hr/>		

01. Introduction

If you work within a data-driven company or organization, chances are you've come up against the challenge of creating a single view of your customers in order to more accurately and effectively communicate with them based on their data.

Maybe you tried to solve it in the data warehouse, or by using a third-party Customer Data Platform (CDP). Can you confidently say that you've solved Identity Resolution and have a single view of your customers?

Once you do, analyzing and activating your customer data is easy. If you're not quite there yet, have no fear - you're not alone! Let's dive in together on the foundations of Identity Resolution and find out why the data warehouse is the best place to build this foundation, and why CDPs do not give you the full picture.

Put yourself in the shoes of Mia, one of your customers. Mia communicates with your organization in many different ways. She may call customer support if she has an issue with a product or service. She may browse your website or app and take any number of actions. She may - or may not - open an email or SMS from your marketing or sales team. Each one of these interactions creates a record in a database.

Identity Resolution is the act of reconciling those records to create a single view of Mia. If you can do it accurately, you end up with a way to tie together everything you know about Mia so you can communicate with her in a way that is more valuable to both her and to your organization.

If you're not able to do it accurately, you could mistake someone else for Mia, send her irrelevant and spammy messages, or disclose something to Mia you shouldn't have. At best, this is frustrating for your customer and inefficient for your organization. On the other end of the spectrum, if you operate in a regulated industry, this could even mean governance or legal issues. Either way, you'll want to put in the effort to get this right.

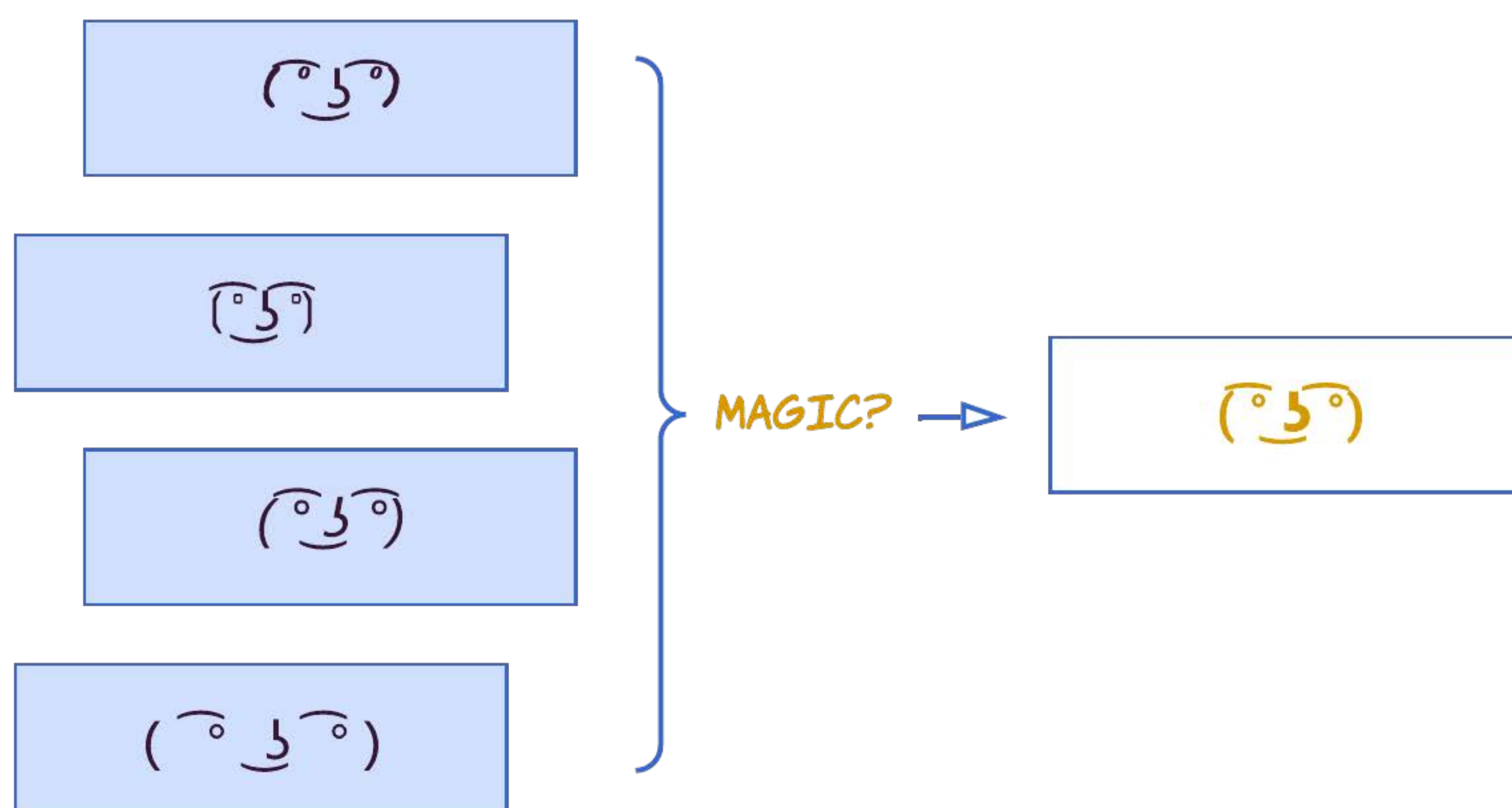
01. Introduction (cont.)

Some data leaders have chosen to outsource the Identity Resolution problem to their Customer Data Platform (CDP) partners.

Today we're seeing many innovative data and analytics leaders move away from CDP-based Identity Resolution to build this vital process into the heart of their **Modern Customer Data Stack**.

Let's take a look at a few key topics regarding Identity Resolution, to set you and your team up for success as you work to better engage with Mia and the rest of your customers:

1. What CDP Identity Resolution is Missing
2. How Identity Resolution Works
3. Choosing a Matching Algorithm
4. Bottom Line: Handle Identity Resolution in the Data Warehouse



02. What CDP Identity Resolution is Missing

CDPs are systems that can sit at the nexus of your websites and applications.

The general idea is that your front end engineers deploy one common set of tags and SDK elements across all of your properties then the CDP centrally translates those events and user metadata into the correct format for many downstream services (think Google Analytics, Facebook, Adobe Analytics, Optimizely, etc) instead of implementing each of the SDKs for the downstream services.

Event collection and tracking are CDPs bread and butter, and they do it well, but these CDP companies have recently been trumpeting their Identity Resolution prowess in addition to their event tracking.

Even if CDPs could possibly execute Identity Resolution perfectly (they can't), there is one big problem: **CDPs do not have access to most of your customer data that you've already collected in your data warehouse or data lake.**

CDPs do not have access to most of your customer data that you've already collected in your data warehouse or data lake.

