IMMUTA

CONTENT BUNDLE

The Databricks Security & Access Handbook

IMMUTA + databricks
E-BOOK
A Guide to Automated Data Security
In Databricks Using Immuta
orce
Code
J.B. HUNT
ls
t
n
e

Table of Contents

Introduction

With an explosion of data, users, and policies, organizations can no longer who has access to what data. They either over or under share data, and lack the necessary tools for proactive monitoring. As a result, they struggle to operationalize data lakehouse platforms, get value from their data and cloud investments, and adapt to evolving risks, which could result in losses of revenue, innovation, and customer trust.

Immuta's integration with Databricks was built to mitigate these challenges so users can maximize the power of their data lakehouses. By discovering, securing, and monitoring data activity to detect threats, Immuta simplifies operations, improves data security, and unlocks more value across Databricks workloads.

We compiled this content bundle to highlight just a few of the things you can do with the combined power of Databricks and Immuta. First, you'll dive into an end-to-end look at how Immuta enables discovery, security, and monitoring for Databricks. Then you'll go a step further, and see how to apply policy-as-code on Databricks data sets Finally, you'll get a glimpse into JB Hunt's journey with Immuta and Databricks, including how its data team was able to increase permitted use cases by 100%.

IMMUTA + databricks

E-BOOK

A Guide to Automated Data Security

In Databricks Using Immuta

Table of Contents

Introduction

Data-driven insights are no longer optional — they're necessary. Organizations that leverage data to make critical business decisions have a definitive competitive advantage. While consumers today appreciate and often pay more for customized products and services, delivering that experience often requires the use of sensitive personal data.

Sensitive data is the most valuable asset for analytics and data science. But if data engineers and architects can't proactively secure this resource while maximizing its utility for real-time access by analysts and data scientists, is it still as valuable?

Centralizing data and making it discoverable and actionable are key to data-driven innovation, personalized user experiences, and revenue growth. But the advent of new data privacy regulations like California's CCPA and Europe's GDPR, in addition to increasingly stringent internal data rules and security policies, have put a tax on data engineering teams, who are charged with translating these rules and regulations into executable policies so that data consumers can gain access to critical data for innovation. This complicated and time-consuming process — which can lead to personal liability if a data leak or beach occurs — can delay or halt advanced analytics and data science projects, which require fast access to data.

To maintain the balance between data security and speed to data, Immuta integrates with Databricks to enable data teams to secure and scale lakehouse access by automating data discovery and classification, providing comprehensive secure protection, and continuous data monitoring. This white paper outlines those capabilities and considerations for when each should be used, so that data teams can unlock even more use cases with Databricks and Immuta.

Why might data teams need Immuta for Databricks?

- They're responsible for managing complex policies across many tables — as well as the ensuing role explosion.
- They need to empower stakeholders with business context of data use for self-service data access management.
- They're unable to easily prove compliant data use with corporate rules and regulations, or respond to investigations.
- They're unable to implement global policies and access controls for each platform in their cloud data ecosystem, so must manually do it for each individual platform.
- They're unable to dynamically restrict access based on time, geography, purpose, data sharing agreements, or other scenarios that may arise.
- They're responsible for consistent security and auditing across multiple data platform technologies, but don't have a centralized way to manage the process or ensure its uniformity, which causes confusion and frustration.
- They're expected to stay up-to-date on evolving regulations and to implement sufficient data security measures accordingly — otherwise they could be held personally liable for leaks or breaches.
- They spend their time managing case-by-case data access within an organization, as opposed to delivering new innovations.
- They need to be able to mask data while preserving its original format for non-production use.
- They're expected to enable compliant, secure collaboration on data sets without inadvertently granting unauthorized access or hindering analytics initiatives.
- Their policies are difficult to scale because they're unable to draw upon existing organizational glossaries or underlying metadata.

Automated Data Discovery & Classification

What is it?

Manual processes are simply not practical for an increasingly cloud-based and decentralized data ecosystem. Users rely on Databricks — in addition to their other cloud platforms — for fast, automated data storage and compute. Why should data teams expect any less from their data security solution?

With Immuta and Databricks, automated data security and privacy controls are enforced using dynamic attribute-based access controls to serve as additional buffers against unauthorized access, data leaks, and re-identification. But before these controls can even be enforced, data first needs to be discovered and classified appropriately. Only after data discovery is completed and sensitive data is correctly tagged can security policies be built to effectively protect it.

Sensitive Data Discovery & Tagging

Manually tagging sensitive data as it is uploaded to Databricks is a significant time commitment for data teams, in addition to introducing the risk of human error. As data becomes available faster than ever before, it's easy to see how this process can quickly become unmanageable for data engineers and architects, who could also potentially be held liable for any sensitive data that slips through the cracks and ends up in the wrong hands.

Immuta leverages Databricks Unity Catalog APIs to monitor for schema and user changes, and enriches user metadata by discovering and tagging sensitive data, inferring additional information like PII, tagging data with external information, and leveraging Unity lineage for tag propagation. This allows Databricks users to discover the sensitive data that is entered into their data ecosystem without requiring manual effort. New data sources are scanned and classified using a combination of more than sixty prebuilt classifiers and any custom or domain-specific classifiers.

Data identified as being sensitive can be assigned tags that correspond to access control policies which are dynamically applied at query time. Immuta automatically tags sensitive fields, like PII, personal data, or PHI, in addition to enabling data engineers and architects to create tags as needed. Tags can also map to data privacy protection laws such as CCPA and GDPR, which in turn correspond to Immuta's templated starter policies that automate policy creation and ensure compliance with regulations. This means data engineers and architects can seamlessly ingest sensitive data and apply regulatory-compliant policies without fear of personal liability or wasted time.

What problem does it solve for data teams?

According to Immuta's report on Data Engineering & Operations for Analytics, the most challenging aspect of managing a data pipeline for data teams is data masking and security, followed by auditing and monitoring. The least challenging part? Extract and load.

Maintaining an ETL pipeline and GRANTs can be complicated and cumbersome for data teams when data discovery, classification, and governance aren't built into the process. This is because data engineers and architects must transform raw data into "clean" data before it can be utilized by analysts, data scientists, or any other data consumers. The report found that this task overwhelmingly falls on the shoulders of data engineers — more than half of survey respondents said transform is done by data engineers, either before or after load. So, in order to make data operational, data engineers must apply comprehensive security and privacy controls as part of transform. This model also sets data engineers and architects up for having to make copies of data and apply controls to them manually.

Without an integrated system that proactively discovers and classifies sensitive data, these time-intensive, manual processes may or may not result in sufficiently protected information.

Immuta's automated security and privacy controls make this task much faster and more secure for Databricks users. Incorporating automated data discovery and classification into the ETL process with Immuta's native Databricks integration eases the burden of combing through and identifying potentially sensitive data, and streamlines the process of updating pipelines and GRANTs as data users and policies change over time.

When data consumers run a query in Databricks, Immuta's dynamic controls are applied at run time based on how data has been discovered and tagged. This captures the most current data policies and permissions, simplifies the ETL process, and eliminates the need for manual processes and data copies. Dynamic, automated controls that reduce the risk of human error, missed sensitive data, and re-identification are a more efficient, secure approach for data teams.

How is it implemented?

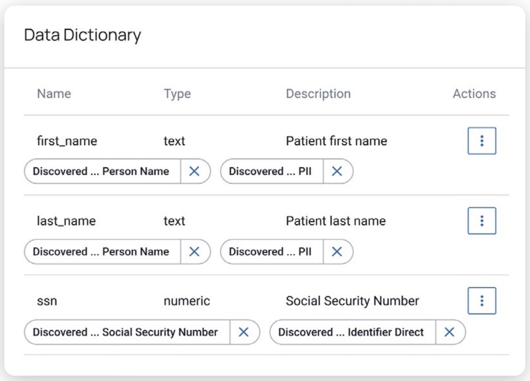
To understand the benefits of data discovery and classification with Databricks and Immuta in practice, use medical records as an example.

In this scenario, a data engineer receives a data set of medical records and is responsible for preparing it for use by a medical center's billing department and the city's public health department. The data set contains protected health information (PHI) including name, address, insurance information, and social security number, and the data engineer must implement policies and privacy controls that restrict data visibility by user attribute and purpose, while achieving HIPAA compliance.

As the data set is uploaded to Databricks, Immuta's sensitive data discovery tags and classifies PHI so the data engineer can assign it to access control policies. The data engineer can then build policies in plain language that restrict data returned from a query based on user attributes. So, an accountant from the billing department may be able to view patients' full addresses and insurance information in order to process claims, but the data engineer can suppress diagnoses with a value of `redacted`.

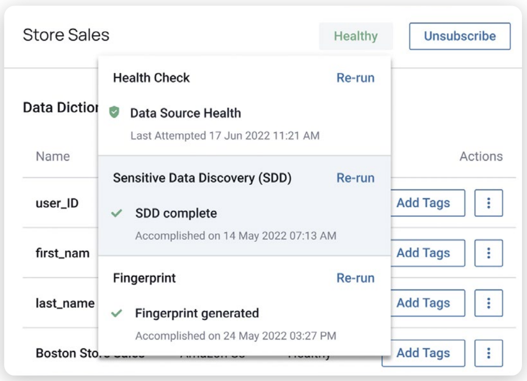
Meanwhile, an analyst from the public health department may be able to view diagnoses, but the data engineer suppressed patients' names, social security numbers, and insurance information, and generalized addresses by only showing the first three numbers of a zip code — for example, `021XX`. This would allow the analyst to see diagnoses by general area, so they can make public health recommendations, without identifying any particular individuals with a positive diagnosis.

Using automated controls based on classified and tagged data, the data engineer in this scenario preserves the original data set within Databricks — without making copies — and ensures data consumers see only the information relevant to their functional need.



