snowflake®

SNOWPARK: BUILDING BETTER DATA PIPELINES AND MODELS IN THE DATA CLOUD

Program in Python, Scala, or Java

CHAMPION GUIDES

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Snowflake started its journey to the Data Cloud by completely reengineering the world of data and rethinking how a reliable, secure, high-performance, and scalable data-processing system should be architected for the cloud.

The result is Snowpark: a new developer framework for Snowflake. Snowpark allows data engineers, data scientists and data developers to code in their familiar way with their language of choice, and execute pipeline, ML workflow and data apps faster and more securely, in a single platform.

The Snowpark API brings deeply integrated, DataFrame–style programming to the languages developers like to use, including Scala and Java Snowpark UDFs help you expand more data use cases easily and run inside of Snowflake, including Java UDFs, JavaScript UDFs, external functions, and Python UDFs.
Snowpark is designed to make building complex data pipelines much easier and to allow developers to interact with Snowflake directly without having to move data.

With Snowpark, possible use cases include:

- **Using machine learning to augment data by hosting trained models in Java**
- **Scanning for anomalies in your data**
- **Developing a routine to identify PII**
- **Deploying shared Java libraries to standardize your business logic**

Snowpark is a significant step forward in data programmability, making it easy to get Snowflake's platform to do more for you. Since its launch, Snowpark has already accumulated compelling use cases, including many from our partners in the Snowpark Accelerated program.

What can you build with Snowpark? We will feature some example use cases in this ebook, but you can build many more. Let's get started.

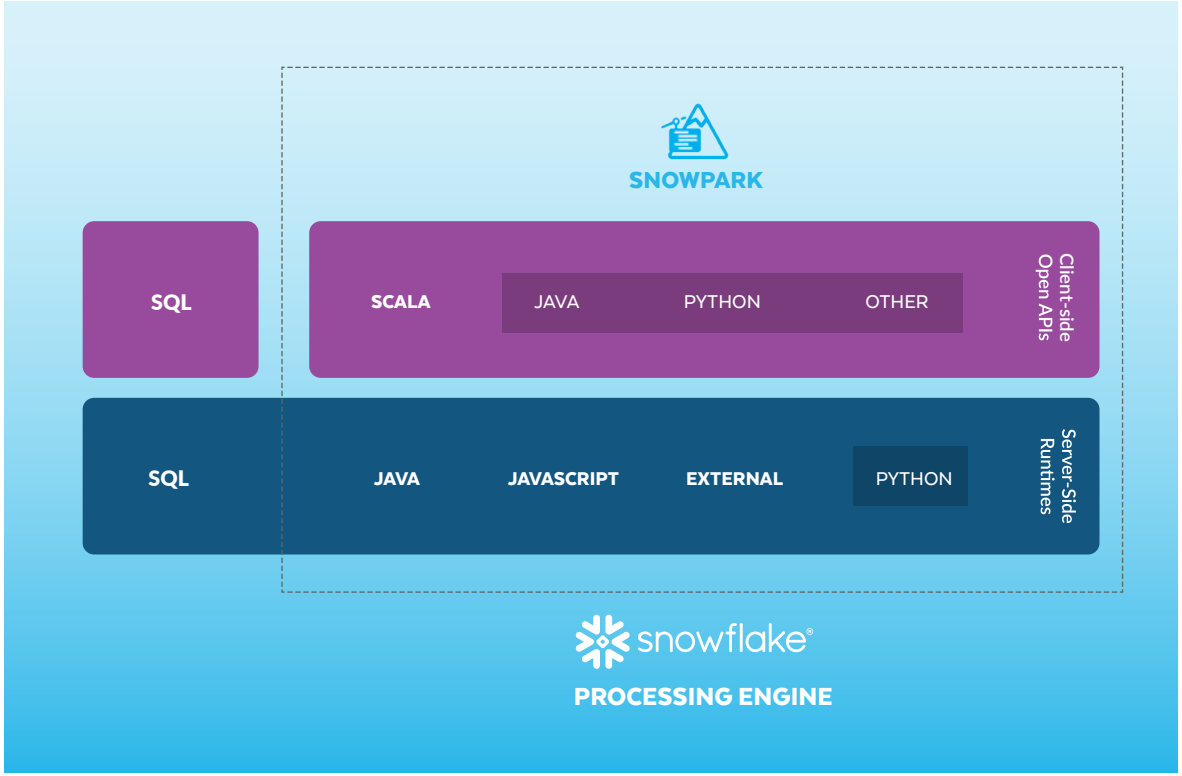


Figure 1: Snowpark enables developers and engineers to interact with Snowflake directly without having to move data.

HOW TO BUILD BETTER PIPELINES AND DATA MODELS WITH SNOWPARK

To understand how Snowpark works, let's take a look at a generic example with PII (personal identifiable information).