02. What CDP Identity Resolution is Missing (cont.)

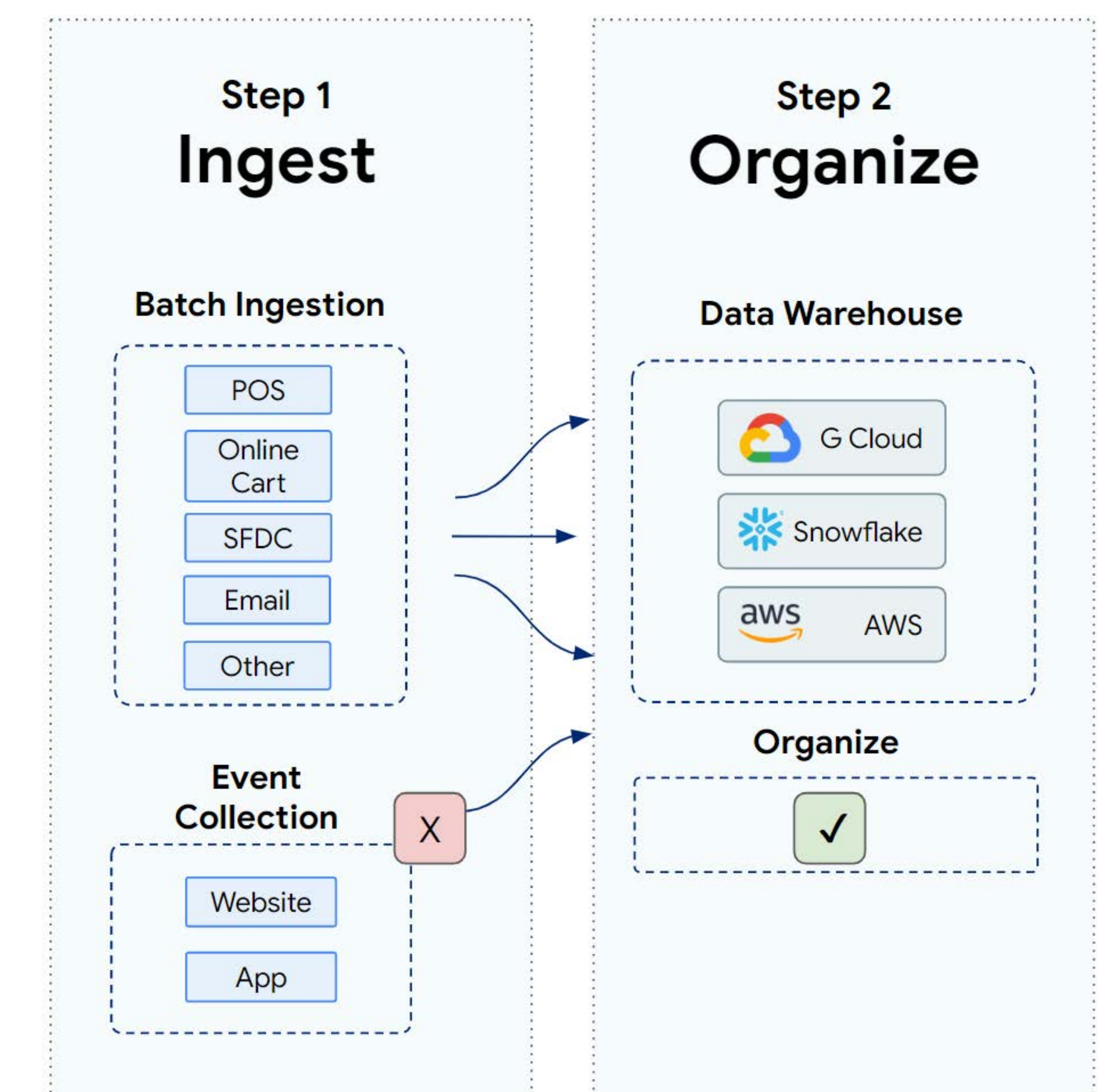
Having a CDP be the foundation of your Identity Resolution strategy would be akin to asking your CRM or Email platform to tackle this problem. These are all platforms that only ever see a small percentage of your customer data.

If you take this approach, where you prioritize Identity Resolution in one or more of your edge platforms, you fail to achieve your goal of a Single View of the Customer - you resolve a view of your customer for web properties, but still have a large portion of the valuable customer data locked up in unresolved records in the Data Warehouse.

This can be problematic if you want to examine or activate customer data unrelated to the data available in your CDP. For example, if you'd like to build a revenue dashboard in Tableau that shows revenue by types of customer, and your CDP does not have access to your company's transactional data, the CDP-based identity resolution is not going to be very helpful to you.

Any Identity Resolution strategy needs to have its foundation in the place where your customer data is aggregated. In the **Modern Customer Data Stack**, that's the data warehouse.

Strong data validation in edge platforms where customer data is being collected, like CRMs and CDPs, is important and Identity Resolution across web properties is valuable, but all of these should be inputs to a central Identity Resolution strategy built on top of the data warehouse.



03. How Identity Resolution Works

The Problem

The example in the introduction above, with Mia, addresses Identity Resolution for people. What we're interested in is resolving the identities of all of the people we communicate with.

In this example, the 'person' is the **entity** for which we are trying to resolve identities. You can also choose to resolve identities for different entities as well.

If Mia works for a company you sell to, you might want to maintain a company or account entity and resolve identities across companies or accounts as well. There may be parent accounts or parent companies that manage multiple accounts or companies, which would be another entity.

You can think of entities as nouns that your organization or business uses to frame and operate the organization (customers or users, transactions, products, events, etc).

For identity resolution, we care about entities that your organization might want to communicate with - like users or customers or prospects or people; accounts or groups of people; parent accounts or groups of accounts. To build a foundation for Identity Resolution, we need to answer a seemingly obvious question: how do you define these entities?

Let's use people. How do you define a person when you're looking at records in a database? Some fields on a record for Mia may look like this:

EMAIL	PHONE	FIRST_NAME	LAST_NAME	DATE_OF_BIRTH	ADDRESS	...
MIA@IJK.NET	8885551212	MIA	PAZ	4/3/1968	123 MAIN ST	...

03. How Identity Resolution Works

The Problem (cont.)

So what field here defines Mia? Email, perhaps? We can formalize this into a rule and say all rows with the same email address represent the same person.

Well, what happens if Mia changes her email address over time? Or if Mia shares an email address with someone in her household? Let's take a look at this set of records below.

Intuitively, it looks like the first two rows represent the same person and the third does not. If we naively decide to resolve identities using only the email field, we would not have come to the same conclusion and we would have decided that the first and third rows represent the same person and that the second row was someone different.

Naively using one column to resolve identities seems like a limiting approach. If you look closely, you'll see that if we chose the 'phone' field we would have been equally unsuccessful.

The first key problem of Identity Resolution is selecting that rule or algorithm.

Identity resolution is the process of using a matching rule or algorithm to determine which records represent the same entity.

EMAIL	PHONE	FIRST_NAME	LAST_NAME	DATE_OF_BIRTH	ADDRESS	...
MIA@IJK.NET	8885551212	MIA	PAZ	4/3/1968	123 MAIN ST	...
MIA@HIJK.NET	8885551213	MIA	PAZ	4/3/1968	123 MAIN ST	...
MIA@IJK.NET	8885551212	SAM	PAZ	3/4/1970	123 MAIN ST	...

03. How Identity Resolution Works

Types of Matching

When selecting a rule or algorithm for Identity Resolution, you can think of a continuum of options on an axis from simple to sophisticated.

In our example above, using one field like 'email' or 'phone' would fall near the left. At the opposite end of the spectrum, you could use something very sophisticated like classification with a state-of-the-art Transformer (machine learning model).

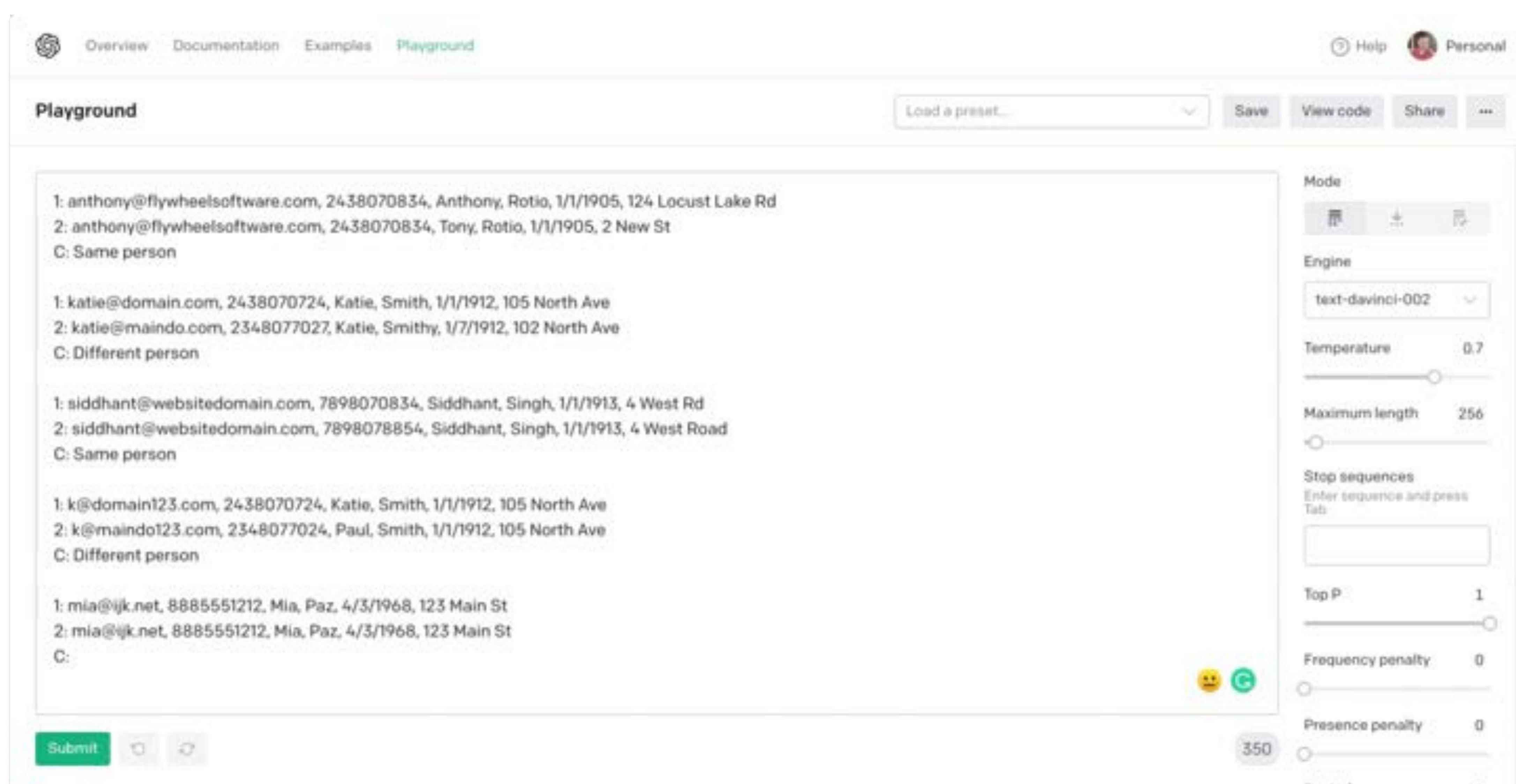
Here's GPT-3 Davinci (close to state-of-the-art at the time of this writing) determining if two records represent the same person after being given a few examples:

Between these two extremes, there exist many approaches that could be a good fit for your data ecosystem. Let's go over a few.

1. Single Field Matching

2. Composite Field Matching

3. Probabilistic Matching



03. How Identity Resolution Works

Types of Matching (cont.)

Single Field Matching

This approach uses a simple rule like if email=email across two records, they represent the same person. So an approach to implementing this method might look something like this:

1. **Gather** all of the source tables that contain records about your customers.
2. **Dedupe** the source tables on the field that you will be matching on.
3. **Write** a query that joins all of the tables on the field you selected for matching.
4. **Decide** which additional fields to select from each table.
5. **For fields that only exist on one source table**, you can simply select them.
6. **For fields that exist across tables**, coalesce each field from each source with your desired ranking or precedence.

Composite Field Matching

Here we develop a rule that is a bit more strict and say that a match exists only if several fields match across records. These are the key steps for composite key identity resolution in your Data Warehouse:

1. **Identify match keys:**
The first step is to determine which fields or columns you'll be using to determine which individuals are the same individual within and across sources. A typical example of match keys might be an email address and last name.
2. **Create a Composite Key:**
When you have your match keys identified, you can combine them into a composite key in each of the data sources, which combines the match keys into one field. A simple approach to deduping is to use the composite key to find duplicates and rank the duplicates in order from most recent to oldest updated timestamp, but there are a few options when assigning a **source rank** to each record. Write these new fields to a new version of your source tables.

03. How Identity Resolution Works

Types of Matching (cont.)

Composite Field Matching (cont.)

3. Aggregate Source Tables:

The next step is to union the previously created tables into a source lookup table that has all the customer records from all of your source tables. We call this the lookup table.

4. Define a Precedence Matrix:

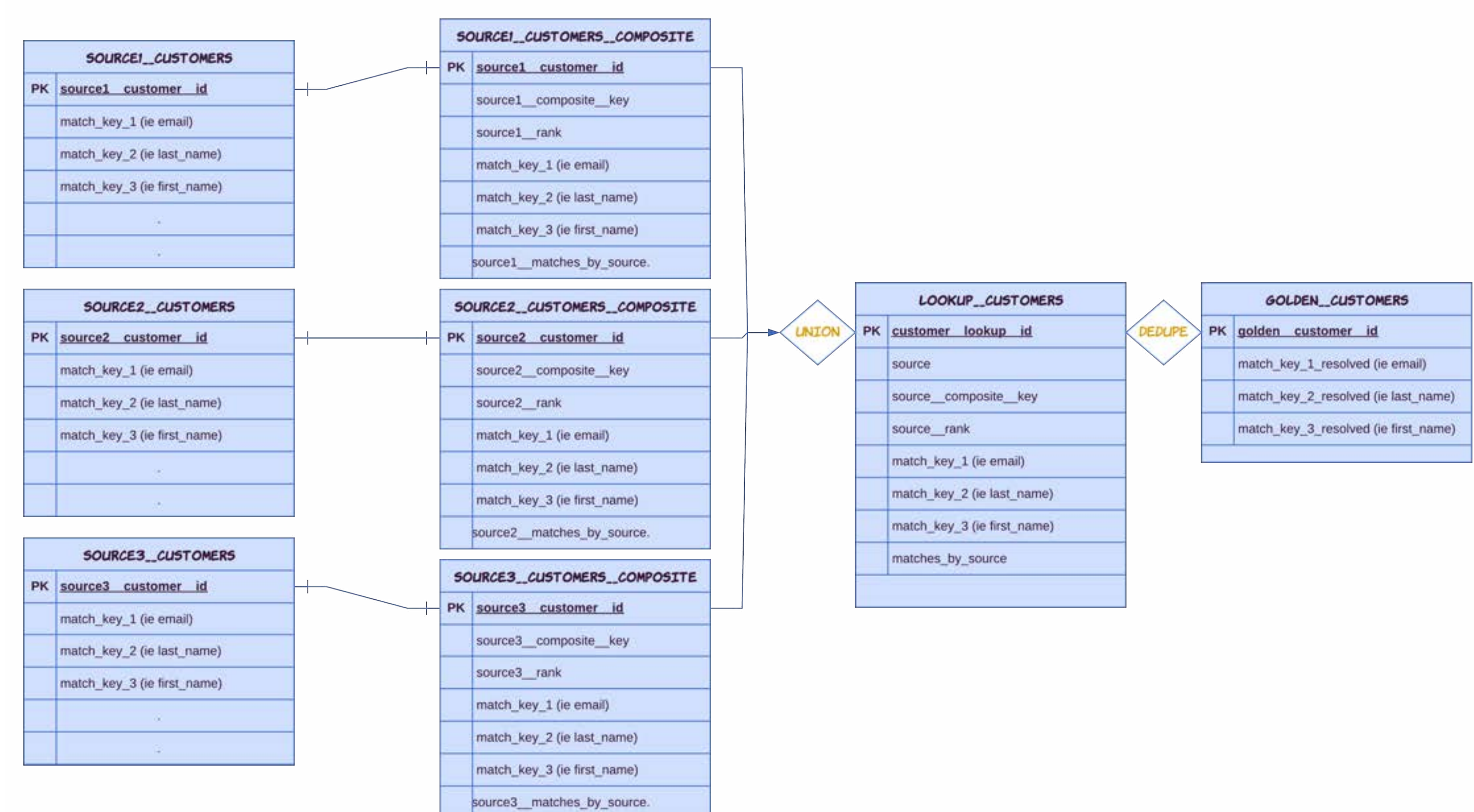
Now we'll need to resolve identities across sources because we should have some overlap across sources where there are records with the same composite key from different sources where the source rank indicates it is the newest or best from each source. Here we can define a **precedence matrix** to determine for each match key and other fields, the order of sources that we trust for that field. For example, data source A may be the best source for the email field, then data source C, then data source B. But for the phone number field, the order may be different.

5. Assign Customer ID:

The final step is to simply take records that have the same match keys and generate a unique customer identifier for that matching group of customer records. Every customer ID generated can be used to link the customer sources together going forward. We have now resolved across sources to create a Golden Record for each unique customer in the data warehouse.

With composite key matching, it's important to decide whether or not you'd like to persist IDs over time, even if an underlying match field is updated in a given source.

An example would be if a Salesforce Contact changes its email address in Salesforce, since the Salesforce ID did not change for that Contact, do you want to consider this the same person in the data warehouse or consider this a different person? We've seen it implemented both ways depending on the business rules.



03. How Identity Resolution Works

Types of Matching (cont.)