Data Dictionary

Name	Type	Description	Actions
first_name	text	Patient first name	⋮
Discovered ... Person Name ✕	Discovered ... PII ✕		
last_name	text	Patient last name	⋮
Discovered ... Person Name ✕	Discovered ... PII ✕		
ssn	numeric	Social Security Number	⋮
Discovered ... Social Security Number ✕	Discovered ... Identifier Direct ✕		



Store Sales

Health Check — Re-run

Data Source Health
Last Attempted 17 Jun 2022 11:21 AM

Sensitive Data Discovery (SDD) — Re-run

✓ SDD complete
Accomplished on 14 May 2022 07:13 AM

Fingerprint — Re-run

✓ Fingerprint generated
Accomplished on 24 May 2022 03:27 PM

Dynamic Data Security & Access Controls

What are they?

As organizations continue to invest resources into migrating data to the cloud with powerful platforms like Databricks, they face an array of common challenges. They desire fast and easy data access for their varied — and likely growing — pool of data users, but need to ensure that this data is effectively and consistently secured against leak or breach.

Securing data in modern cloud data ecosystems can be an extremely complex process. With more data sources, users, and platforms introduced into the data stack, it becomes difficult to create policies that protect sensitive data wherever it lives and moves. On top of this, further decentralization can add increased risk of data breaches and leaks. This creates an ultimatum: companies can either halt the breakneck progress of their data use by aggressively locking it down, or leave it completely exposed to the dire effects of misuse.

Simply safeguarding access to raw data inputs is no longer sufficient to protect personal data. Data teams must also consider what information can be inferred about an individual from a model's behavior or API output, as well as any risks that may arise from publishing a data set. Dynamic data security and access controls protect data in a manner that maximizes privacy and utility, enabling both internal or external data use with a reduced possibility of attacks on the privacy of individuals. Databricks users can implement a range of dynamic security and privacy controls through Immuta:

Attribute-Based Access Control (ABAC)

Immuta's data access and security policies are built and applied based on user attributes. Attribute-based access control (ABAC) is an approach to data security that permits or restricts data access based on assigned user, object, action, and environmental attributes. In contrast to role-based access control (RBAC), which relies on the privileges specific to one role for data protection, ABAC has multiple dimensions on which to apply access controls. This makes attribute-based access control a highly dynamic model because policies, users, and objects can be provisioned independently, and policies make access control decisions when the data is requested.

Similar to how Databricks separates compute and storage, Immuta separates access control policy logic from the data platform. Within Databricks, Immuta extracts access control policy logic from both compute and storage, eliminating the need for data engineers and architects to spend time and energy recreating access control policies for each individual data tool. The flexibility to utilize the same access control policies — whether role-, attribute-, or purpose-based — consistently across distributed data storage and compute providers is a powerful capability, particularly in satisfying data use compliance, changing business rules, or working within a cloud data ecosystem that includes Databricks.

Data Masking

Data masking is the process of removing or obscuring identifiers, altering existing sensitive information in a data set to make a fake–but still convincing–version of it. This allows sensitive data to be stored and accessed, while maintaining the anonymity and safety of the information involved. This type of privacy-enhancing measure is crucial for organizations looking to secure their most sensitive data, and can facilitate measures like secure internal and external data sharing and protecting sensitive personal health information (PHI) and financial data.

Masking can be applied using a range of techniques, most commonly by way of generalization or suppression. Generalization assigns the same broad value for any given attribute — for example, replacing an attribute, `hair color`, with the value any. Suppression works the opposite way, by removing values entirely or replacing them with a constant — for instance, replacing an attribute, `hair color`, with the value `redacted`.

Differential Privacy

The sheer amount of personal information being collected and used in today's environment means that no single piece of data exists in a vacuum. Differential privacy aims to mathematically limit an outsider's ability to confidently use the output of an analysis to make inferences about its input. This allows individuals providing their personal data to credibly deny their participation in the input.

Differential privacy requires the data analysis mechanism to give the same answers with similar probabilities over any pair of databases that differ by a single row. In other words, Immuta injects noise into the data analysis in order to render inference attacks nearly impossible. This way, an individual may claim that the output of the mechanism came from a database that did not include their data.

Randomized Response

While differential privacy enables people to credibly deny their participation in a data input, randomized response — also known as local differential privacy — makes it possible for participating individuals to credibly deny the contents of their participation records. This approach allows data subjects to answer sensitive or potentially embarrassing questions confidentially.

Like differential privacy, randomized response employs randomization to enhance privacy; however, unlike differential privacy, randomization is applied prior to submission and formal constraints are applied to the randomized substitutions. This means that any chosen substitution must be nearly — though not necessarily exactly — as likely to arise from any given input. As a result, all potential inputs look plausible to an attacker wishing to undo the randomized substitution.

Since the randomized response technique is applied prior to the data leaving a device, data subjects are assured protection from the moment of submission. This protection remains privatized — even in the case of subsequent breach.

k-Anonymization

k-Anonymization is the data equivalent of hiding in a crowd; the more people — or in this case, data points — that are present and generally similar, the harder it is to pick out the details that can identify individuals. This approach reduces re-identification risk by anonymizing indirect identifiers, thereby destroying the signal of data.

In k-anonymization, *K* represents instances of tuples in a data set. A data set is k-anonymous when attributes within it are generalized or suppressed until each row is identical to at least *k*-1 other rows. Therefore, the higher the value of k, the lower the re-identification risk. Just as the larger the crowd is, the less likely you'll find exactly the person you're looking for, k-anonymization works particularly well with large data sets. However, lines of data may have to be redacted if there isn't enough data to anonymize indirect identifiers.

k-Anonymization can help transform, analyze, and share secure data at scale, making it an important privacy enhancing technique for Databricks users dealing with large sets of sensitive data.

What problem do they solve for data teams?

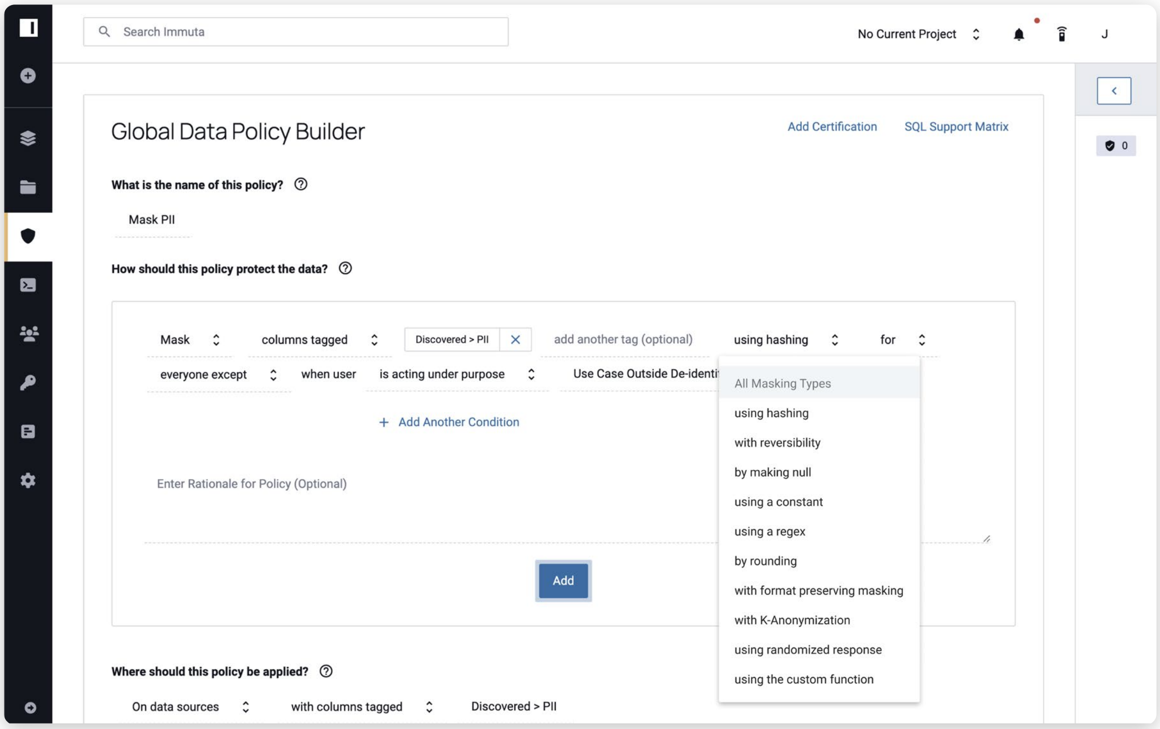
Data teams are often responsible for walking the line between data utility and security. The inherent problem is that creating and implementing controls that maximize both security and utility is highly complex and risky. Effective implementation that mathematically guarantees against re-identification often requires a PhD in applied mathematics.

Although the dynamic security and privacy controls shield some data fields, they are better able to preserve data's utility than some other methods.

In the k-anonymized health data example, suppressing `[age]` and `[zip code]` reduces the risk of re-identification by nulling the data, but a statistically relevant number of `[age]` and `[zip code]` combinations can still be analyzed to observe diagnosis trends by age and gender cohorts, for instance. For researchers at WorldQuant Predictive analyzing COVID-19 spread, this technique enabled data sharing from multiple data sources that helped streamline collaboration and quickly generate predictive models to form hypotheses without inadvertently exposing sensitive PHI or identifying information in the process.

> Read More: Explore WorldQuant Predictive's Immuta journey in this case study.

A key benefit of dynamic security and privacy controls like attribute-based policies, k-anonymization, randomized response, and differential privacy is that they simultaneously reduce the risk of re-identification, maximize data utility and privacy, and enable data sharing between teams. Without these techniques, data teams are hindered in their ability to unlock collaboration and speed to data access without compromising data integrity, security, and compliance. Immuta's natively enforced advanced security capabilities mean Databricks users can access and share critical data — including sensitive data — quickly, efficiently, and securely.