STREAMLINED QUERIES, SEAMLESS CONVERSIONS

With the Snowpark API, developers can build queries using DataFrames in their code without creating and passing along SQL strings, for example:

```
val sess = // get connection to Snowflake


val sales:DataFrame = sess.table("sales")
val line_items:DataFrame = sess.table("sales_details")

val query = sales.join(line_items, sales("id") ===
line_items("sid"))
                 .groupBy(line_items("product_id"))

                 .count()
```

Because the Snowpark API uses first-class language constructs, it provides first-class support in the development environment, including type checking, IntelliSense, and error reporting. And Snowpark seamlessly converts these operations into SQL that runs inside Snowflake's high-performance, scalable engine.

But the Snowpark API is more than just a simpler way to write queries. You can also use your own custom logic. Here's an example that uses custom code to mask PII:

```
val maskPii = (s:String) => {
  // Custom PII detection logic.
}
```

Using the Snowpark API, you can easily classify this code as a user-defined function (UDF). Then you can use it in DataFrame operations:

```
val maskPiiUdf = udf(maskPii)
sess.table("emails")
    .withColumn("body", maskPiiUdf(col("body")))
    .show()
```

The Snowpark API pushes your logic to Snowflake, so it runs next to your data. Your code is hosted in a secure, sandboxed JVM (Java Virtual Machine) inside Snowflake's warehouses.

AUTOMATION ELIMINATES HAND-CODING

To see how Snowpark simplifies common operations, let's look at how it enables you to apply your PII detection logic to all the string columns in a table. With SQL, you'd have to hand-code a query for each table or write code to generate the query. With Snowpark, you can easily write a generic routine:

```
val maskTable = (df:DataFrame) => {
  df.select(df.schema.map(field =>
    if (field.dataType == StringType) maskPiiUdf(col(field.name))
    else col(field.name))
  )
}
```

Then you can use this generic routine to mask all the PII in any table with ease:

```
val maskedEmails = maskTable(sess.table("emails"))
```

You just need these few lines of code to get Snowpark to generate a robust, schema-driven query dynamically.

BUILD COMPLEX LOGIC WITH JAVA UDFS

SQL is still Snowflake's bread and butter, and SQL users can get the full benefit of the platform's new capabilities through Java UDFs.

With Java UDFs, you can build complex logic that exposes a simple function interface:

```
public class Sentiment
{
   public float score(String text)
   {
       // Your sentiment analysis logic here.
   }
}
```

In building these functions, you can use your existing toolset, including source control, development environments, and debugging tools, as well as any libraries you need. Snowpark gives you the ability to take useful code from GitHub or other sources and use it in Snowflake.

The process to put your code into SQL is simple. Build a JAR (or JARs), load into Snowflake, and register a function:

```
create function sentiment(txt string) returns float
language java
imports = ('@jars/Sentiment.jar')
handler = 'Sentiment.score';
```

Now, any SQL user can use the logic you've built like any other function:

```
select id, sentiment(body)
from emails;
```

Java UDFs let you use your existing tooling for complex cases. But sometimes your use case is more basic, so this new functionality includes simple, inline definitions as well:

```
create or replace function reverse(s string) returns string
language java
handler = 'Reverse.reverse'
target_path = '@jars/Reverse.jar'
as
$$
public class Reverse
{
    public String reverse(String s)
    {
        return new StringBuilder(s).reverse().toString();
    }
}
$$;
```

SNOWPARK USE CASES FOR DATA ENGINEERING

Data engineering is complex for two main reasons. First, data engineers often need to use multiple systems and solutions, making the data pipeline architecture rigid and overly complicated. One of the primary reasons for this is that data engineering is a collaborative effort across teams. Data analysts may prefer SQL and GUI-based tools. In contrast, data scientists generally prefer to prepare data using notebooks and Python, and data engineers and developers need additional tools to tackle complex code and programming constructs. Data often needs to travel across these different systems to make pipelines work, leading to complex architectures that can jeopardize security and impair data governance.

Second, managing and working with data processing infrastructure typically requires significant manual effort and maintenance overhead. As a result, data engineers are often spread thin, spending most of their time maintaining and fixing pipelines.

Snowpark is designed to solve both these problems. Read more in the following use cases from our partners.

USE CASE: DELIVERING TRUSTED DATA WITH TALEND

Your analytics capability is only as good as the quality of your data. How do you measure it? Now, you can perform a health check on your data within Snowflake using Talend's at-a-glance Data Trust Score™, a core capability of the Talend Data Inventory product. The Trust Score™ helps you identify and diagnose key issues with your data, and helps all key stakeholders embark on a journey towards building confidence in your decision-making.

Typically, to profile a large amount of data sets by an external application, a sample of your data is extracted so the analysis can run in a different system from where it's stored. Moving data outside of Snowflake

introduces its own challenges; data security and privacy risks, data ingress egress charges, failed processing due to non-scalable resources, and the outcomes are still inaccurate, simply because they are based upon a sample of data.