Probabilistic Matching

This approach uses probabilities to score pairs of records on how alike they are. You then set a threshold for this score, based on the scores of some examples which you already know do or do not represent the same individual. If a pair of records score at or above the threshold, they are considered to represent the same individual; if they score below, they are considered unique. There are many scoring methods for Probabilistic Matching used in Identity Resolution. Here are some popular choices:

Fuzzy String Matching

- **Levenshtein Distance Algorithm:** roughly based on the number of edits needed to match strings
- **Hamming Distance Algorithm:** roughly based on the number of characters that are different across strings, based on position
- **Affine Gap Distance Algorithm:** similar to Hamming but deletions or insertions are given a bit of a break
- **Jaro-Winkler Distance Algorithm:** roughly the number of characters that appear within a similar location across strings, discounted for characters that are out-of-order

Phonetic Matching

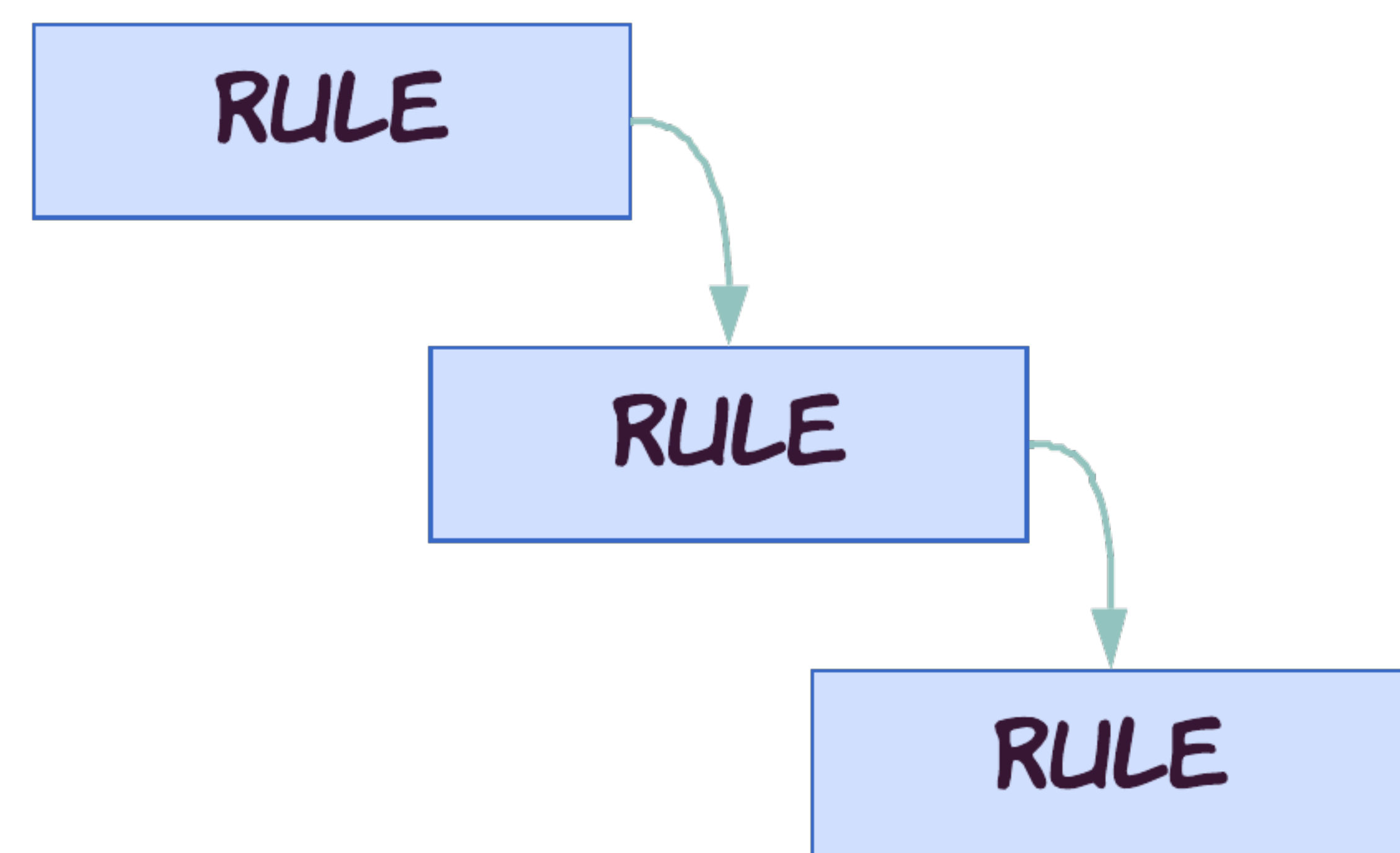
- Metaphone Algorithm
- Soundex

03. How Identity Resolution Works

Types of Matching (cont.)

Cascading Deterministic Heuristic Matching (Cascading Rules)

Cascading Heuristics or Cascading Rules is a type of Identity Resolution Algorithm where you apply different rules or sub-algorithms in order of most strict to least strict. An example might look something like this:



1. Match records exactly on email, last name, date of birth

2. For all records that did not match:

- Attempt to match them to the previously matched records on email, and the first three letters of the last name
- Attempt to match them to other remaining unmatched records on email and the first three letters of the last name

3. For all records that still did not match:

- Attempt to match them to the previously matched records on email address only
- Attempt to match them to other remaining unmatched records on email address only

03. How Identity Resolution Works

Types of Matching (cont.)

Advanced Machine Learning Matching

While probabilistic matching could be considered a type of basic type machine learning matching, there are many other more advanced machine learning approaches that can be used for identity resolution as well. This is probabilistic matching on steroids.

Think clustering algorithms and deep neural nets giving you answers that are often uninterpretable. This makes auditing challenging and can seem like a black box to your business stakeholders. With the Transformer example shown above, you can effectively train a neural network to give similar results to what a human would give, as long as humans are labeling the training data.

A key question when thinking about this approach is do you have a few humans on your team that you would have faith in to decide if records should match? Is there anyone at your organization that you trust that much to put the organization's credibility on the line? If those folks exist, those are the folks you should have labeling your training data and auditing the inferencing pipeline from time to time. If those folks do not exist, this is probably not a great solution for you.

Cascading Mixed Heuristic Matching (including Probabilistic)

Cascading Mixed Heuristics is a type of Identity Resolution Algorithm where you apply different rules or sub-algorithms in order of strictest to least strict that can also include probabilistic matching. Here's an example using the previous example of cascading deterministic matching but adding a final probabilistic matching step:

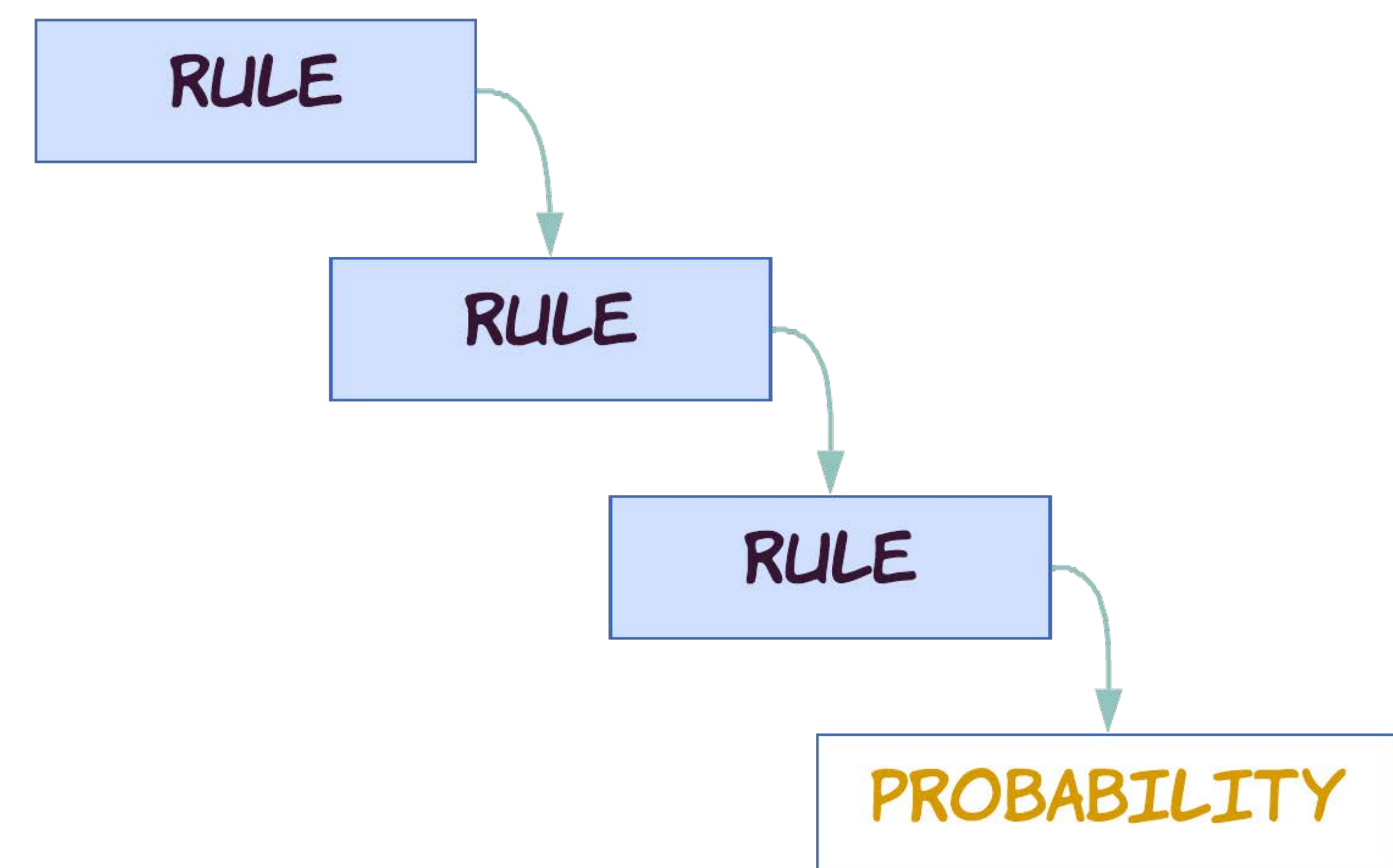
1. Match records exactly on email, last name, date of birth
2. For all records that did not match:
 - Attempt to match them to the previously matched records on email, and the first three letters of the last name
 - Attempt to match them to other remaining unmatched records on email and the first three letters of the last name