Global Data Policy Builder

What is the name of this policy?

Mask PII

How should this policy protect the data?

Mask columns tagged Discovered > PII add another tag (optional) using hashing for everyone except when user is acting under purpose Use Case Outside De-identi

Where should this policy be applied?

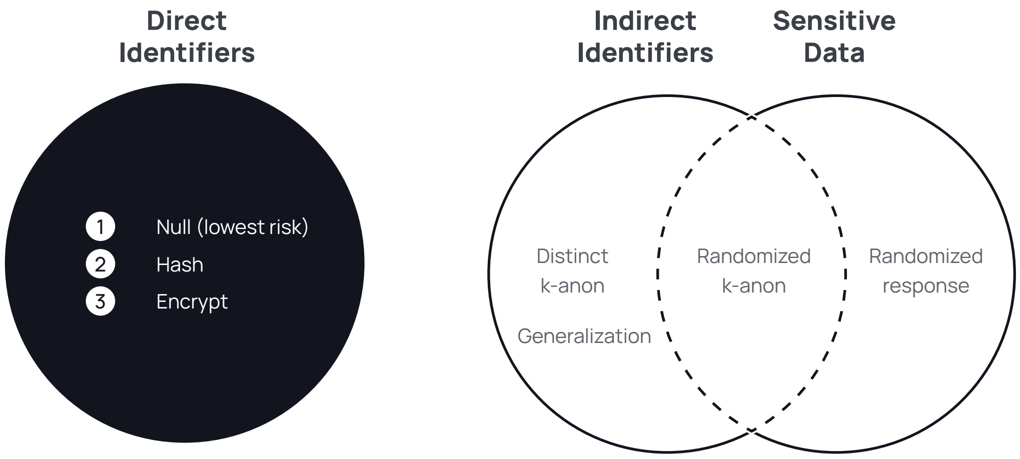
On data sources with columns tagged Discovered > PII

How are they implemented?

To understand how dynamic security and privacy controls apply to and help de-identify specific types of data, it helps to divide data into segments: that which contains direct identifiers, indirect identifiers and sensitive data.

It's important to first understand the two types of identifiers — in other words, the personal information that can be used to help identify an individual.

1. Direct identifiers are the pieces of personal information that are unique to an individual and can be used in isolation to identify that single person. For this reason, direct identifiers are highly sensitive and strictly regulated. Examples of direct identifiers include social security numbers, passport numbers, taxpayer identification numbers, full facial images, and medical record numbers.
2. Indirect identifiers, also known as quasi-identifiers, are pieces of personal information that are not unique to a single person. While indirect identifiers cannot alone be used to identify a specific individual, they are still considered sensitive because they can often be combined with other information to single somebody out. Examples of indirect identifiers include height, ethnicity, hair color, car make and model, and occupation.



As this figure demonstrates, these privacy enhancing technologies (PETs) can buffer protection for indirect identifiers and sensitive data, and vastly reduce the opportunity for inference or linkage attacks.

These advanced controls have grown in popularity across industries to enable secure data sharing. Data teams are well versed in the stringent requirements — and enforcements — associated with regulations like GDPR, HIPAA, and CCPA, which explicitly state that data must be protected such that the chances of an individual being re-identified remain as close to unfeasible as possible. History has shown that personal data is easy to re-identify if not adequately protected — Harvard professor Latanya Sweeney found that 87% of the population can be re-identified simply using birth date, zip code, and gender.

For Databricks users, leveraging personal data like credit card transactions and cell phone location data to drive timely analytics is a necessity, but one that comes at a high risk. By law, a Databricks table containing PHI, such as genders, zip codes, and credit card numbers, must have PETs applied before it can be shared with data scientists and analysts.

Once the table has been registered in Databricks with Immuta, a data engineer can create a new data source and Databricks connection. Then, automated data discovery and tagging flags identifiers and sensitive data in the set. Using Immuta's plain language policy builder, the data engineer can enforce one or more dynamic security and privacy controls. For example, k-anonymity can be enforced using the masking or suppression method on columns tagged as `[gender]` and `[zip code]`.

With this policy selected, Immuta scans the Databricks table to calculate the statistics required for k-anonymization. A fingerprint service runs a query against Databricks to collect counts for each possible group of values in the data source and produces custom predicates for each column. The data engineer can either use this automated return's minimum group size, k, or can manually specify a value for k. To protect identity data, the predicates only contain a whitelist of values visible to users.

This secured data set is exposed as a table in Databricks so data analysts and scientists can access and query the table. Since the controls are enforced natively on read from Databricks, the underlying data remains unmodified and not copied, and policies are applied to the plan that Spark builds for a user's query from the Notebook.

Continuous Data Security Monitoring

What is it?

Data engineers and architects are often asked by security and compliance teams, “who accessed this data over the past two months?” On the surface, this seems like a simple question; in reality, it’s much more complicated.

Continuous data security monitoring and auditing are necessary parts of the data pipeline management process, but in a multi-cloud compute platform environment they are difficult to execute. Disparate compute layers mean security and privacy controls may be enforced in the compute layer, passed through to storage or even implemented in analytical applications. Various data consumer roles and their corresponding permissions add an extra layer of complexity.

Databricks users can bypass tedious manual monitoring and auditing processes with Immuta’s natively integrated capabilities. All audit logs and information, as well as cluster queries to the cloud provider, are done with system accounts so that data usage across cloud compute platforms — not just within Databricks or your cloud provider services — can be captured consistently. This includes audits of not just data queries, but also of all policy actions being taken in Immuta, such as changing policies or subscribers to a table. Additionally, Databricks data teams can implement purpose restrictions that trigger consent workflows and simplify the process of monitoring and auditing data usage.

Immuta’s continuous data monitoring, unified audit logs, automated reports and purpose-based access controls provide granular snapshots of which data consumers accessed specific data sources, when, and for what purpose, as well as changes to data over time. As a result, legal and compliance stakeholders always have quick access to insights that prove compliant data usage across the organization.

What problem does it solve for data teams?

As personal and sensitive data has become more widely available and collected by organizations, data subjects have become increasingly aware of and concerned with how their information is being used.

As personal and sensitive data has become more widely available and collected by organizations, data subjects have become increasingly aware of and concerned with how their information is being used. Regulations have followed accordingly and now continuous data security monitoring and auditing is not an option — it's a requirement.

Yet, as multi-cloud compute platform adoption becomes the new normal, data teams' ability to monitor and audit data use consistently across platforms like Databricks and others is exponentially more complicated. It also broadens the risk landscape for sensitive data, as it moves between more tools and platforms for storage and analysis. In turn, there are a wider array of places that data breaches could potentially occur, creating a need for increased monitoring and breach detection capabilities. This has the potential to substantially limit acceptable use cases for data in cloud analytics. Furthermore, it makes the process of responding to security and compliance requests time-consuming, if not impossible.

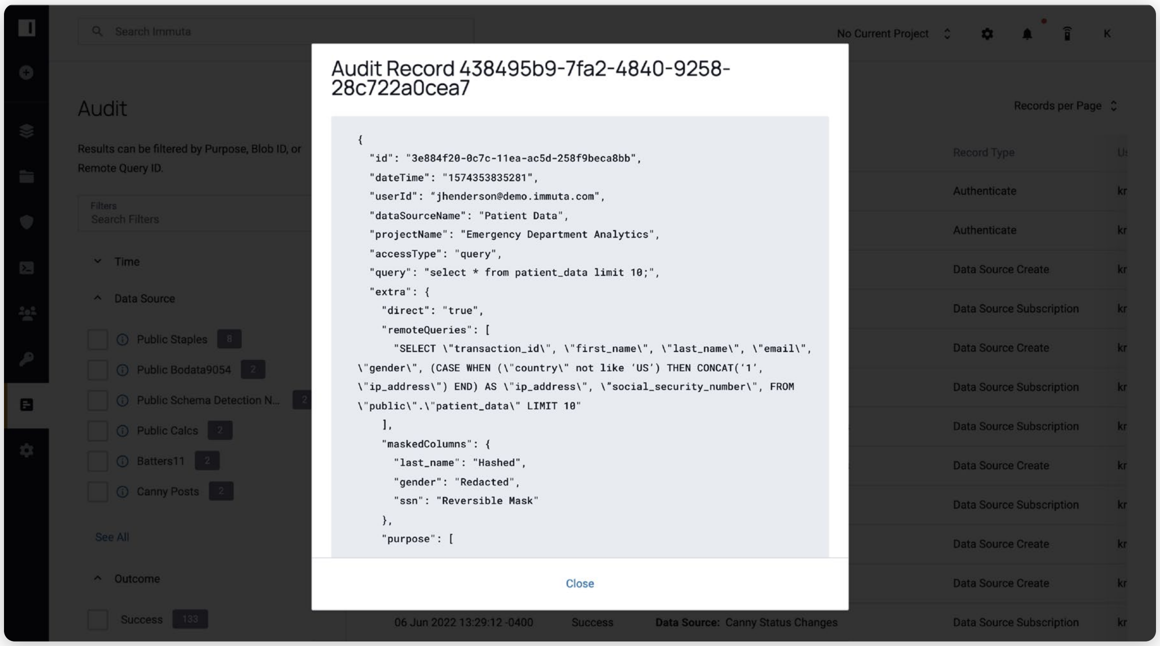
Databricks users avoid these concerns and roadblocks with Immuta's integrated detection, monitoring, and auditing capabilities. Data teams can efficiently provide legal and compliance teams with detailed logs and reports at the data level, to provide full transparency about what data was accessed, by whom, when, and for what purpose. This automates the otherwise-laborious process of formalizing proof of compliance to adhere to federal, state, and industry standards and regulations, which in turn bolsters collaboration between data and compliance teams and mitigates concerns about legal enforcement and personal liability. Data security posture management (DSPM) is also enabled through an analysis engine with both sensitivity level and risk profile indicators to address and mitigate any potential threats.