Calculation of the Trust Score™ for Snowflake datasets alleviates many of these concerns because the processing to calculate the score happens natively inside of Snowflake using Java UDFs. It's highly accurate, because the outcomes are based upon the entire dataset, not just a sample. Java UDFs can be invoked either using Snowpark, or SQL. Talend used both approaches: one during the prototyping phase, and the other as we operationalized the feature in our product Talend Data Inventory. Start your journey towards a culture of healthy data today by trying Talend on Snowflake Partner Connect.

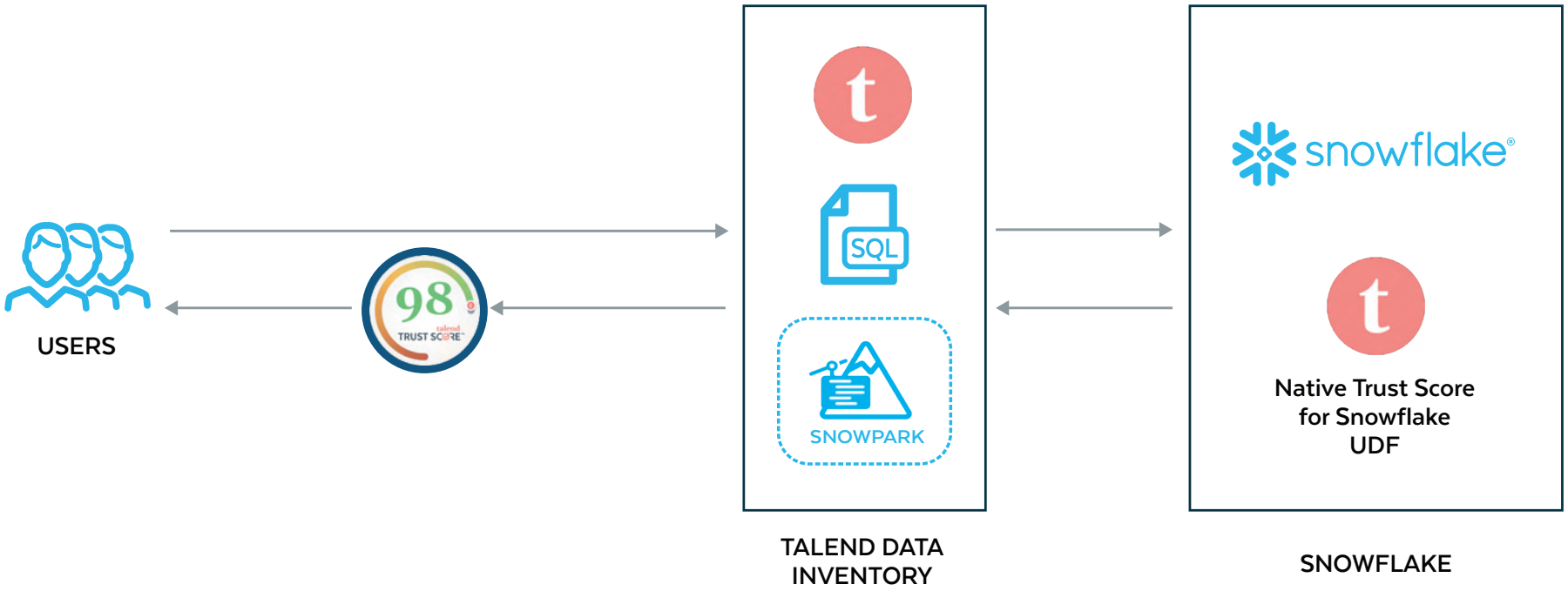


Figure 2: Delivering trusted data with Snowflake and Talend [1]