03. How Identity Resolution Works

Types of Matching (cont.)

Cascading Mixed Heuristic Matching (including Probabilistic)

Cascading Mixed Heuristics is a type of Identity Resolution Algorithm where you apply different rules or sub-algorithms in order of strictest to least strict that can also include probabilistic matching. Here's an example using the previous example of cascading deterministic matching but adding a final probabilistic matching step:



1. Match records exactly on email, last name, date of birth
2. For all records that did not match:
 - Attempt to match them to the previously matched records on email, and the first three letters of the last name
 - Attempt to match them to other remaining unmatched records on email and the first three letters of the last name
3. For all records that still did not match:
 - Attempt to match them to the previously matched records on email address only
 - Attempt to match them to other remaining unmatched records on email address only

4. For all records that did not match:
 - Attempt to match them to the previously matched records on phonetically alike email strings (**probabilistic matching**)
 - Attempt to match them to other remaining unmatched records on phonetically alike email strings (**probabilistic matching**)

04. Choosing a Matching Algorithm

False Positives & False Negatives

In the context of identity resolution, we can think of a **false positive** being a match that shouldn't have happened.

For example, combining the data of two people that are not the same person. Imagine an example in the financial industry where a false positive could mean a bank incorrectly tells someone that they have overdrafted their account or, in the healthcare industry, disclosing HIPAA-sensitive information to the wrong patient. This could be catastrophic for the customer and for your company.

On the flip side, a **false negative** is missing a match that should have been made. This can be costly in the world of Customer Service where vital information might not get to a customer. Usually in the world of sales and marketing, this means a small percentage of revenue is left on the table because of a smaller targeting audience for a given campaign.

Outside of high stakes, critical customers service use cases, false positives are typically much worse than false negatives. Let's think about our matching algorithms and their propensity for false positives.

For deterministic options like single field matching, composite field matching, and cascading deterministic heuristics, the false positive rate is typically low. The rate is closely correlated to the quality of your data and is no worse. If you have up-to-date information about your customers with strong validation at the points where your customers are communicating with you, deterministic options will give you the lowest rate of false positives, so this is generally a good approach.

For probabilistic options like fuzzy/phonetic string matching, advanced ML, and cascading mixed heuristics, the false positive rate is typically higher. This is because the rate depends on implementation and on threshold selection (in addition to the quality of your data). Here, you start with the same baseline as the deterministic options - the quality and freshness of your data - and can be further negatively affected by setting loose thresholds on the scores that determine matches. Typically probabilistic options give a higher rate of false positives and are generally a worse approach for this decision criteria.

04. Choosing a Matching Algorithm

Cost

Let's take cost into account for a moment. That fancy GPT-3 matching example above cost ~\$0.01 using OpenAI's API... for a single call. On its own, a penny may not sound like much but consider a database of 100,000 customers.

Say we have records for these 100,000 customers across three different sources - product_users, salesforce_contact, marketo_leads. A naive implementation of Identity Resolution using GPT-3 might look something like this:

1. Iterate through each customer record in the product_users table (100,000)
2. For each record, compare that record to each record in the salesforce_contact table using GPT-3 to match (100,000)
3. Iterate through all resulting matched and unmatched records (100,000-200,000)
4. For each record, compare that record to each record in the marketo_leads table using GPT-3 to match (100,000)

At \$0.01 per API call, you're looking at a cool \$200 million to \$300 million for your little identity resolution project. This is without considering deduping and merging within each source. And with a relatively small number of customers at 100,000. Ouch.

There are techniques to avoid those naively exhaustive loops. A popular approach is called blocking. Blocking is all about limiting the size of the chunk of data you use for comparison, without sacrificing much in terms of accuracy. One simple form of blocking looks a little like this:

1. Sort all of your sources by the field(s) you want to use for matching
2. Choose a record from Source A. Let's call it record R
3. Only compare that record to records in Source B and C that are +/- 1000 records from the position record R would be in this source, if it were located in the proper spot in the sort order.

04. Choosing a Matching Algorithm

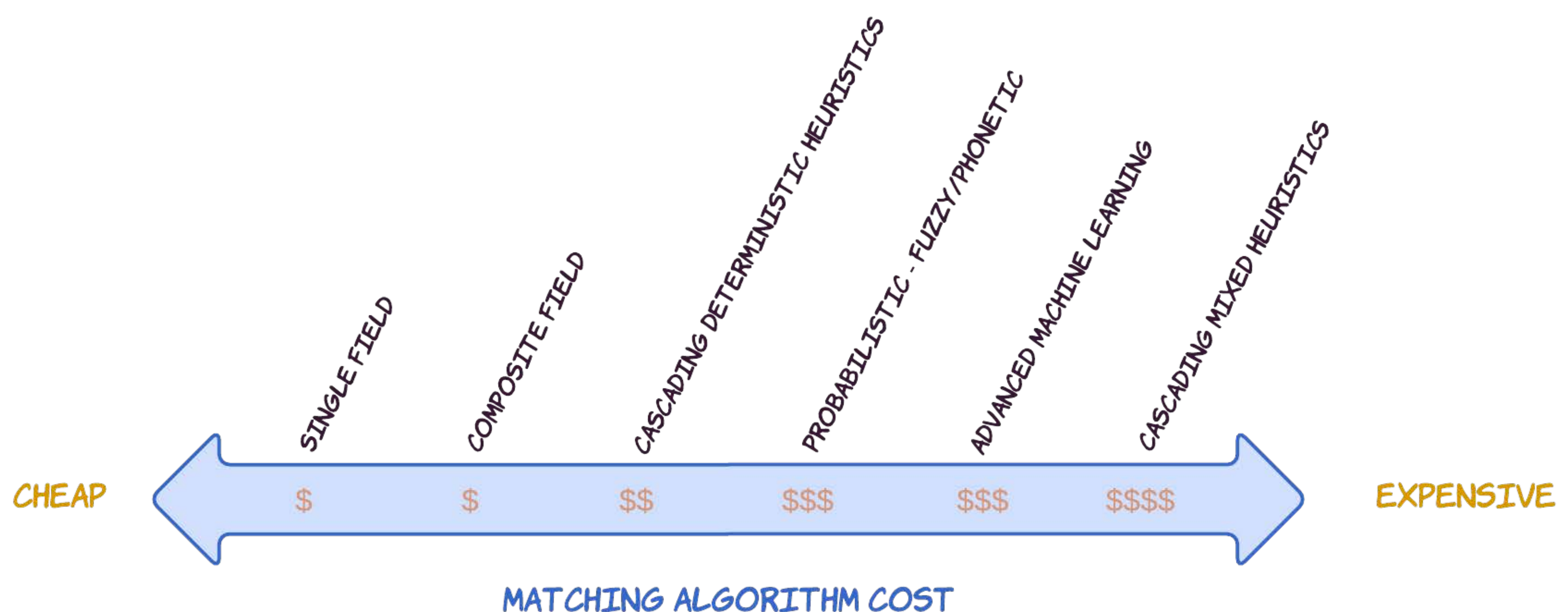
Cost (cont.)