How is it implemented?

The GDPR requires data consumers to have an acceptable purpose for accessing data, and consequently, that data teams can provide evidence of that purpose.

This is a complicated process because, without the right tools, purpose is difficult to capture. Yet, GDPR and other regulations, including HIPAA, HITRUST, and SOC-2, are not optional and data teams can only avoid incurring noncompliance enforcement with comprehensive audits. Layered on top of regulations are employment and industry standards or contractual agreements, among others.

Immuta captures all details of where data is stored, who owns it, under what conditions it can be accessed, when it was added, and how recently it was queried. Immuta's integration with Databricks enables full monitoring of all user activity, policy related activity and history, compliance and anomaly reports, and alerts and notifications. In addition to auditing, the plain language policy builder also means business stakeholders can assess the strength of policies in meeting compliance standards without having to rely on IT.



Audit Record 438495b9-7fa2-4840-9258-28c722a0cea7

```
{
  "id": "3e884f20-0c7c-11ea-ac5d-258f9beca8bb",
  "dateTime": "1574353835281",
  "userId": "jhenderson@demo.immuta.com",
  "dataSourceName": "Patient Data",
  "projectName": "Emergency Department Analytics",
  "accessType": "query",
  "query": "select * from patient_data limit 10;",
  "extra": {
    "direct": "true",
    "remoteQueries": [
      "SELECT \"transaction_id\", \"first_name\", \"last_name\", \"email\",
\"gender\", (CASE WHEN (\"country\" not like 'US') THEN CONCAT('1',
\"ip_address\") END) AS \"ip_address\", \"social_security_number\", FROM
\"public\".\"patient_data\" LIMIT 10"
    ],
    "maskedColumns": {
      "last_name": "Hashed",
      "gender": "Redacted",
      "ssn": "Reversible Mask"
    },
    "purpose": [
```

Conclusion

Centralized, secure, self-service data access is the best way to maximize data's impact. In today's rapidly evolving market, that can be the secret weapon to gaining a critical competitive edge. Whether analyzing data to inform public health policy decisions or to predict consumer behavior, data consumers need secure access to data — as fast as possible.

This isn't an option without the right combination of data science and automated data security capabilities. Immuta's integration with Databricks enables organizations to automatically secure sensitive data for analytical use in industries ranging from insurance to transportation. With automated data discovery and classification, dynamic security and access controls, and continuous data security monitoring and auditing, Databricks and Immuta together enable data teams to maximize data utility and security.

In fact, Databricks customers that take advantage of Immuta's core, native capabilities experience results such as 100x faster data access, a 93x reduction in data policies, and a 4x increase in data utilization. This means teams are able to accomplish more and unlock more data-driven outcomes in Databricks when Immuta is natively implementing dynamic data security and privacy measures.

To find out what you can accomplish when you combine the power of Databricks and Immuta, schedule a demo today. →

About Immuta

Immuta enables organizations to unlock value from their cloud data by protecting it and providing secure access. The Immuta Data Security Platform provides sensitive data discovery, security and access control, data activity monitoring, and has deep integrations with the leading cloud data platforms. Immuta is now trusted by Fortune 500 companies and government agencies around the world to secure their data. Founded in 2015, Immuta is headquartered in Boston, MA.

About Databricks

Databricks is the data and AI company. More than 7,000 organizations worldwide — including Comcast, Condé Nast, H&M and over 40% of the Fortune 500 — rely on the Databricks Lakehouse Platform to unify their data, analytics and AI. Databricks is headquartered in San Francisco, with offices around the globe. Founded by the original creators of Apache Spark™, Delta Lake and MLflow, Databricks is on a mission to help data teams solve the world's toughest problems. To learn more, follow Databricks on Twitter, LinkedIn and Facebook.

IMMUTA

25 Thomson Place, 4th Floor, Boston, MA 02210 immuta.com (800) 655-0982

IMMUTA

WHITE PAPER

How to Enforce Policy-As-Code

for Databricks Tables

Table of Contents

Introduction

Data security is the responsibility of everyone in the organization. From ETL developers to business users and data consumers, anyone who relies on data shares a responsibility to use it appropriately. However, with several different systems and, in many cases, silos, it can often be difficult to effectively put this ethos into practice.

Immuta is the data security platform that seamlessly enables shared responsibility of security and privacy between organizations, people, and data consumers. In this blog, we will discuss the shift to a security-first data architecture with Databricks and Immuta, and how creating policy-as-code spans the data lifecycle from ingestion to consumption by applications and users.

Immuta allows Databricks users to manage data security and access control in three phases:

Discover

Sensitive data discovery

Secure

Security & access control

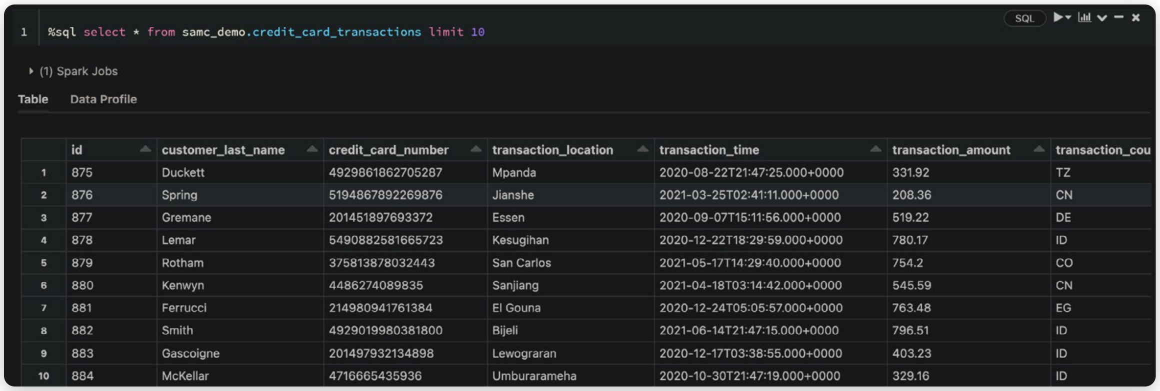
Detect

User & data activity monitoring

Each of these phases is essential to security-first data architectures. Immuta's goal is to help simplify, reduce, and scale data policies so that data teams don't become overwhelmed with managing rules that require constant maintenance.

In this example, we'll walk through how to use Immuta to build a masking policy that obfuscates a credit card field based on the data type rather than hard-coded column names.

Let's start with a sample transaction table. Here's what this table looks like before we apply a masking policy:



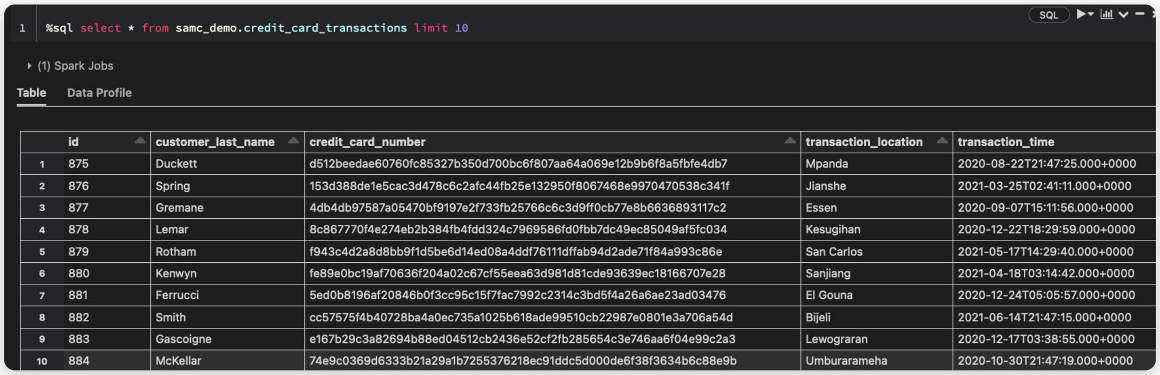
```
%sql select * from samc_demo.credit_card_transactions limit 10
```

(1) Spark Jobs

Table Data Profile

	id	customer_last_name	credit_card_number	transaction_location	transaction_time	transaction_amount	transaction_cou
1	875	Duckett	4929861862705287	Mpanda	2020-08-22T21:47:25.000+0000	331.92	TZ
2	876	Spring	5194867892269876	Jianshe	2021-03-25T02:41:11.000+0000	208.36	CN
3	877	Gremane	201451897693372	Essen	2020-09-07T15:11:56.000+0000	519.22	DE
4	878	Lemar	5490882581665723	Kesugihan	2020-12-22T18:29:59.000+0000	780.17	ID
5	879	Rotham	375813878032443	San Carlos	2021-05-17T14:29:40.000+0000	754.2	CO
6	880	Kenwyn	4486274089835	Sanjiang	2021-04-18T03:14:42.000+0000	545.59	CN
7	881	Ferrucci	214980941761384	El Gouna	2020-12-24T05:05:57.000+0000	763.48	EG
8	882	Smith	4929019980381800	Bijeli	2021-06-14T21:47:15.000+0000	796.51	ID
9	883	Gascoigne	201497932134898	Lewograran	2020-12-17T03:38:55.000+0000	403.23	ID
10	884	McKellar	4716665435936	Umburarameha	2020-10-30T21:47:19.000+0000	329.16	ID

Our goal is to create a policy that masks the data like this:



```
%sql select * from samc_demo.credit_card_transactions limit 10
```