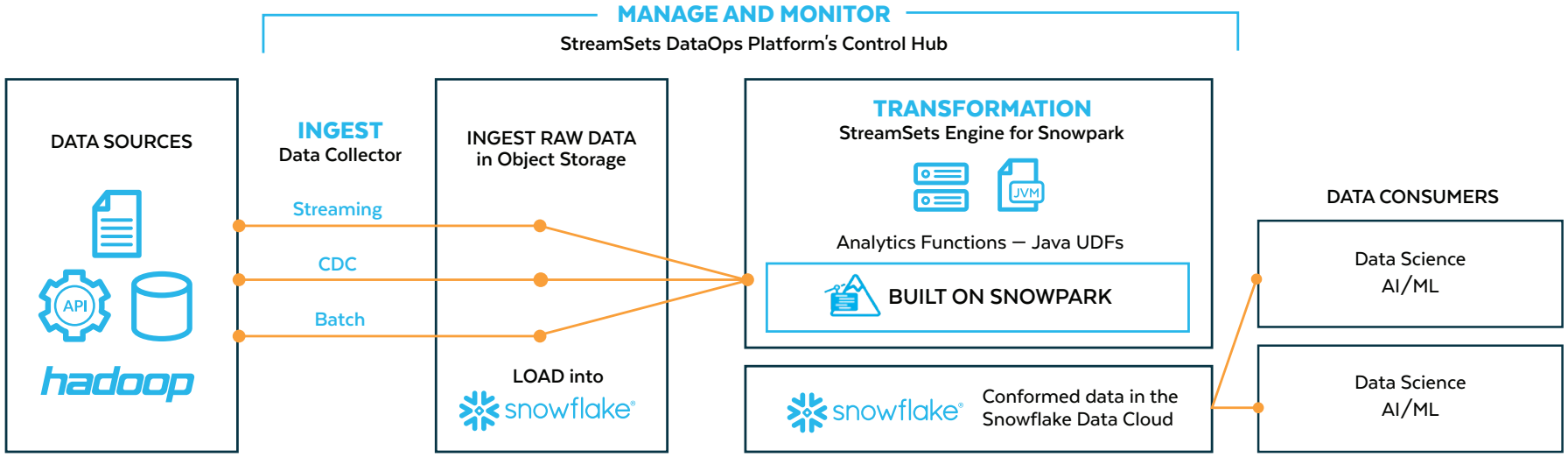


Figure 3: Optimizing DataOps with the StreamSets engine for Snowpark.[3]

USE CASE: OPTIMIZING DATAOPS WITH THE STREAMSETS ENGINE FOR SNOWPARK

The StreamSets engine for Snowpark builds on top of Snowpark to enable both the expressiveness and flexibility of Snowpark's multi-language support as well as the simplicity of Data Cloud operations. Data engineers can go beyond SQL to express powerful data pipeline logic with the StreamSets DataOps Platform.

Using Scala or Java through an intuitive GUI, you can use no code or you can drop in code when you want. The StreamSets engine for Snowpark includes the benefits of the StreamSets DataOps Platform—built-in monitoring and orchestration of complex data pipelines at scale—all in the cloud and with no additional hardware required.

USE CASE: DEVELOPMENT LIFECYCLE MANAGEMENT WITH DATAOPS.LIVE

An expanded partnership between DataOps.live and Snowflake enables Snowpark workloads to be fully managed alongside all other objects in Snowflake throughout the development, test, and production lifecycle. In addition to all the normal "source of truth for everything data" that is stored in the DataOps.live Git repository, the repository also provides for all the usual software development requirements. With DataOps.live, you can branch, version, compile, test, and deploy software and produce artifacts just like you do for any other software project.

Running the Snowpark API really means running a Snowpark application in which the Snowpark libraries are being used. For example, if Scala is being used, then an environment is needed with all the runtime tools for that specific language and the appropriate libraries (plus the Snowpark libraries themselves).

In many cases, a Snowpark application will be used to do advanced data manipulation, in particular manipulation beyond what can be achieved within SQL, but the results will still be stored back into Snowflake. In these cases, the automated data testing within the DataOps.live's modelling and transformation engine can be used to validate the results of the Snowpark application.

SNOWPARK USE CASES FOR DATA SCIENCE

The data programmability advancements in Snowpark provide greater flexibility and extensibility, so data scientists can leverage their favorite programming languages, such as Python, to access, visualize, and process data as part of their machine learning workflows. That helps organizations maximize the value of data, including unstructured data and third-party data.

Snowflake's data science partners have taken the lead in leveraging these advancements to accelerate machine learning workflows with a more integrated experience. Snowpark enables data scientists to work from their notebook of choice, while being able to push down processing where data lives.

Snowpark use cases for data science include, but are not limited to:

- **Feature engineering**
- **ML model inference**
- **End-to-end ML in Snowflake with SQL**

USE CASE: PREPARING A DATAFRAME WITH SNOWPARK TO INVOKE AI TRAINING AND INFERENCING

Snowpark enables the data in Snowflake to be available as a DataFrame using Scala and Java code that is executed within the Snowflake environment.

Snowpark is designed to make building complex data pipelines easy and empower data engineers, data scientists, and developers who are using code as part of their notebook-based programming using Jupyter or in Dataiku, H2O.ai, and Zepl (now part of DataRobot's platform). With Snowpark, they can use their preferred language to accelerate feature engineering efforts by using familiar programming concepts such as DataFrames and then execute these workloads directly within Snowflake. For example, data scientists can define new features in H2O.ai notebooks and execute the processing in Snowflake to benefit from its scalability and performance.

