snowflake®

THE DATA ENGINEER'S GUIDE TO PYTHON FOR SNOWFLAKE

CHAMPION GUIDES

EBOOK

TABLE OF CONTENTS

INTRODUCTION

Python consistently ranks in the top three most popular programming languages, and 68% of all developers surveyed said they "love" working in Python, according to Stack Overflow's Annual Developer survey.[1]

But for many years, data engineers have had to use separate tools for data transformations in Python versus other programming languages. Even with knowledge in those languages, setting up and managing separate compute environments for each one can be frustrating and time-consuming.

Snowpark is the set of libraries and runtimes that enable all data users to bring their work to the Snowflake Data Cloud with native support for Python, SQL, Java, and Scala. With Snowpark, data engineers can execute pipelines that feed ML models and applications faster and more securely in a single platform using their language of choice.

In the pages that follow, we'll discuss Snowpark and best practices for using Python within the Snowflake Data Cloud. You will learn how:

- Snowflake supports data engineering with Snowpark, its main benefits, and use cases
- Snowpark supports Python and other programming languages, in addition to SQL
- Data engineers can use Python efficiently and with impact within the Snowflake platform
- Snowpark fits into the larger Snowflake data engineering ecosystem

We'll also share resources designed to help data engineers get started with Snowflake and Snowpark.

SNOWPARK FOR DATA ENGINEERING

Snowpark is the set of libraries and runtimes that securely enable data engineers to deploy and process non-SQL code, including Python, Java and Scala as shown in Figure 1.

On the client side, Snowpark consists of libraries including the DataFrame API. Snowpark brings deeply integrated, DataFrame-style programming and OSS compatible APIs to the languages data practitioners like to use. It provides familiar APIs for various data centric tasks, including data preparation, cleansing, preprocessing, model training, and deployments tasks.

On the server side, Snowpark provides flexible compute and runtime constructs that allow users to bring in and run custom logic on warehouses or Snowpark Container Services (private preview). In the warehouse model, users can seamlessly run and operationalize data pipelines, ML models, and data applications with user-defined functions (UDFs) and stored procedures (sprocs). For workloads that require use of specialized hardware like GPUs, custom runtimes/libraries or hosting of long running full-stack applications, Snowpark Container Services offers the ideal solution.

Snowflake offers data engineers many benefits with Snowpark:

- A single platform that supports multiple languages, including SQL, Python, Java, and Scala
- Consistent security across all workloads with no governance trade-offs
- Faster, cheaper, and more resilient pipelines.

A SINGLE PLATFORM

Architecture complexity increases significantly when different teams use different languages across multiple processing engines. Snowpark streamlines architectures by natively supporting programming languages of choice, without the need for separate processing engines. Instead, Snowpark brings all teams together to collaborate on the same data in a single platform—Snowflake.