Blocking can help drastically reduce the cost of advanced techniques, but they are still very expensive compared to the alternative.

For some of the simpler approaches, you can complete all or nearly all of the processing using native SQL queries on the data, in the data warehouse or data lake where it exists.

There has been very healthy competition across cloud data warehouse solutions that has driven the cost of SQL-based compute down and makes SQL-based approaches to Identity Resolution attractive from a cost perspective.

Here's a rough sketch of what the cost comparison could look like across different matching strategies:



04. Choosing a Matching Algorithm

In-Place vs In-Memory Processing

In addition to cost benefits from in-place processing in the data warehouse or data lake using SQL, this is also a very data-governance and data-transparency-friendly way of managing Identity Resolution.

Data Governance

In-place processing is a governance-friendly approach because the data never leaves the data warehouse for Identity Resolution with this approach.

There is less potential for copies of customer data being stored elsewhere when you process the data using the native SQL engines of the data warehouse or data lake than when you do what I'll refer to as in-memory processing.

The data warehouse engines use memory as well but we'll use in-memory here to mean taking the data out of the data warehouse or data lake into some other memory and doing some processing.

An example of this would be running some python identity resolution code on a Kubernetes cluster. Say you're using one of the cascading approaches and you decide to store a file in a blobstore like an S3 or Google Cloud Storage bucket of the results from the current cascade step, for reference in a later cascade step. This can be a governance problem if you're not careful about managing the blobstore.

Under GDPR and CCPA, a customer has the right to have their data deleted within 30-45 days of requesting deletion. Managing these requests is challenging even when the problem is scoped to only the data in the data warehouse.

If you have files floating around in S3 that could contain customer data, you need to monitor and search those as well when these delete requests (sometimes called "sanitize requests") come in.

04. Choosing a Matching Algorithm

In-Place vs In-Memory Processing (cont.)

A good principle is whenever possible keep the data in the data warehouse or data lake, to scope the problem of managing these requests to one place. If you do have buckets with files floating around, a good practice is to have time-to-live limits on the buckets of 14 days or less, so they are automatically deleted within the 30-45 day time constraints.

Data Transparency

There are some third-party services like Customer Data Platforms (CDPs) that offer to do the Identity Resolution work for you if you send the data to their SaaS platform to be processed.

There are a few challenges with this approach but something we hear often is how this can be a “black box” that isn’t transparent. Organizations often do not have visibility into how the Identity Resolution is happening, which can lead to a lack of faith in the process, non-portability, and vendor lock-in - even if you no longer require other services that the CDP provides.

Putting it All Together - Choosing a Matching Algorithm

When evaluating matching algorithm options, two choices stand out as best for most of the hundreds of organizations I’ve spoken with:

Composite Field Matching and Cascading Deterministic Heuristic Matching

These both provide low false positive rates, relatively low costs, and can be implemented to process data in-place, natively using SQL. Tradeoffs between the two are the simplicity to implement and cost - with Composite Field Matching providing a simpler implementation and lower cost at the expense of a typically higher false negative rate (more opportunity left on the table for campaign targeting).

IN-PLACE	IN-MEMORY
<i>SINGLE FIELD</i>	<i>PROBABILISTIC FUZZY/PHONETIC</i>
<i>COMPOSITE FIELD</i>	<i>ADVANCED MACHINE LEARNING</i>
<i>CASCADING DETERMINISTIC HEURISTICS</i>	<i>CASCADING MIXED HEURISTICS</i>

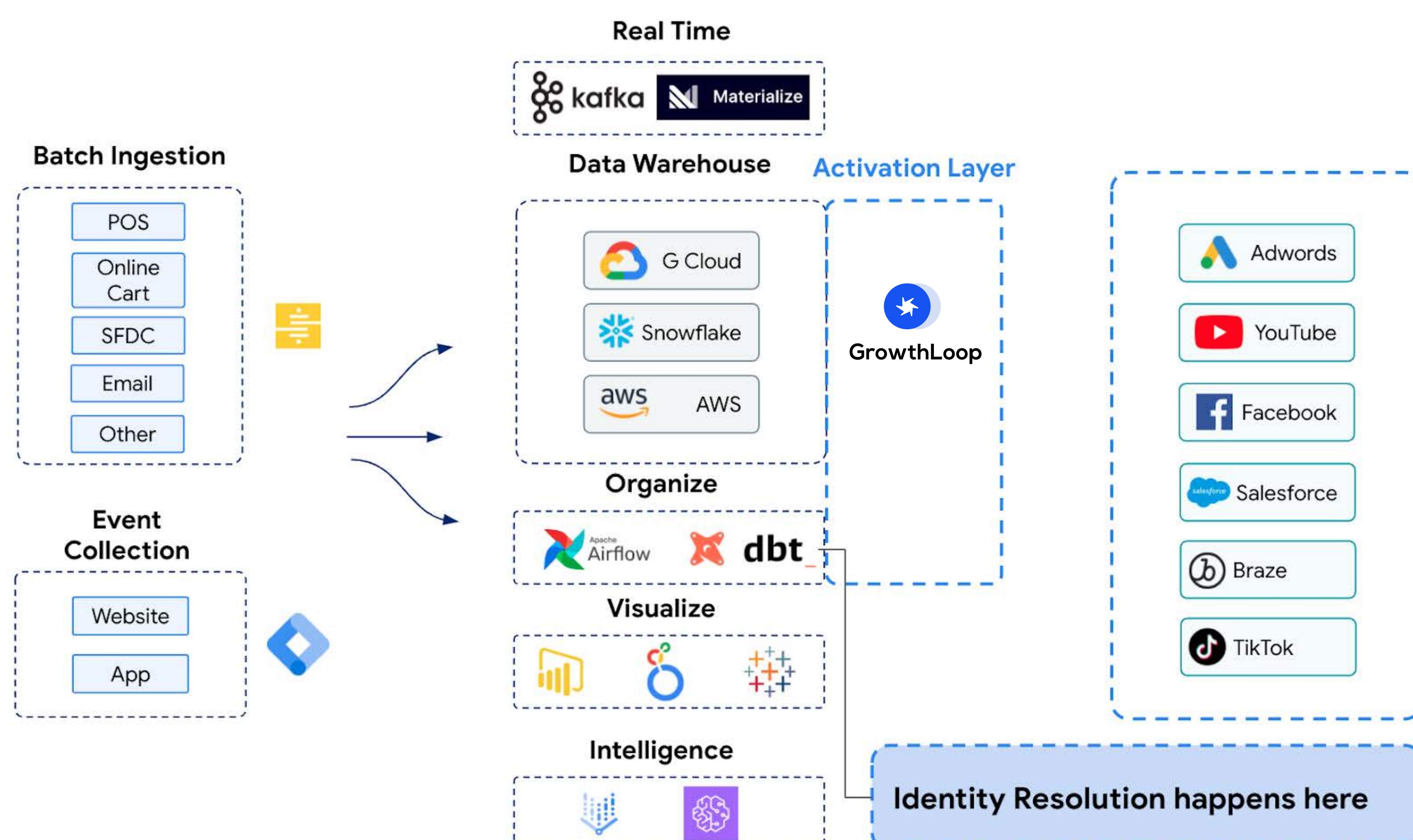
05. Bottom line: Handle Identity Resolution in the Data Warehouse

In the Modern Customer Data Stack, Identity Resolution is implemented on top of the Data Warehouse. Most often, we see this being implemented in open source tools like dbt for declarative SQL and orchestrated via Airflow. This is the most accurate and efficient way to ensure that Identity Resolution is taking into account all of the data you have available for each customer.

There is a lot of work happening in the industry today around Identity Resolution. Many providers will offer one-size-fits-all solutions with black box implementations that may not be right for your business.

We also understand that Identity Resolution can be daunting, and the thought of implementing this yourself may be something you do not have the bandwidth for on your team today. We think good, transparent Identity Resolution should be accessible for everyone since it makes communication between customers and organizations less prone to fraud or annoyance, more valuable for both parties, and simply better.

With that in mind, we've built an entire product called Blueprint that simplifies Identity Resolution into an easy-to-use config-driven framework. Blueprint allows you and your team to focus on the business-level decisions defining your matching rules for Identity Resolution, like the ones covered in this article, and automatically implements the approach you define on your Data Warehouse.