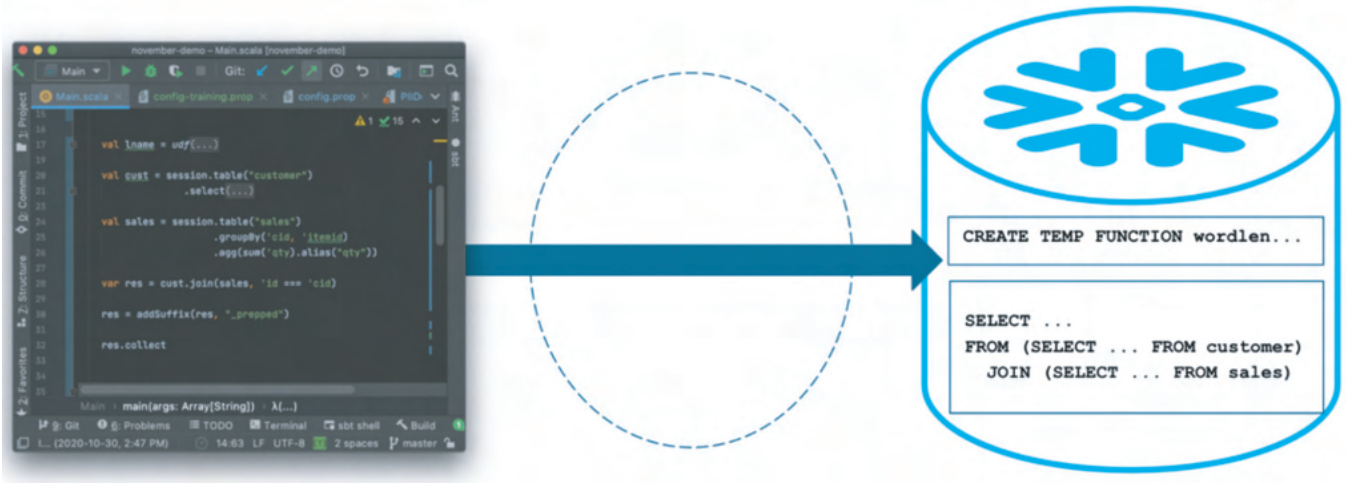


Figure 4: Preparing a DataFrame with Snowpark to invoke Driverless AI training in an H2O.ai notebook [7]

USE CASE: SCALING DATA PREPARATION FUNCTIONS AND PREDICTIVE MODEL SCORING WITH DATAIKU

Before Snowpark, several data preparation functions and predictive model scoring happened in Dataiku or other engines because they could not be expressed in SQL. That required data movement in and out of Snowflake, which impacted performance. Now, customers can take full advantage of Snowflake's performance and governance benefits by fully operating their Dataiku pipelines in Snowflake.

No-code and low-code users who want to build data preparation pipelines in Dataiku benefit from Snowflake's engine with Java UDFs. Java UDFs allow workloads expressed in Java to run in a JVM inside Snowflake.

Since Dataiku's core engine is Java, it's convenient for Dataiku to repackage data preparation functions and push down data preparation for execution inside Snowflake. That allows you to effortlessly scale to handle any number of users, jobs, or data while at the same time simplifying the architecture.

USE CASE: SIMPLIFYING THE PATH TO PRODUCTION WITH MODEL INFERENCE INSIDE SNOWFLAKE

To execute model inference at scale, operations teams typically need to move production data from a governed source of truth to a different environment to deploy the model. Moving large volumes of data has both cost and security implications that often prevent models from getting into production.