(1) Spark Jobs

Table Data Profile

	id	customer_last_name	credit_card_number	transaction_location	transaction_time
1	875	Duckett	d512beedae60760fc85327b350d700bc6f807aa64a069e12b9b6f8a5fbfe4db7	Mpanda	2020-08-22T21:47:25.000+0000
2	876	Spring	153d388de1e5cac3d478c6c2afc44fb25e132950f8067468e9970470538c341f	Jianshe	2021-03-25T02:41:11.000+0000
3	877	Gremane	4db4db97587a05470bf9197e2f733fb25766c6c3d9ff0cb77e8b6636893117c2	Essen	2020-09-07T15:11:56.000+0000
4	878	Lemar	8c867770f4e274eb2b384fb4fdd324c7969586fd0fbb7dc49ec85049af5fc034	Kesugihan	2020-12-22T18:29:59.000+0000
5	879	Rotham	f943c4d2a8d8bb9f1d5be6d14ed08a4ddf76111dffab94d2ade71f84a993c86e	San Carlos	2021-05-17T14:29:40.000+0000
6	880	Kenwyn	fe89e0bc19af70636f204a02c67cf55eea63d981d81cde93639ec18166707e28	Sanjiang	2021-04-18T03:14:42.000+0000
7	881	Ferrucci	5ed0b8196af20846b0f3cc95c15f7fac7992c2314c3bd5f4a26a6ae23ad03476	El Gouna	2020-12-24T05:05:57.000+0000
8	882	Smith	cc57575f4b40728ba4a0ec735a1025b618ade99510cb22987e0801e3a706a54d	Bijeli	2021-06-14T21:47:15.000+0000
9	883	Gascoigne	e167b29c3a82694b88ed04512cb2436e52cf2fb285654c3e746aa6f04e99c2a3	Lewograran	2020-12-17T03:38:55.000+0000
10	884	McKellar	74e9c0369d6333b21a29a1b7255376218ec91ddc5d000de6f38f3634b6c88e9b	Umburarameha	2020-10-30T21:47:19.000+0000

Registering a Databricks Data Source with Immuta

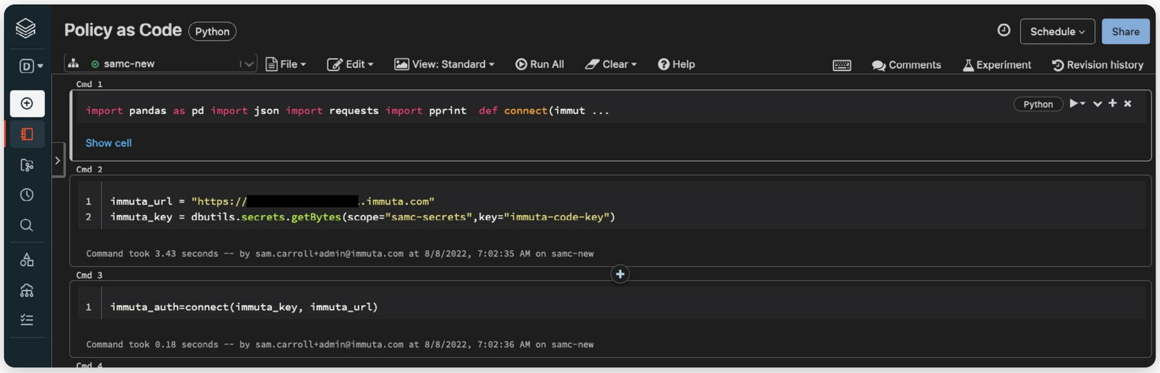
The first step in the process of building a policy is to tell Immuta to protect the table, which starts by registering a data source. Once registration is done, we can create a connection to the Immuta API from our notebook and run sensitive data detection. This will allow us to understand what types of sensitive data we have and where it lives, and build an appropriate policy that can be applied against our table of credit card data.

Let's walk through the first two phases of implementation for data access controls with the Immuta API. In our Databricks notebook, we will illustrate how Immuta automates and orchestrates policy creation in a scalable way.

The first thing we need to do is create a connection to Immuta. Below is a sample Python function that we can call to create a connection with a bearer token.

```
def connect(immuta_apikey, immuta_url):
    payload = {"apikey": immuta_apikey }
    r = requests.post(immuta_url + '/bim/apikey/authenticate', data=payload)
    bearer_token=r.json()['token']
    header_yaml={"Authorization": "Bearer" + bearer_token , "Content-Type":
"text/plain"}
    return header_yaml
```

Let's walk through the first two phases of implementation for data access controls with the Immuta API. In our Databricks notebook



Cmd 1

```
import pandas as pd import json import requests import pprint  def connect(immut ...
```

Show cell

Cmd 2

```
immuta_url = "https://[redacted].immuta.com"
immuta_key = dbutils.secrets.getBytes(scope="samc-secrets",key="immuta-code-key")

Command took 3.43 seconds -- by sam.carroll+admin@immuta.com at 8/8/2022, 7:02:35 AM on samc-new
```

Cmd 3

```
immuta_auth=connect(immuta_key, immuta_url)

Command took 0.18 seconds -- by sam.carroll+admin@immuta.com at 8/8/2022, 7:02:36 AM on samc-new
```

Cmd 4

Now that we have the token, we can start interacting with Immuta directly from Databricks.

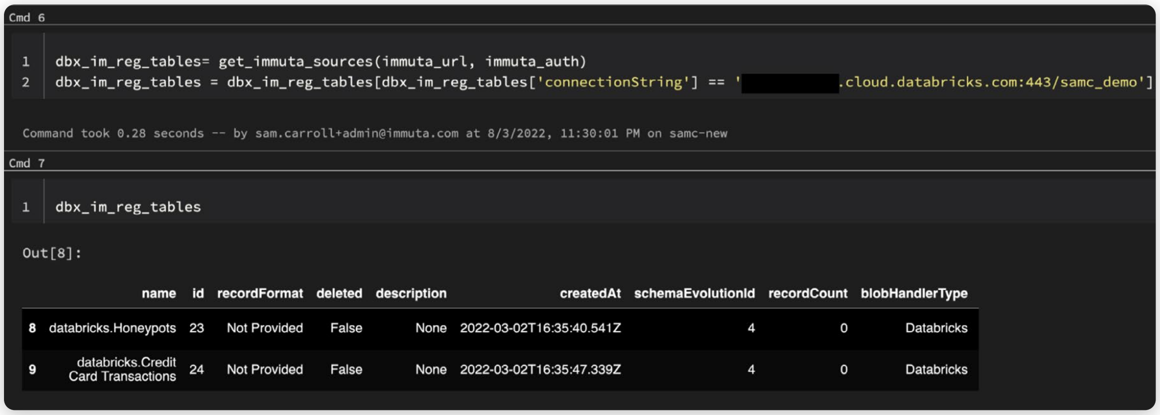
First, we will make sure that the table has been registered. This is done by calling the Immuta API to list all data sources. Since we don't know the data source id, we will filter all of our sources by the name of our Databricks host to see if the table has been registered:

```
immuta_auth=connect(immuta_key, immuta_url)

dbx_im_reg_tables= get_immuta_sources(immuta_url, immuta_auth)
dbx_im_reg_tables = dbx_im_reg_tables[dbx_im_reg_tables['connectionString']
== databrickshost.cloud.databricks.com:443/samc_demo']
```

The function get_immuta_sources is just a wrapper function for simplicity. It passes the appropriate Immuta API path to a function that sends a request object with authorization to the API. Below is an example of these functions. Throughout this blog, we'll use wrapper functions to highlight what exactly the call is doing:

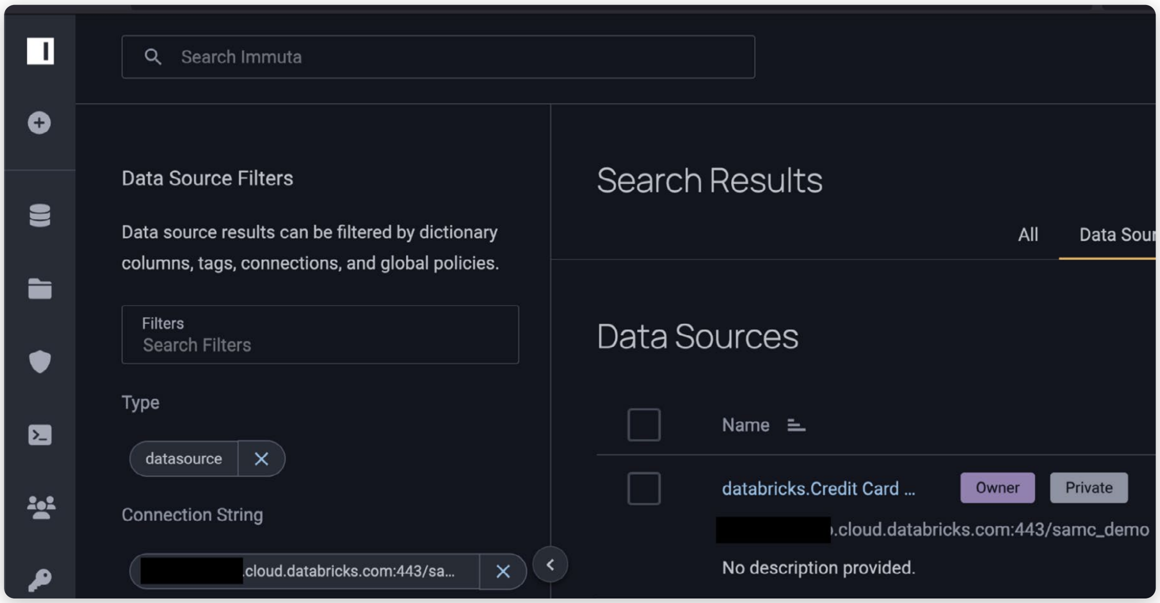
```
def get_immuta_endpoint(immuta_url, header_yaml, uri,record_path):
    r = requests.get(immuta_url + '/' + uri , headers=header_yaml)
    if record_path == '':
    df = pd.json_normalize(r.json())
    else:
    df = pd.json_normalize(r.json(),record_path=record_path)
    return df

def get_immuta_sources(immuta_url, header_yaml):
    return get_immuta_endpoint(immuta_url, header_yaml,'dataSource', 'hits')
```

Now, let's run this in Databricks and see if our credit card table has already been registered with Immuta:



```
Cmd 6

dbx_im_reg_tables= get_immuta_sources(immuta_url, immuta_auth)
dbx_im_reg_tables = dbx_im_reg_tables[dbx_im_reg_tables['connectionString'] == '[redacted].cloud.databricks.com:443/samc_demo']

Command took 0.28 seconds -- by sam.carroll+admin@immuta.com at 8/3/2022, 11:30:01 PM on samc-new
Cmd 7

dbx_im_reg_tables

Out[8]:

                  name  id  recordFormat  deleted  description                 createdAt  schemaEvolutionId  recordCount  blobHandlerType
8  databricks.Honeypots  23  Not Provided    False         None  2022-03-02T16:35:40.541Z                  4            0       Databricks
9     databricks.Credit  24  Not Provided    False         None  2022-03-02T16:35:47.339Z                  4            0       Databricks
     Card Transactions
```

This could also be done by searching in the Immuta UI:



Data source results can be filtered by dictionary columns, tags, connections, and global policies.