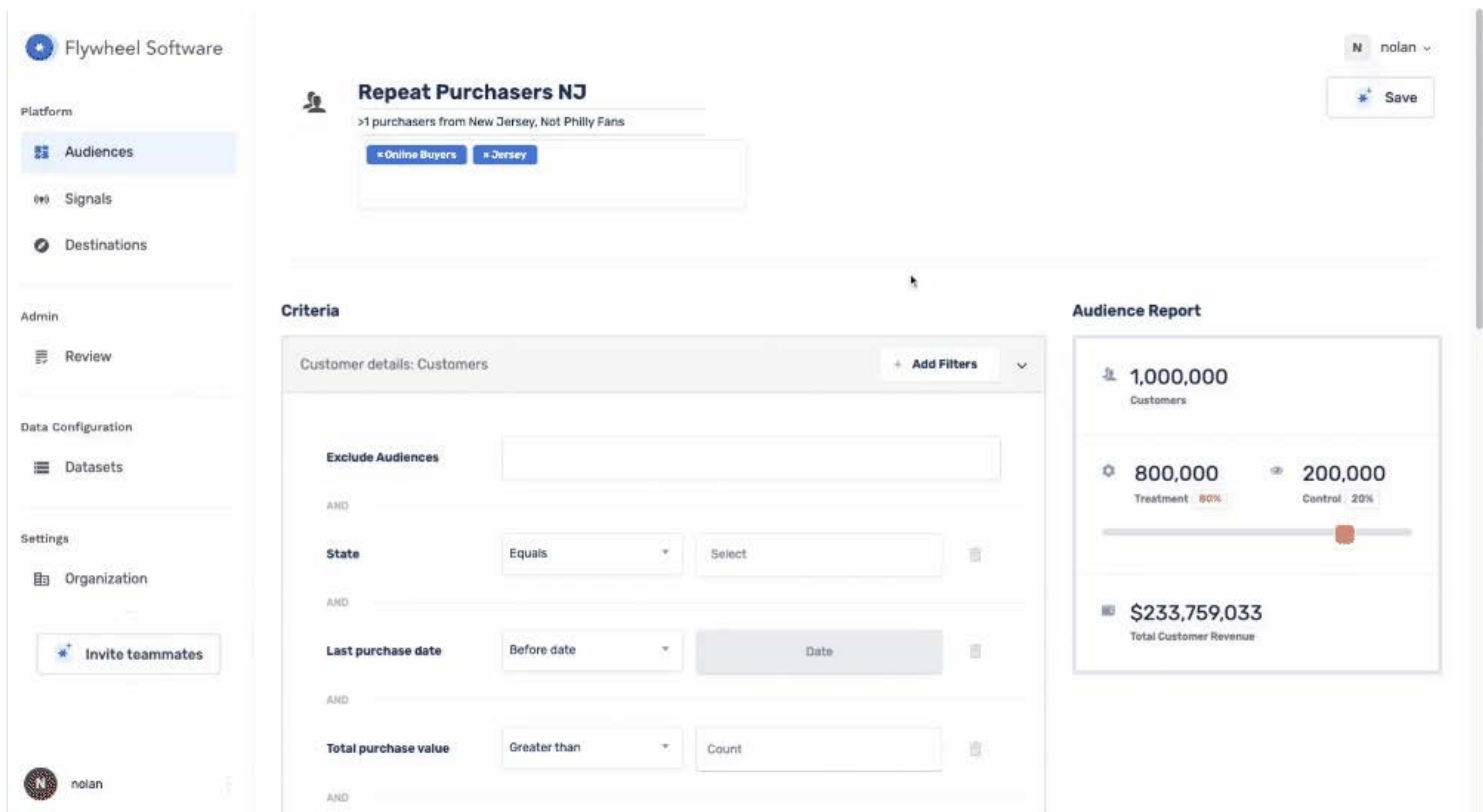


06. Ready to get started?

Once identities are resolved and the Customer Data Model is completed you'll want to activate that data to channels where the business can use it. That's why we built GrowthLoop.

If you want to hear more about how this approach can help your organization, check out our Modern Customer Data Stack blog post or reach out to me directly: anthony@growthloop.com

We've created a thin software layer that is fast and simple to set up to allow business users to take advantage of all of your customer data with an easy-to-use, no-code interface that queries the data in place. No pipeline and schema setup, no integrations, just connect and go.



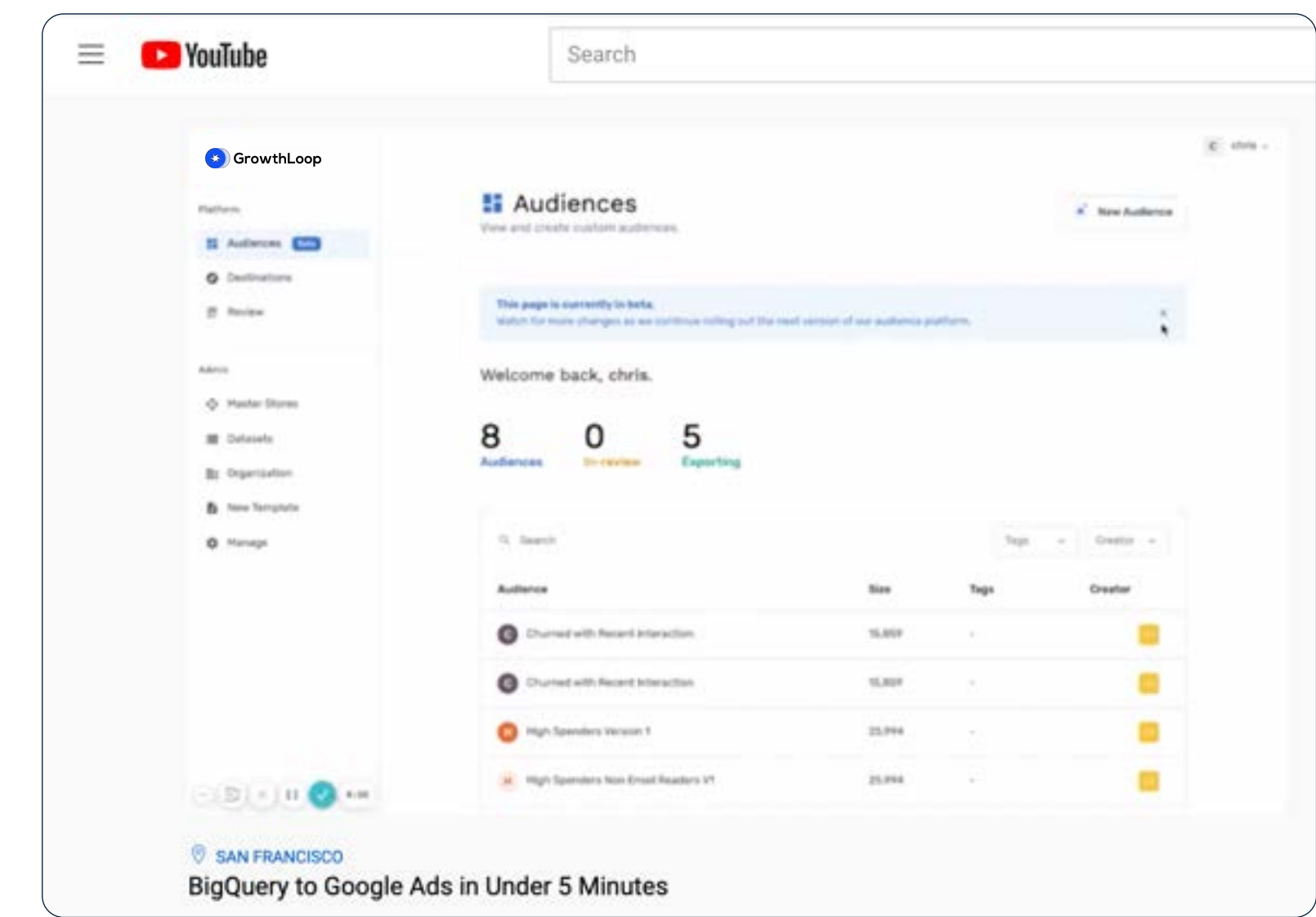
The screenshot displays the GrowthLoop interface. On the left is a sidebar with navigation options: Platform (Audiences, Signals, Destinations), Admin (Review), Data Configuration (Datasets), and Settings (Organization, Invite teammates). The main area is titled 'Repeat Purchasers NJ' with a subtitle '>1 purchasers from New Jersey, Not Philly Fans'. Below this are filters for 'Online Buyers' and 'Jersey'. The 'Criteria' section shows a list of filters: 'Exclude Audiences', 'State' (Equals, Select), 'Last purchase date' (Before date, Date), and 'Total purchase value' (Greater than, Count). The 'Audience Report' section shows a total of 1,000,000 Customers, with 800,000 in the Treatment group (80%) and 200,000 in the Control group (20%). The total customer revenue is \$233,759,033.