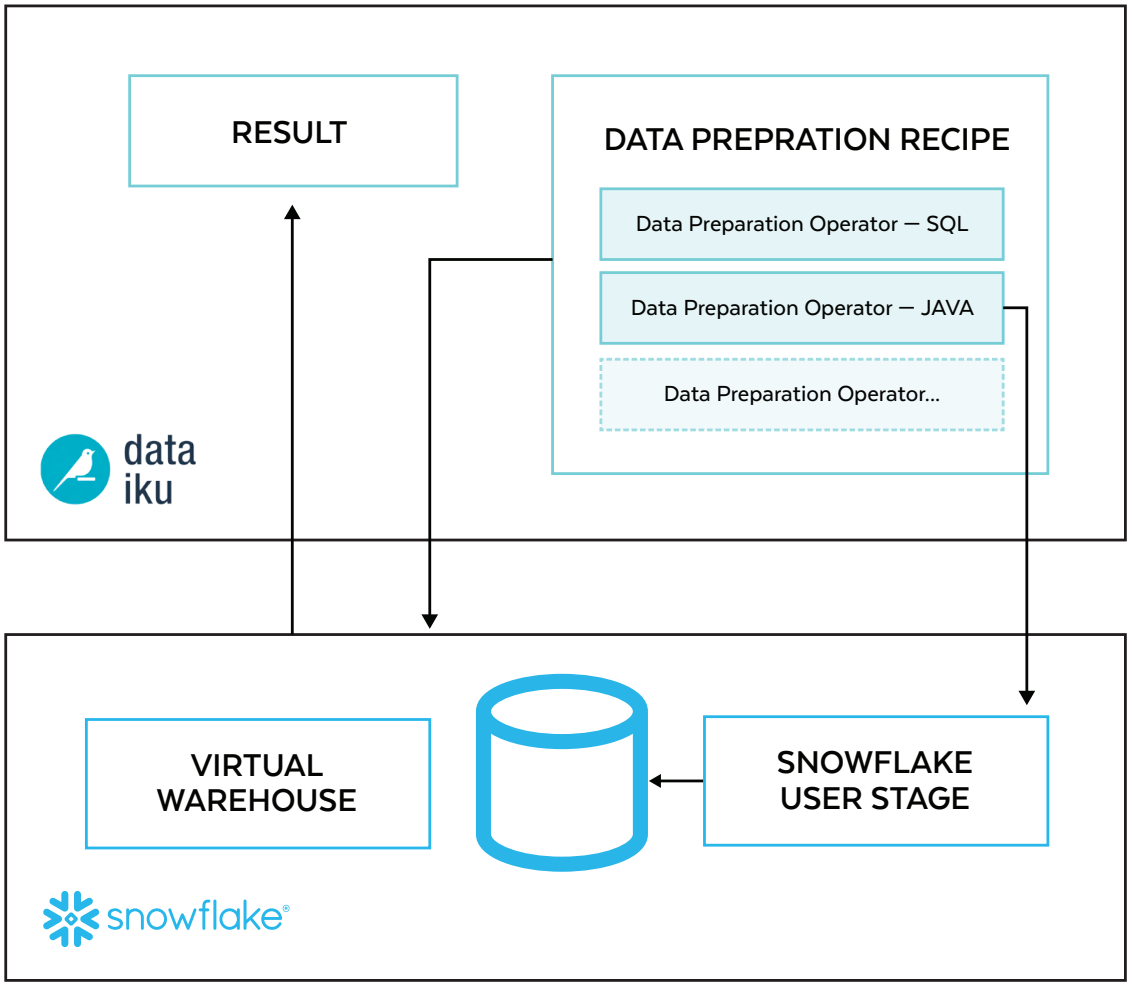


Figure 5: Scaling data preparation functions and predictive model scoring with Dataiku[8]

To address the complexity of moving data from where it lives, ML development and operations teams can shift to an approach where the model comes to the data using Java UDFs. Java UDFs can use trained models, expressed in Java, to run inference inside Snowflake.

Snowflake data science partners DataRobot and H2O.ai already have the functionality to export models and deploy them inside Snowflake for scalable, batch-oriented model inference, and call the external function for real-time inferencing.

DataRobot, the AI Cloud, delivers continuous AI lifecycle development and management for your Data Cloud. With DataRobot you can seamlessly connect and leverage data from Snowflake and transform and prepare that data for AI, perform automated Feature Discovery inside of Snowflake, build your AI models then export these models into production as JAR files and deploy them inside Snowflake for scalable model inference.

The DataRobot AI Cloud has flexible deployment options, including producing scoring code. With DataRobot, all models, in which Snowflake has a supporting library, can now be directly deployed inside a Snowflake Java UDF. This deployment option provides enhanced scoring speed beyond traditional model-scoring methods that previously ran outside of Snowflake.

Further, DataRobot users can ingest service and prediction data back into the DataRobot AI Cloud to analyze model drift over time. In addition, DataRobot will allow you to monitor any non-DataRobot created model deployed in Snowflake to help you maintain a single source of AI truth across your AI initiatives.

In one use case, H2O.ai combined its data set to predict loan defaults with a publicly available data set from LendingClub and a demographic data set from Snowflake Data Marketplace. That improved the model's accuracy and reduced scoring (inference time), as it could scale within the Snowflake environment. Once H2O.ai trained its model on its data plus the third-party data from Snowflake Data Marketplace, it was then able to import the model to where the data lives for easier deployment and more efficient inferencing, as the model is taken to where the data lives.

Finally, when you run a model inside Snowflake, you retain the ability to use your ML platform of choice for model monitoring. For example, DataRobot users can ingest service and prediction data back into DataRobot MLOps to analyze model drift over time.

In addition, Dataiku uses the same technology to package and deploy machine learning models in Snowflake therefore allowing predictive scoring to happen without moving the data out of Snowflake.