LIBRARIES AND RUNTIMES IN SNOWPARK

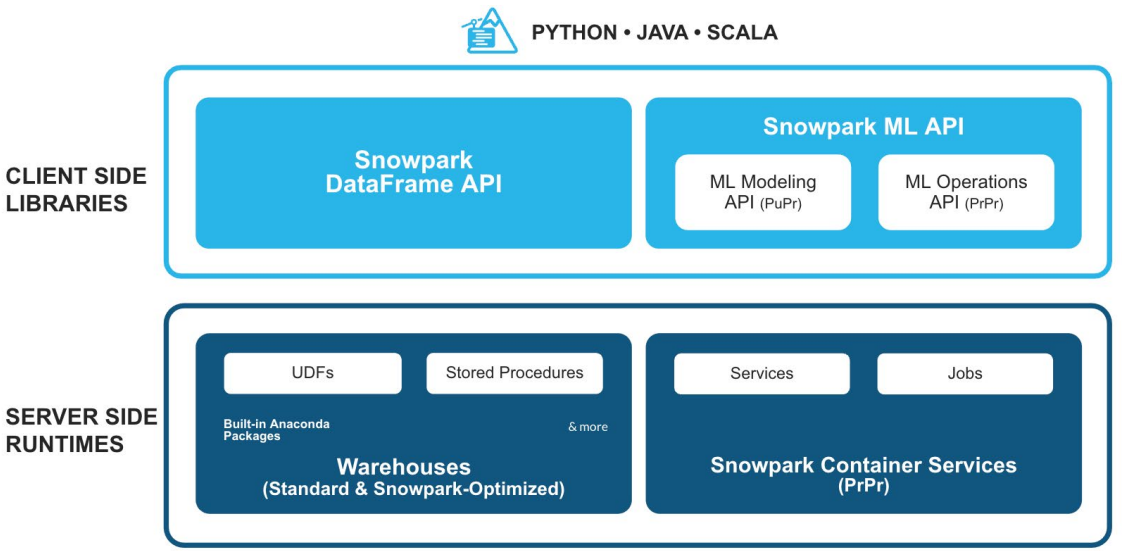

Figure 1: Snowpark allows developers working in many languages to leverage the power of Snowflake.

CUSTOMER SPOTLIGHT

HyperFinity

Snowflake customer, HyperFinity, a no-code decision intelligence platform for retailers and CPGs, uses SQL and Python for their ML and AI initiatives. With Snowpark, HyperFinity has a single platform that supports both languages, thereby eliminating cumbersome data movement and code developed to keep data movement across different services. As a result, HyperFinity works more seamlessly—developing, testing, and deploying Python and SQL in one environment for more agile overall operations.

NO GOVERNANCE TRADE-OFFS

Enterprise-grade governance controls and security are built into Snowflake. For example, Snowpark is secure with a design that isolates data to protect the network and host from malicious workloads, and gives administrators control over the libraries developers execute. Developers can build confidently, knowing data security and compliance measures are consistent and built-in.

CUSTOMER SPOTLIGHT

EDF

EDF, a supplier of gas and zero-carbon electricity to homes and businesses in the United Kingdom, tapped Snowpark to help deploy data applications. By working within Snowflake, the project did not require additional sign-offs and meetings to approve data accessibility. Instead, the EDF team could scale seamlessly by working within the security rules Snowflake enables and that applied to the project.

Since integrating Snowpark into its data engineering operations, EDF sped up production of customer-facing, ML-driven programs, from several months to just three to four weeks, increasing output up by 4x.

FASTER, CHEAPER PIPELINES

Snowpark enables pipelines with better price performance, transparent costs, and less operational overhead thanks to Snowflake's unique multi-cluster shared data architecture. Snowflake is a single, integrated platform that delivers the performance, scale, elasticity, and concurrency today's organizations require.

CUSTOMER SPOTLIGHT

IQVIA

These benefits can be seen at IQVIA, a leading provider of analytics, technology solutions, and clinical research services in the life sciences industry, and a Snowflake customer. As IQVIA processed increasingly large volumes of structured, semistructured, and unstructured data, the company had to manage mounting complexity as its business scaled.

Since implementing Snowpark in Snowflake, IQVIA has developed data engineering pipelines and intelligent apps more quickly and easily, with consistent enterprise-level governance features such as row-level access, data masking, and closer proximity of data to processing. By leveraging Snowpark to build their pipelines that process large volumes of data, IQVIA has realized a cost savings of 3x compared to previous pipeline processes.

SNOWFLAKE (AND SNOWPARK) FOR DATA ENGINEERING

Using Snowpark runtimes and libraries, data engineers can securely deploy and process Python code to build pipelines in Snowflake. Some of the critical use cases for data engineers working in Snowpark include:

- **ETL/ELT:** Data teams can use Snowpark to transform raw data into modeled formats regardless of type, including JSON, Parquet, and XML. All data transformations can then be packaged as Snowpark stored procedures to operate and schedule jobs with Snowflake Tasks or other orchestration tools.
- **Custom logic:** Users can leverage Snowpark's User Defined Functions (UDFs) to streamline architecture with complex data processing and custom business logic written in Python or Java in the same platform running SQL queries and transformations. There are no separate clusters to manage, scale, or operate.
- **Data science and ML pipelines:** Data teams can use the integrated Anaconda repository and package manager to collaborate in bringing ML data pipelines to production. Trained ML models can also be packaged as a UDF to run the model inference close to data, enabling faster paths from model development to production.