Contact us

Email: solutions@growthloop.com

Schedule demo: GrowthLoop.com/contact-us

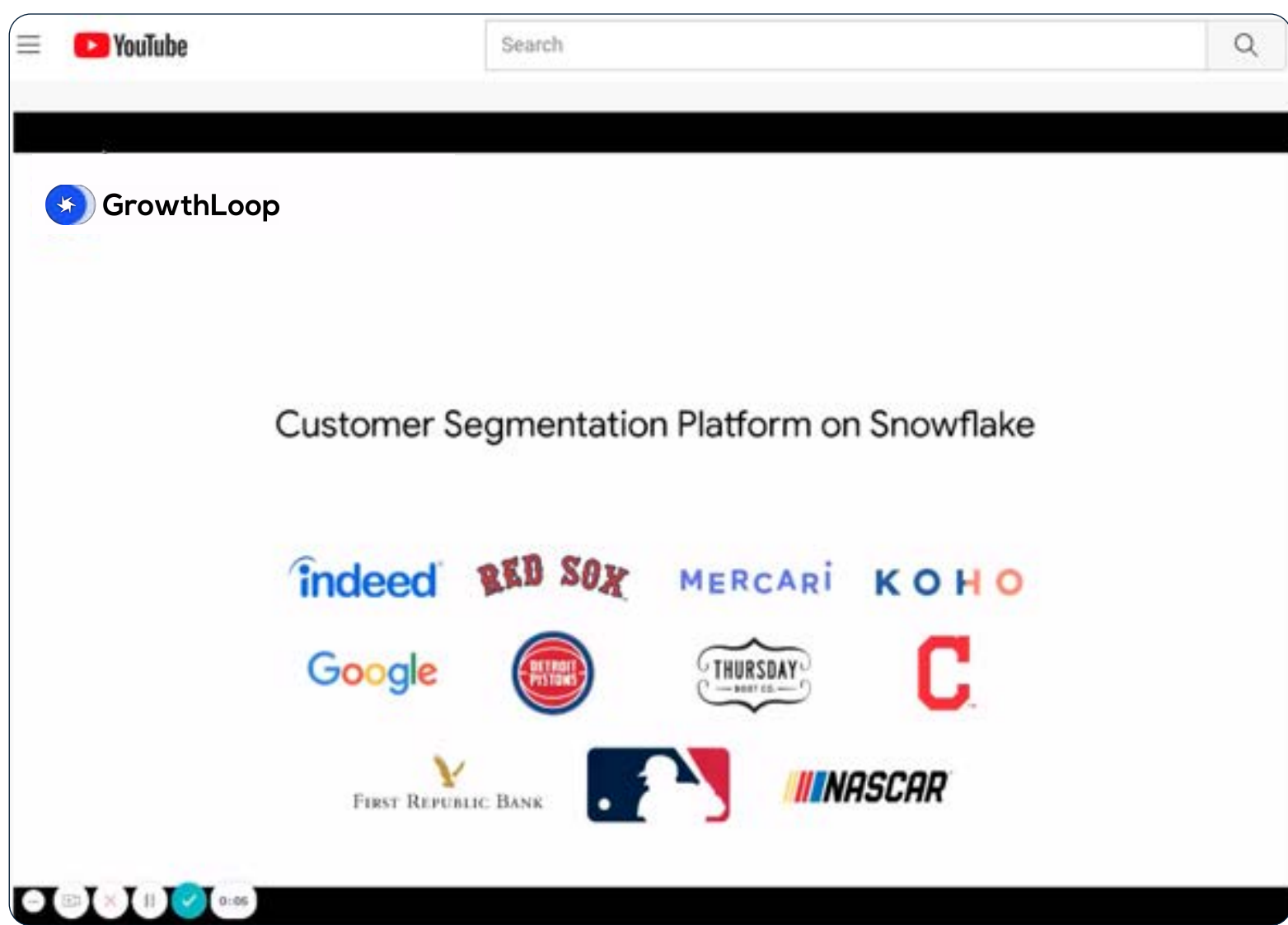
Linkedin: [@growthloop-co](https://www.linkedin.com/company/@growthloop-co)



On-Demand BigQuery Demo

Learn how you can activate customer data on Google BigQuery to any destination in less than 5 minutes.

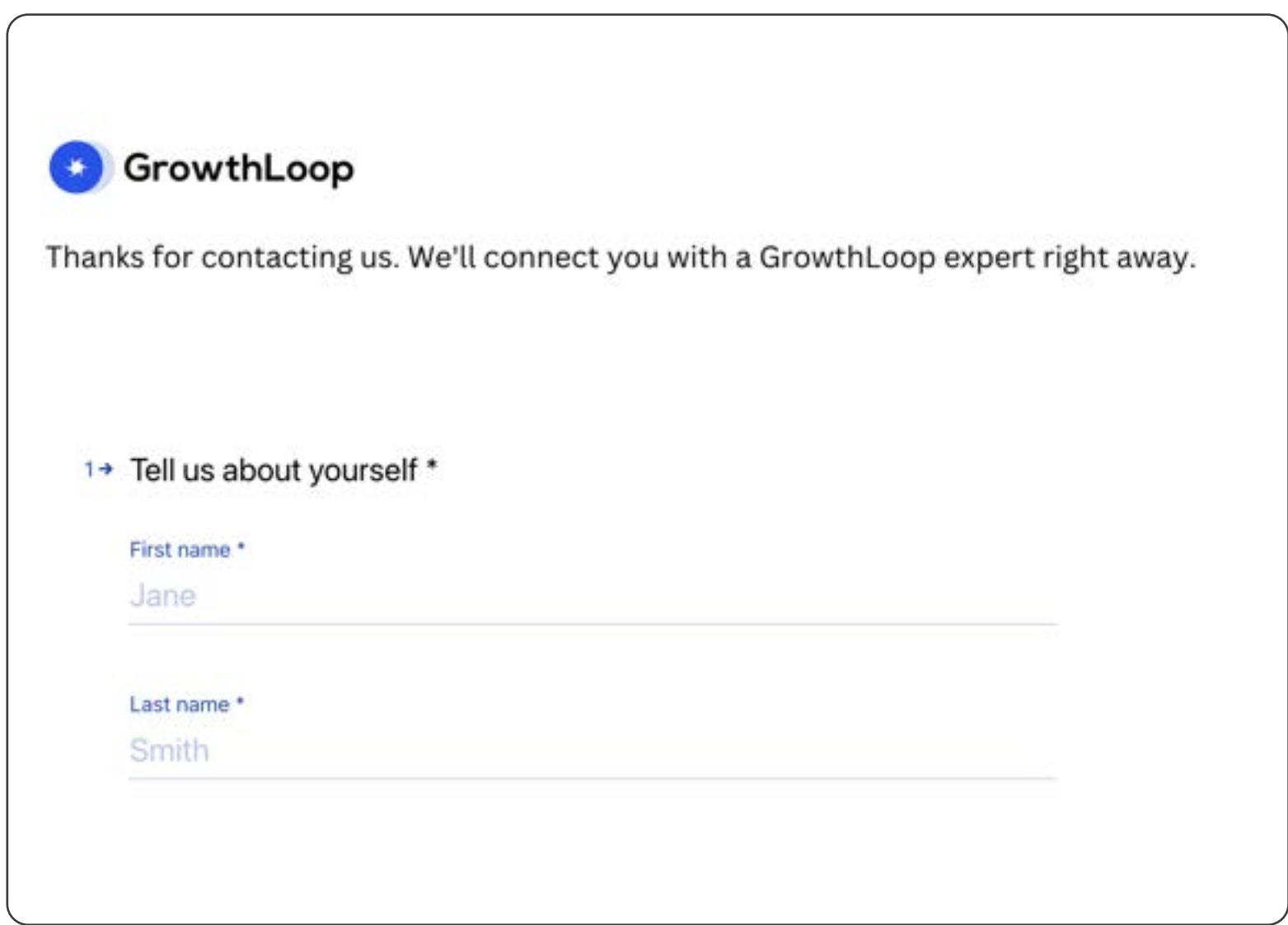
Play Now



On-Demand Snowflake Demo

Learn how you can activate customer data on Snowflake to any destination in less than 5 minutes.

Play Now



Personalized 1:1 Demo

Get a tailored walkthrough and learn how GrowthLoop can activate the customer data in your warehouse.

Request a Demo