[redacted].cloud.databricks.com:443/sa...

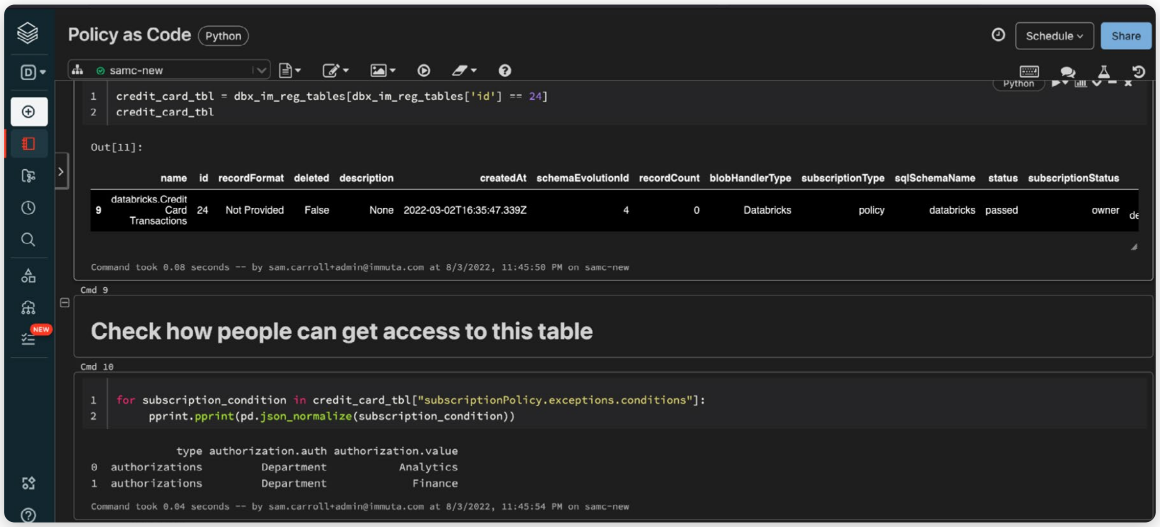
databricks.Credit Card ... Owner Private
[redacted].cloud.databricks.com:443/samc_demo
No description provided.

Utilizing Sensitive Data Discovery for Databricks Policies

Now that we know the table has been registered (this step could also be done programmatically), let's take a look at what data access rules the data owner has in place. We will do this by looking up the data source by its id, as listed above:

```
#use the id from the list data source command to get additional information
from Immuta about this table:
credit_card_tbl = immuta_sources[dbx_im_reg_tables['id'] == 24]

#unnest the json policy conditions for subscribing to the dataset:
for subscription_condition in
credit_card_tbl["subscriptionPolicy.exceptions.conditions"]:
    pprint.pprint(pd.json_normalize(subscription_condition))
#output:
type authorization.auth authorization.value
0 authorizations Department Analytics
1 authorizations Department Finance
```

The image below shows the outputs of these commands:



```
credit_card_tbl = dbx_im_reg_tables[dbx_im_reg_tables['id'] == 24]
credit_card_tbl
```

Out[11]:

	name	id	recordFormat	deleted	description	createdAt	schemaEvolutionId	recordCount	blobHandlerType	subscriptionType	sqlSchemaName	status	subscriptionStatus	de
9	databricks.Credit Card Transactions	24	Not Provided	False	None	2022-03-02T16:35:47.339Z	4	0	Databricks	policy	databricks	passed	owner	

```
Command took 0.08 seconds -- by sam.carroll+admin@immuta.com at 8/3/2022, 11:45:50 PM on samc-new
```

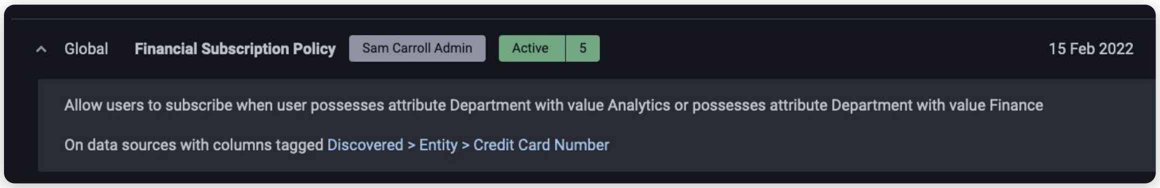
Cmd 9

Check how people can get access to this table

Cmd 10

```
for subscription_condition in credit_card_tbl["subscriptionPolicy.exceptions.conditions"]:
    pprint.pprint(pd.json_normalize(subscription_condition))

            type authorization.auth authorization.value
0  authorizations          Department           Analytics
1  authorizations          Department             Finance
Command took 0.04 seconds -- by sam.carroll+admin@immuta.com at 8/3/2022, 11:45:54 PM on samc-new
```

This shows that users who are in the Department of Finance or Analytics can subscribe to the data set. Investigating further, you can also see that this data set has a global subscription policy enforced on it. A global policy is one that will apply to any data set with a specific Immuta-discovered classification (in this case, credit card information):



Global Financial Subscription Policy Sam Carroll Admin Active 5 15 Feb 2022

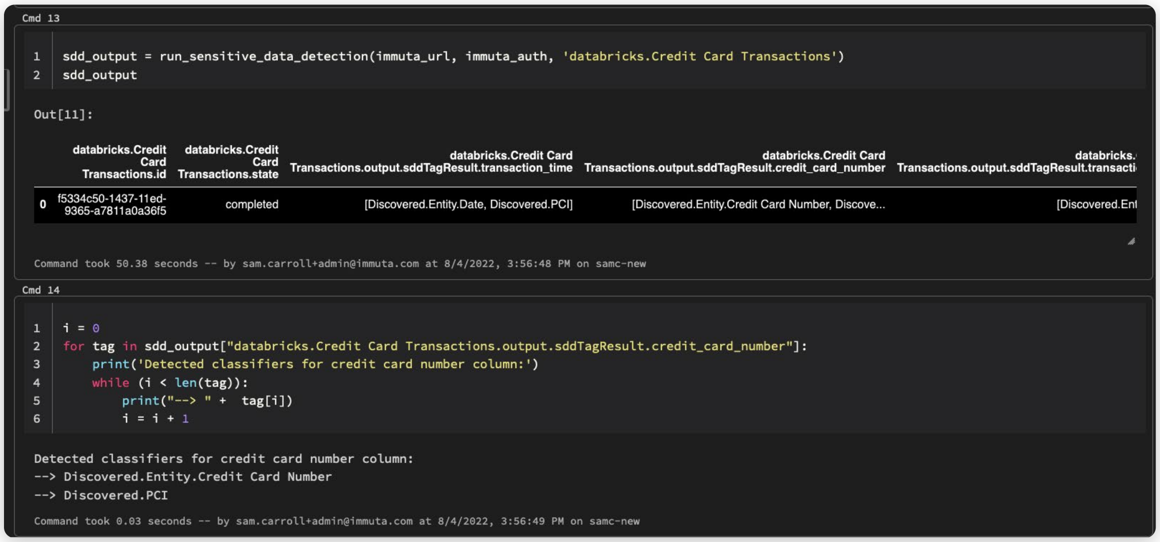
Allow users to subscribe when user possesses attribute Department with value Analytics or possesses attribute Department with value Finance

On data sources with columns tagged Discovered > Entity > Credit Card Number

Global policies are critical for scalability because they reduce the number of policies we have to create when adding new tables that contain this particular type of information.