SNOWFLAKE FOR PYTHON

Using Snowpark for Python, data engineers can take advantage of familiar tools and programming languages while benefiting from the scale, security, and performance of the Snowflake engine. All processing is run in a secure Python runtime right next to your data, resulting in faster, more scalable pipelines with built-in governance regardless of the language used. Figure 2 gives an overview of both the Snowpark client side libraries and the Snowflake server side runtimes.

The diagram illustrates an overview of the Snowpark for Python architecture, which consists of DataFrames, UDFs, and stored procedures that can be developed from any client IDE or notebook. Their execution can all be pushed down to Snowflake to benefit from the performance, elasticity, and governance of the Snowflake processing engine.

Depending on what you develop, how the code is executed in Snowflake varies. First, there are DataFrame operations. Think of these as your transformations and operations on data such as filters, aggregations, joins, and other similar operations. These DataFrame operations are converted into SQL to leverage the proven performance of Snowflake to distribute and scale the processing of that data.

SNOWPARK FOR PYTHON ARCHITECTURE

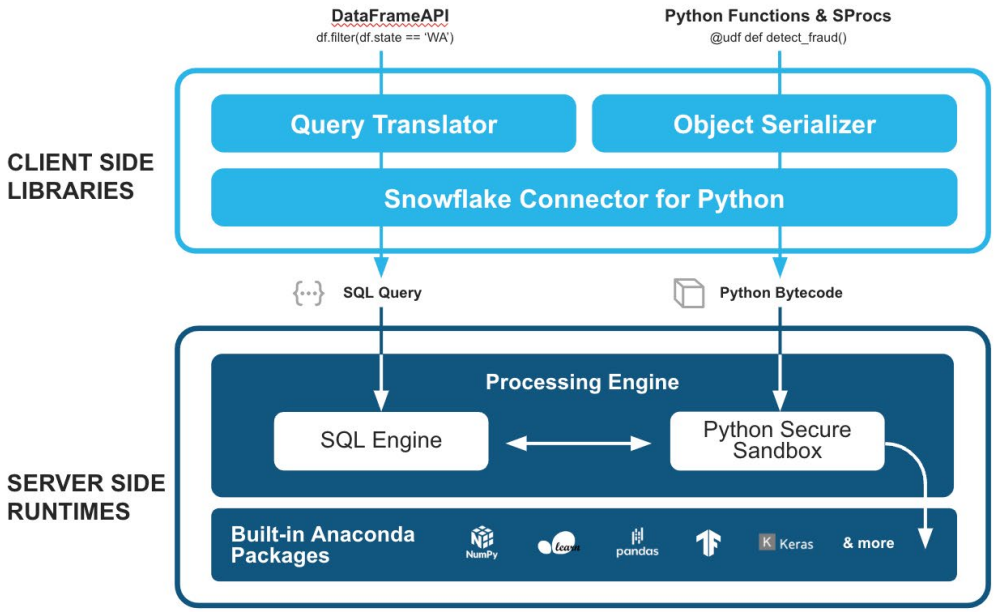

Figure 2: Snowpark DataFrames and Python functions work seamlessly together.

For custom Python or Java code, there is no translation to SQL. Rather the code is serialized and sent to Snowflake to be processed inside the Java or Python Secure Sandbox. In the case of Python, if the custom code includes any third-party open source libraries available in the integrated Anaconda package repository, the package manager can help ensure code runs without complex environment management.

SNOWPARK CLIENT SIDE LIBRARIES

The Snowpark client side libraries are open source and work with any Python environment. This includes the Snowpark DataFrame API, which allows data engineers to build queries using DataFrames right in their Python code, without having to create and pass along SQL strings.

SNOWPARK DATAFRAME API

Snowpark brings deeply integrated, DataFrame-style programming to the languages that data engineers prefer to use. Data engineers can build queries in Snowpark using DataFrame-style programming in Python, using their IDE or development tool of choice. Behind the scenes, all DataFrame operations are transparently converted into SQL queries that are pushed down to the Snowflake scalable processing engine. Because DataFrames use first-class language constructs, engineers also benefit from support for type checking, IntelliSense, and error reporting in their development environment.

SNOWFLAKE SERVER SIDE RUNTIME

Snowflake is cloud-built as a data platform that architecturally separates but logically integrates storage and compute, and optimized to enable near-limitless amounts of these resources. Elastic scaling, multi-language processing, and unified governance also underpin Snowflake's architecture.