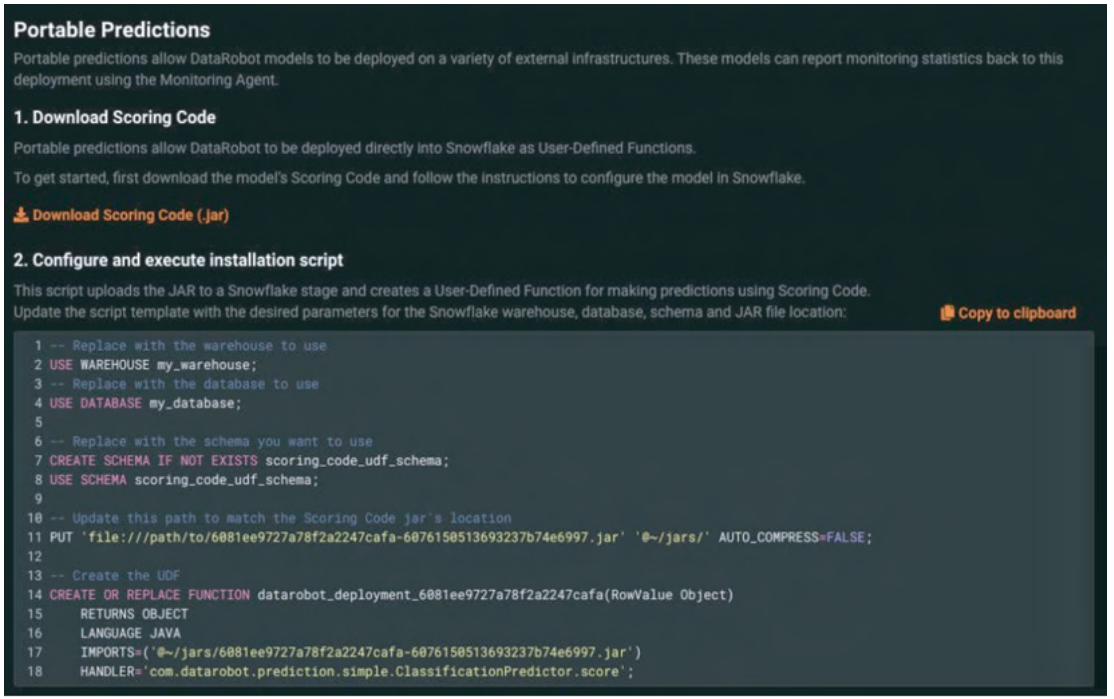


Figure 6: DataRobot-generated script to upload a JAR and create an associated UDF to perform inference directly in Snowflake[9]