To help streamline data access management, Immuta's sensitive data discovery can be called directly from a Databricks notebook to auto-classify data. This classification allows you to build a policy based on the contents of the data, in addition to the hard-coded column, table, or database name. Below is an example call to Immuta's sensitive data discovery API that will return the classifiers to your Databricks notebook:

```
sdd_output = run_sensitive_data_detection(immuta_url, immuta_auth, 'data-
bricks.Credit Card Transactions')
sdd_output

i = 0
for tag in sdd_output["databricks.Credit Card Transactions.output.sddTagRe-
sult.credit_card_number"]:
    print('Detected classifiers for credit card number column:')
    while (i < len(tag)):
    print("--> " +  tag[i])
    i = i + 1
#output:
Detected classifiers for credit card number column:
--> Discovered.Entity.Credit Card Number
--> Discovered.PCI
```



Cmd 13

```
sdd_output = run_sensitive_data_detection(immuta_url, immuta_auth, 'databricks.Credit Card Transactions')
sdd_output
```

Out[11]:

	databricks.Credit Card Transactions.id	databricks.Credit Card Transactions.state	databricks.Credit Card Transactions.output.sddTagResult.transaction_time	databricks.Credit Card Transactions.output.sddTagResult.credit_card_number	databricks. Transactions.output.sddTagResult.transacti
0	f5334c50-1437-11ed-9365-a7811a0a36f5	completed	[Discovered.Entity.Date, Discovered.PCI]	[Discovered.Entity.Credit Card Number, Discove...	[Discovered.Ent

Command took 50.38 seconds -- by sam.carroll+admin@immuta.com at 8/4/2022, 3:56:48 PM on samc-new

Cmd 14

```
i = 0
for tag in sdd_output["databricks.Credit Card Transactions.output.sddTagResult.credit_card_number"]:
    print('Detected classifiers for credit card number column:')
    while (i < len(tag)):
        print("--> " +  tag[i])
        i = i + 1
```

```
Detected classifiers for credit card number column:
--> Discovered.Entity.Credit Card Number
--> Discovered.PCI
```

Command took 0.03 seconds -- by sam.carroll+admin@immuta.com at 8/4/2022, 3:56:49 PM on samc-new

Building a Databricks Data Masking Policy As Code

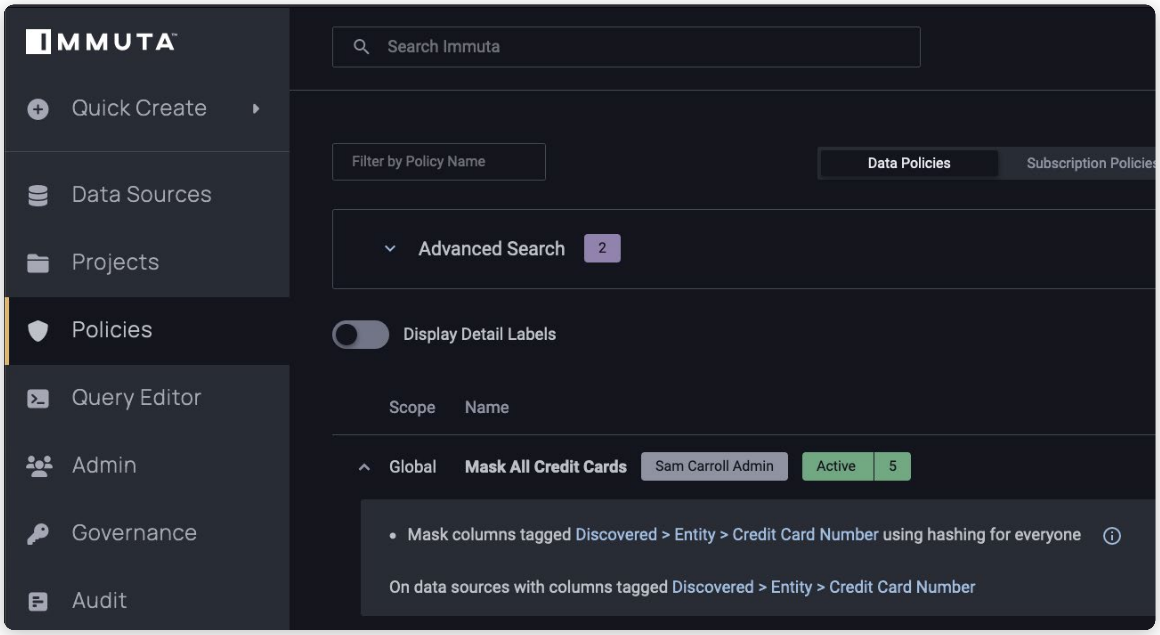
Now that we see how Immuta classified the credit card number column, let's use that classification to build a masking policy that will dynamically mask it for every column in which we find that type of data:

```
masking_policy = """
actions:
    - description: Mask all Credit Cards
    rules:
    - config:
            fields:
                    - columnTag: Discovered.Entity.Credit Card Number
                    type: columnTags
            maskingConfig:
                    type: Hash
            type: Masking
circumstanceOperator: any
circumstances:
    - columnTag: Discovered.Entity.Credit Card Number
    type: columnTags
name: Mask All Credit Cards
policyKey: Mask All Credit Cards
staged: false
template: false
type: data"""
```

This policy will mask all data classified as a credit card number using hashing for everyone in the organization. Now, let's tell Immuta to apply this policy:

```
set_immuta_policy(immuta_url, immuta_auth, masking_policy)
#output:
    dryRun creating updating policyId
0 False  True     False    35
```

Notice that you can now see the policy set in the Immuta UI. You could also set up policy promotion settings that require approval by a data owner/steward before activating. This is useful for the separation of duties between developers, data stewards, and data consumers.

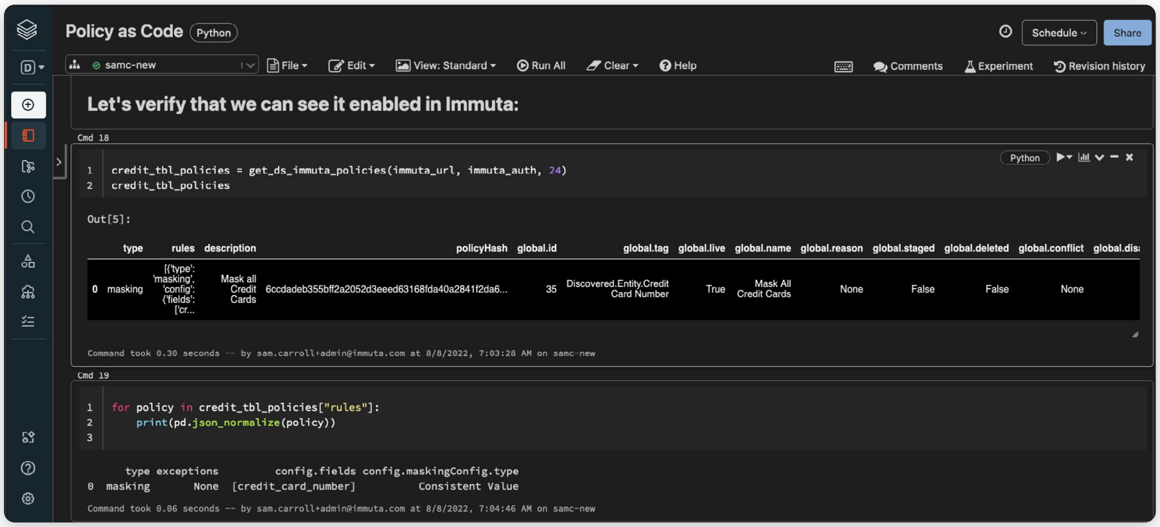


- Mask columns tagged Discovered > Entity > Credit Card Number using hashing for everyone

On data sources with columns tagged Discovered > Entity > Credit Card Number

Now let's check programmatically to see if this policy was enforced on our credit_card_transactions table using the Immuta table id we discovered earlier:

```
credit_tbl_policies = get_ds_immuta_policies(immuta_url, immuta_auth, 24)

for policy in credit_tbl_policies["rules"]:
    print(pd.json_normalize(policy))
#output
    Type    exceptions config.fields        config.maskingConfig.type
0 masking None       [credit_card_number] Consistent Value
```



Let's verify that we can see it enabled in Immuta:

Cmd 18

```
credit_tbl_policies = get_ds_immuta_policies(immuta_url, immuta_auth, 24)
credit_tbl_policies
```

```
Out[5]:
```

	type	rules	description	policyHash	global.id	global.tag	global.live	global.name	global.reason	global.staged	global.deleted	global.conflict	global.dis
0	masking	[{'type': 'masking', 'config': {'fields': ['cr...	Mask all Credit Cards	6ccdadeb355bff2a2052d3eeed63168fda40a2841f2da6...	35	Discovered.Entity.Credit Card Number	True	Mask All Credit Cards	None	False	False	None	

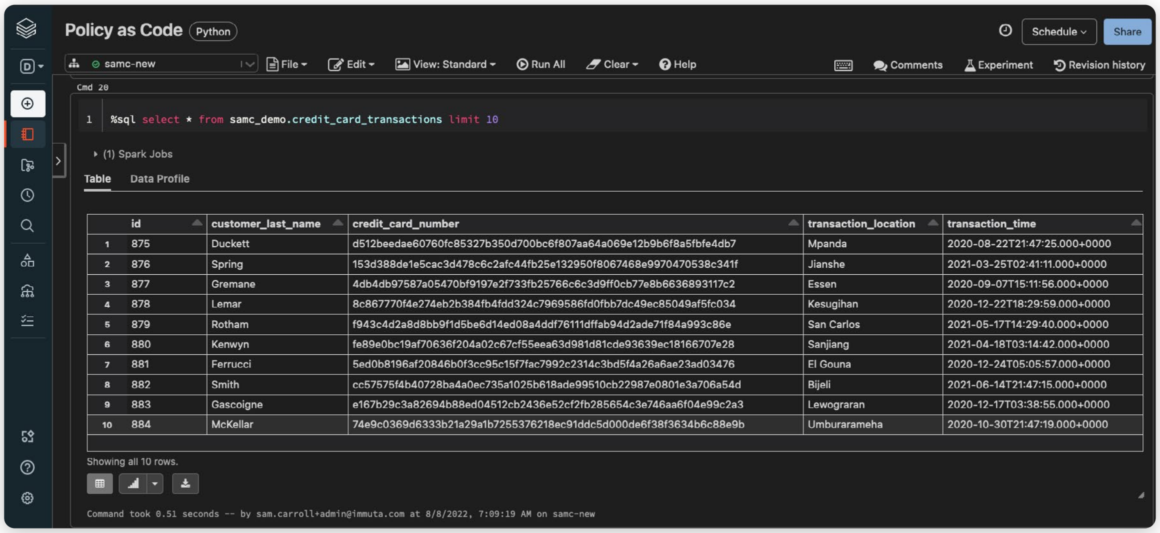
```
Command took 0.30 seconds -- by sam.carroll+admin@immuta.com at 8/8/2022, 7:03:28 AM on samc-new
```