The intelligent infrastructure is what makes everything just work. Compute clusters can be started, stopped, or resized—automatically or on the fly—accommodating the need for more or less compute resources at any time. Along with flexibility, Snowflake prioritizes speed, granting near-instant access to dedicated compute clusters for each workload, so users can take advantage of near-limitless concurrency without degrading performance. The three architectural layers that integrate within Snowflake's single platform are shown in Figure 3.

The Snowpark Python server-side runtime makes it possible to write Python UDFs and Stored Procedures that are deployed into Snowflake's secured Python runtime. UDFs and stored procedures are two other key components of Snowpark that allow data engineers to bring custom Python logic to Snowflake's compute engine, while taking advantage of open source packages pre-installed in Snowpark.

THE SNOWFLAKE PLATFORM ARCHITECTURE

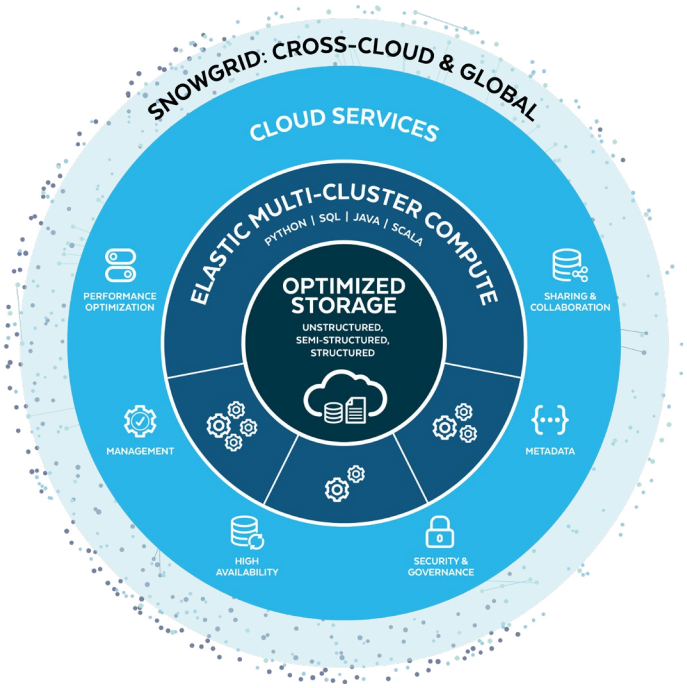

Figure 3: Snowflake's single platform integrates three unique architectural layers.

SNOWPARK USER DEFINED FUNCTIONS (UDFS)

Custom logic written in Python runs directly in Snowflake using UDFs. Functions can stand alone or be called as part of a DataFrame operation to process the data. Snowpark takes care of serializing the custom code into Python byte code and pushes all of the logic to Snowflake, so it runs next to the data. To host the code, Snowpark has a secure, sandboxed Python runtime built right into the Snowflake engine. Python UDFs scale out processing associated with the underlying Python code, which occurs in parallel across all threads and nodes, and comprising the virtual warehouse on which the function is executing.

There are several types of UDFs that data engineers can use in Snowpark, including:

- **Scalar UDFs:** Operate on each row in isolation and produce a single result
- **Vectorized UDFs:** Receive batches of input rows as Pandas DataFrames and return batches of results as Pandas arrays or series
- **User-Defined Table Functions:** Return multiple rows for each input row, return a single result for a group of rows, or maintain state across multiple rows

To the right is an example of a Snowpark UDF used to calculate the distance between a distribution center and shipping locations.

```
#Given geo-coordinates, UDF to calculate distance between
distribution center and shipping locations


from snowflake.snowpark.functions import udf
import geopandas as gpd
from shapely.geometry import Point


@udf(packages=['geopandas'])
def calculate_distance(lat1: float, long1: float, lat2: float, long2:
float)-> float:
  points_df = gpd.GeoDataFrame({'geometry': [Point(long1, lat1),
Point(long2, lat2)]}, crs='EPSG:4326').to_crs('EPSG:3310')
  return points_df.distance(points_df.shift()).iloc[1]


# Call function on dataframe containing location coordinates
distance_df = loc_df.select(loc_df.sale_id, loc_df.distribution_
center_address, loc_df.shipping_address, \
   calculate_distance(loc_df.distribution_center_lat, loc_
df.distribution_center_lng, loc_df.shipping_lat, loc_df.shipping_
lng) \
   .alias('distribution_center_to_shipping_distance'))
```