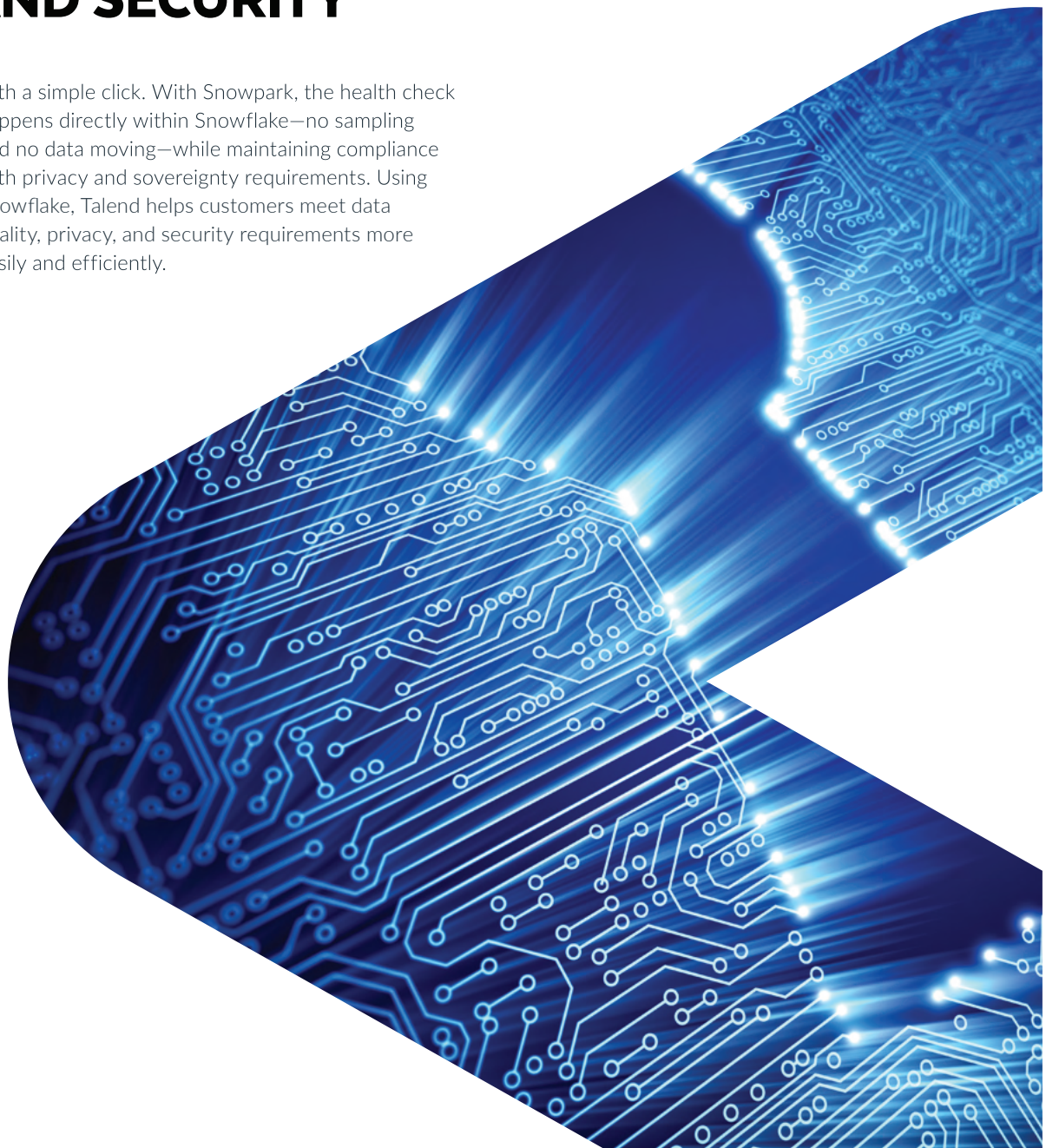
SNOWPARK USE CASES FOR DATA GOVERNANCE AND SECURITY

Snowpark can be used to build solutions that support data governance efforts. Below are a few examples of what can be built:

- **Data quality profilers:** Snowpark can be used to help organizations understand and improve the quality of their data. A function can scan data for completeness and detect anomalies to flag potential data quality issues.
- **Data classifiers:** Although Snowflake's classification capability automatically detects PII, Snowpark can be used to build the logic for classifying other kinds of data that's critical to an organization.
- **Format-preserving masking functions:** Snowpark can be used to support more-customized needs for Snowflake's Dynamic Data Masking feature. For example, you can create custom functions to mask or replace credit card numbers.

USE CASE: DELIVERING TRUSTED DATA WITH TALEND

Some of the data engineering use cases mentioned earlier are very close to data governance. With Snowpark, partners can run their software directly in Snowflake, simplifying customers' governance posture by reducing the need for data movement. For example, Snowflake and Talend customers can now use Talend Trust Assessor to perform an instant health check on all their Snowflake data with a simple click. With Snowpark, the health check happens directly within Snowflake—no sampling and no data moving—while maintaining compliance with privacy and sovereignty requirements. Using Snowflake, Talend helps customers meet data quality, privacy, and security requirements more easily and efficiently.



USE CASES FROM OTHER SNOWFLAKE PARTNERS

LTI

With Snowpark, LTI Mosaic[5] can simplify and amplify the following capabilities for Snowflake:

- **DataOps (the application of DevOps principles to data)**
- **MLOps (machine learning operations)**
- **ModelOps (a holistic method for rapidly and iteratively moving models through the analytics lifecycle)**

By leveraging Snowpark, Mosaic customers can deploy machine learning models, push model inferences, and write code that can be pushed down to Snowflake for processing.

pHData

phData customers want to perform machine learning tasks such as natural language processing (NLP) or image recognition on their Snowflake data. Traditionally this has been a complex process that you couldn't do with SQL alone. Using Snowpark, phData customers can now manage this complex pipeline inside Snowflake, significantly reducing cost and complexity.[6]

START YOUR SNOWPARK JOURNEY

To start using Snowpark, please see our documentation and step-by-step quickstart guide. We're eager to see the fantastic things you'll build.

Contributors:

Carlos Bouloy

Frank Pacione

Julian Forero

Miles Adkins

Paul Gantz

Shiyi Gu

ABOUT SNOWFLAKE

Snowflake delivers the Data Cloud—a global network where thousands of organizations mobilize data with near-unlimited scale, concurrency, and performance. Inside the Data Cloud, organizations unite their siloed data, easily discover and securely share governed data, and execute diverse analytic workloads. Wherever data or users live, Snowflake delivers a single and seamless experience across multiple public clouds. Snowflake's platform is the engine that powers and provides access to the Data Cloud, creating a solution for data warehousing, data lakes, data engineering, data science, data application development, and data sharing. Join Snowflake customers, partners, and data providers already taking their businesses to new frontiers in the Data Cloud. **Snowflake.com**.

CITATIONS

[1] bit.ly/2XAxZGU
[2] bit.ly/3nZClCA
[3] bit.ly/3tWMAbw
[4] bit.ly/3Avdr0V
[5] bit.ly/3nO8VY1
[6] bit.ly/3zyQpEZ
[7] bit.ly/3tLY1mh
[8] bit.ly/39bbue0
[9] bit.ly/3tNTkZh
[10] bit.ly/3zcZzqH
[11] bit.ly/3tLY1mh
[12] bit.ly/3kgmugA