Cmd 19

```
for policy in credit_tbl_policies["rules"]:
    print(pd.json_normalize(policy))

```

```
     type exceptions          config.fields config.maskingConfig.type
0  masking       None  [credit_card_number]          Consistent Value
Command took 0.06 seconds -- by sam.carroll+admin@immuta.com at 8/8/2022, 7:04:46 AM on samc-new
```

Finally, let's re-run the query to see the masking policy in effect on the Databricks table:



Cmd 20

```
%sql select * from samc_demo.credit_card_transactions limit 10
```

	id	customer_last_name	credit_card_number	transaction_location	transaction_time
1	875	Duckett	d512beedae60760fc85327b350d700bc6f807aa64a069e12b9b6f8a5fbfe4db7	Mpanda	2020-08-22T21:47:25.000+0000
2	876	Spring	153d388de1e5cac3d478c6c2afc44fb25e132950f8067468e9970470538c341f	Jianshe	2021-03-25T02:41:11.000+0000
3	877	Gremane	4db4db97587a05470bf9197e2f733fb25766c6c3d9ff0cb77e8b6636893117c2	Essen	2020-09-07T15:11:56.000+0000
4	878	Lemar	8c867770f4e274eb2b384fb4fdd324c7969586fd0fbb7dc49ec85049af5fc034	Kesugihan	2020-12-22T18:29:59.000+0000
5	879	Rotham	f943c4d2a8d8bb9f1d5be6d14ed08a4ddf76111dffab94d2ade71f84a993c86e	San Carlos	2021-05-17T14:29:40.000+0000
6	880	Kenwyn	fe89e0bc19af70636f204a02c67cf55eea63d981d81cde93639ec18166707e28	Sanjiang	2021-04-18T03:14:42.000+0000
7	881	Ferrucci	5ed0b8196af20846b0f3cc95c15f7fac7992c2314c3bd5f4a26a6ae23ad03476	El Gouna	2020-12-24T05:05:57.000+0000
8	882	Smith	cc57575f4b40728ba4a0ec735a1025b618ade99510cb22987e0801e3a706a54d	Bijeli	2021-06-14T21:47:15.000+0000
9	883	Gascoigne	e167b29c3a82694b88ed04512cb2436e52cf2fb285654c3e746aa6f04e99c2a3	Lewograran	2020-12-17T03:38:55.000+0000
10	884	McKellar	74e9c0369d6333b21a29a1b7255376218ec91ddc5d000de6f38f3634b6c88e9b	Umburarameha	2020-10-30T21:47:19.000+0000

Showing all 10 rows.

```
Command took 0.51 seconds -- by sam.carroll+admin@immuta.com at 8/8/2022, 7:09:19 AM on samc-new
```

Immuta's integration with Databricks makes it so that the data can be masked without users even being aware that Immuta is enabled on the table, which removes any potential performance impacts. This single policy has a global scope, meaning it will automatically be applied to any table that is registered with Immuta and contains credit card information.

Immuta enables collaborative data security across all users in the organization. With Immuta and Databricks, ETL developers, business users, and data consumers can easily understand data security and share the responsibility of using all their data appropriately. By enabling policy-as-code, in addition to plain language policy authoring, Immuta helps ensure that data is protected from curation to consumption, throughout the life of the data set. As a result, organizations are able to simplify operations, improve data security, and unlock more value from their Databricks data.

CASE STUDY

How J.B. Hunt Is Driving Freight Transportation Into the Future

J.B. Hunt Transport Services, Inc. is an American transportation and logistics company based in Lowell, Arkansas.

It was founded by Johnnie Bryan Hunt and Johnelle Hunt over 60 years ago and has grown to be a Fortune 500 company that is one of the largest trucking firms in the U.S. It is their mission to build North America's most efficient transportation network and they are working on streamlining freight logistics and providing the best carrier experience possible.

Key Takeaways

- Increased permitted use cases for cloud analytics by 100% with Immuta while also achieving cost savings and productivity gains
- Provided 200+ active users with access to 100+ databases using global PII policies across 250 tables and local PII policies across 50 tables

"Databricks opens up many opportunities for self-service data analytics, data science, and enterprise reporting. Paired with Immuta, we can make all our data available to all types of business analysts, data scientists and data engineers."

— **Ajay Sahu**, Director Of Enterprise Data Management, J.B. Hunt

Architecture

IMMUTA

databricks

Google BigQuery

Google Cloud

Industry

Transportation

Challenge

In order to fulfill their goal of becoming the most efficient freight transportation network in North America, J.B. Hunt focuses primarily on connecting carriers with their ideal shipper, taking into consideration details such as price, weight and location.

> "What we're doing from a data science point of view is building pricing models and load recommendation models to improve operations," explained Doug Mettenburg, Vice President of Engineering and Technology at J.B. Hunt. "But in my 20 years here, data refreshes have only been, at best, every night, and that's the case across our industry. The problem is, trucks move. Having to wait overnight for data made a lot of what we wanted to provide around tracking and modeling impossible."

Simply put, legacy architecture, a lack of AI capabilities, and the inability to securely handle big data caused significant roadblocks.

Prior to Databricks and Immuta, J.B. Hunt had their data locked in legacy enterprise data warehouse (EDW) platforms. Their systems struggled to process and store the massive data generated by hundreds of thousands of equipment pieces. They also lacked the necessary levels of data security and the ability to support data streams generated by IoT sensors on their trucks and carriages. J.B. Hunt knew it was time for a change.

Solution

J.B. Hunt's goal was to modernize their data infrastructure by building an open, scalable and unified cloud data architecture. To achieve their goal, J.B. Hunt moved to the Databricks Lakehouse Platform for its ability to unify data engineering and data science functions on one open and scalable platform.

> "As we look toward expanding our ML and real-time analytics capabilities, it was critical that we built upon a platform that provides the flexibility to quickly deploy use cases regardless of which cloud or tool sets are being leveraged across our diverse operations — and that's what Databricks provided us."

With Databricks Delta Lake, J.B. Hunt not only has the ability to put all their data in one place for easy access across the organization — they can also ensure the performance and reliability of streaming data pipelines at any scale. The support for Delta Lake as the open storage layer brought efficiency and portability to J.B. Hunt's teams as they moved terabytes of their existing data onto the platform. By streaming in real time to Delta Lake — with web, mobile, location, IoT and other application data — J.B. Hunt can analyze larger, more complete data sets to run analytics and ML faster than ever. With MLflow, the data science team is now able to establish reproducibility of code and experiments to ensure they're reusable by multiple data scientists.

Additionally, J.B. Hunt integrated Immuta with Databricks to provide cloud data access control. Immuta gives J.B. Hunt a level of data security that wasn't possible with their legacy EDW.

J.B. Hunt had over 200 users that needed access to sensitive data and they required full auditing, anonymization, and time based access controls. They had tried using manual tools to protect sensitive data but the complexity and staffing requirements to mask and unmask data was too cumbersome and delayed access to the data, resulting in increased costs and limiting analytics. Also, they couldn't risk delays in complying with privacy laws such as the California Consumer Privacy Act (CCPA) that would increase the potential for regulatory fines.

"Before Immuta, we could not make our data widely available to the users due to security concerns," explained Tina Headrick, Senior Manager of Data Governance and Privacy at J.B. Hunt. "Immuta has allowed us to open up the data to a wide variety of users."

"Databricks paired with Immuta opens up many opportunities for self-service data analytics, data science, and enterprise reporting," explained Ajay Sahu, the Director of Enterprise Data Management at J.B. Hunt. "With this modern data stack, we can make all our data available to all types of business analysts, data scientists and data engineers."

Although large-scale cloud migration projects are often incredibly complex, implementing Immuta on Databricks was straightforward and led to massive cost savings. J.B. Hunt's operations couldn't slow down and the solution required flexibility to rapidly create and update data access rules.

REQUIREMENT	OUTCOME WITH DATABRICKS, WITHOUT IMMUTA	OUTCOME WITH DATABRICKS AND IMMUTA
Redaction of older data	✓	✓
Masking sensitive data	⊖	✓
Delete detail data but keep aggregates	✓	✓
Row-level security	✗	✓
Auditing	⊖	✓
Manageability	⊖	✓
Performance	✓	✓

Results

With Databricks and Immuta, J.B. Hunt now has a single, secure, governed source of truth that delivers operational efficiency. Across the company, they've increased permitted use cases for cloud analytics by 100% by using Immuta.

In terms of collaboration, the team has succeeded in bringing the various data teams together to accelerate data science productivity. The company's HR team can run ML models to predict resourcing for their fleets and marketing can quickly perform analytics on Marketo data.

"Immuta has allowed us to automate data access control & privacy protection. As we expand our data estate to Google's BigQuery platform, we will be able to implement consistent data access policies on both Databricks and Big Query," added Tina.

J.B. Hunt has seen a steady rise in the number of Immuta users who are attracted to the platform's ease of use and ready availability of secure data that was previously out of reach for them.

J.B. Hunt has experienced a significant growth in their business operations, and the demand for securely managing data continues to increase.

> "We now have a single, secure source of truth to run our business on," added Ajay. "Databricks and BigQuery allow us to show the true value of our data to the company and Immuta allows us to do it securely and in full compliance. To top it off, we're saving huge amounts of time and money in the process."

About Immuta

Immuta enables organizations to unlock value from their cloud data by protecting it and providing secure access. The Immuta Data Security Platform provides sensitive data discovery, security and access control, data activity monitoring, and has deep integrations with the leading cloud data platforms. Immuta is now trusted by Fortune 500 companies and government agencies around the world to secure their data. Founded in 2015, Immuta is headquartered in Boston, MA.

IMMUTA

25 Thomson Place, 4th Floor, Boston, MA 02210 immuta.com (800) 655-0982