STORED PROCEDURES

Snowpark stored procedures help data engineers operationalize their Python code and run, orchestrate, and schedule their pipelines. A stored procedure is created once and can be executed many times with a simple CALL statement in your orchestration or automation tools. Snowflake supports stored procedures in SQL, Python, Java, Javascript, and Scala so data engineers can easily create polyglot pipelines.

To use a stored procedure, developers can use the sproc() function in Snowpark to bundle the Python function and have Snowpark deploy it on the server side. Snowpark will serialize the Python code and dependencies into bytecode and store them in a Snowflake stage automatically. They can be created either as a temporary (session-level) or permanent object in Snowflake.

Stored procedures are single-node, which means transformations or analysis of data at scale inside a stored procedure should leverage the DataFrame API or other deployed UDFs to scale compute across all nodes of a compute cluster.

To the right is a simple example of how to operationalize a Snowpark for Python pipeline that calculates and applies a company's sales bonuses on a daily basis.

```
-- Create python stored procedure to host and run the snowpark pipeline
to calculate and apply bonuses
create or replace procedure apply_bonuses(sales_table string, bonus_table
string)
  returns string
  language python
  runtime_version = '3.8'
  packages = ('snowflake-snowpark-python')
  handler = 'apply_bonuses'
AS
$$
from snowflake.snowpark.functions import udf, col
from snowflake.snowpark.types import *

def apply_bonuses(session, sales_table, bonus_table):
  session.table(sales_table).select(col("rep_id"), col("sales_amount")*0.1).
write.save_as_table(bonus_table)
  return "SUCCESS"
$$;

--Call stored procedure to apply bonuses
call apply_bonuses('wholesale_sales','bonuses');

— Query bonuses table to see newly applied bonuses
select * from bonuses;

— Create a task to run the pipeline on a daily basis
create or replace task bonus_task
warehouse = 'xs'
schedule = '1440 minute'
as
call apply_bonuses('wholesale_sales','bonuses');
```

SNOWFLAKE, ANACONDA, AND THE OPEN SOURCE ECOSYSTEM

One of the benefits of Python is its rich ecosystem of open-source packages and libraries. In recent years, open-source packages have been one of the biggest enablers for faster and easier data engineering. To leverage open-source innovation, Snowpark has partnered with Anaconda for a product integration without any additional cost to the user beyond warehouse usage.

Data engineers in Snowflake are now able to speed up their Python-based pipelines by taking advantage of the seamless dependency management and comprehensive set of curated open-source packages provided by Anaconda—all without moving or copying the data. All Snowpark users can benefit from thousands of the most popular packages that are pre-installed from the Anaconda repository, including fuzzy wuzzy for string matching, h3 for geospatial analysis, and scikit-learn for machine learning and predictive data analysis. Additionally, Snowpark is integrated with the Conda package manager so users can avoid dealing with broken Python environments because of missing dependencies.

Using open-source packages in Snowflake is as simple as the code below, which demonstrates how users can call packages such as NumPy, XGBoost, and Pandas, directly from Snowpark.

Snowpark also fully supports dbt, one of the most popular solutions for data transformation today. It supports a SQL-first transformation workflow, and in 2022, dbt introduced support for Python. With dbt's support for both SQL and Python, users can write transformations in the language they find most familiar and fit for purpose. And dbt on Snowpark allows analyses using tools available in the open source Python ecosystem, including state-of-the-art packages for data engineering and data science, all within the dbt framework familiar to many SQL users. It supports a SQL-first workflow, and in 2022, dbt introduced Python as a second language running Snowpark under the hood to perform analyses using tools available in the open-source Python ecosystem.

```
-- Returns an array of the package versions of NumPy, Pandas, and XGboost
create or replace function py_udf()
returns array
language python
runtime_version = 3.8
packages = ('numpy','pandas==1.4.*','xgboost==1.5.0')
handler = 'udf'
as $$
import numpy as np
import pandas as pd
import xgboost as xgb
def udf():
  return [np.__version__, pd.__version__, xgb.__version__]
$$;
```

BEST PRACTICES: DATA ENGINEERING IN SNOWPARK WITH PYTHON

As the Snowpark for Python developer community grows rapidly, data engineers are looking for "best practices" to guide their work. Understanding how Snowpark DataFrames, UDFs, and stored procedures work together can make data engineers' work in Snowflake more efficient and secure. We've compiled a short list of best practices for data engineers working with Python in Snowpark.

1. **Maximize use of the Snowpark libraries for development and secure execution.**

   Snowpark can be used with your preferred IDE and development and debugging tools, and the execution can be transparently pushed down to Snowflake. Maximize this utility while being mindful of the use of to_pandas() from Snowpark, which brings full data into memory. Also, Cachetools is a Python library that provides a collection of caching algorithms to store a limited number of items for a specified duration. They can be used to speed up UDFs and stored procedures by ensuring the logic is cached in memory in cases of repeated reads.

2. **Accelerate development to production flow with Anaconda integration.**

   We recommend using the Snowflake Anaconda channel for local development to ensure compatibility between client- and server-side operations. Building your code using the latest stable versions of third-party packages doesn't require users to specify dependencies because the Conda Package Manager takes care of this, offering tremendous peace of mind. If a desired package is not available inside Snowflake, please submit feedback through the Snowflake Community so Snowflake teams can facilitate its integration. If the package is a pure Python package, you can unblock yourself and bring in the package via Stages.

3. **Use vectorized UDFs for feature transformations and ML scoring.**

   Vectorized UDFs using the Batch API can execute scalar UDFs in batches. Use the Python UDF Batch API if leveraging third-party Python packages where transformations are independently done row by row and the process could be efficiently scaled out by processing rows in batches. This is a common scenario when using third-party Python packages to do machine learning-specific transformations on data as part of feature engineering or when executing ML batch inference.

4. **Use Snowpark-optimized warehouses for memory-intensive workloads.**

   Snowpark-optimized warehouses are important for data engineers working on large data sets. Consider using a Snowpark-optimized warehouse when you run into a `100357 (P0000): UDF available memory exhausted` error during development. Avoid mixing other workloads with workloads that require Snowpark-optimized warehouses. If you must mix them, consider calling the `session.use_warehouse()` method to switch back to standard warehouses.

BEYOND SNOWPARK: OTHER CAPABILITIES IN THE SNOWFLAKE DATA ENGINEERING ECOSYSTEM

In addition to Snowpark, Snowflake has many other data engineering capabilities that make it a fast and flexible platform that comprehensively supports simple, reliable data pipelines in any language of choice.

Figure 4 offers an overview of Snowflake's advanced functionality for ingestion, transformation, and delivery that simplify data engineering.

Snowflake allows data engineering teams to ingest all types of data using a single platform, including streaming or batch and structured, semi-structured, or unstructured. Supported data formats include JSON, XML, Avro, Parquet, ORC, and Iceberg. Streaming data, including streams from Apache Kafka topics, can also be ingested directly to a Snowflake table with Snowpipe Streaming. Thanks to the Data Cloud, all this data can be accessed and shared across providers and between internal teams, customers, partners, and other data consumers via the Snowflake Marketplace.

Data can be transformed using the data engineer's language of choice using Snowpark. Tasks can be combined with table streams for continuous ELT workflows to process recently changed table rows. Tasks are easily chained together for successive execution to support more complex periodic processing. All of this can be done fast, and scaled to meet the evolving number of users, data, and jobs of complex projects.

Snowflake is constantly enhancing functionality. Dynamic tables, which provide a way to build declarative pipelines, are currently in private review and offer a different approach to building pipelines from Snowpark for Python UDFs and Stored Procedures. These tools are designed to automatically process data incrementally as it changes to simplify data engineering workloads. Snowflake automates all the database objects and data manipulation language management, enabling data engineers to easily build scalable, performant, and cost-effective data pipelines.

The resulting data pipelines have intelligent infrastructure, pipeline automation, and data programmability. Snowflake's simplified pipelines then power analytics, applications, and ML models with only one copy of data to manage and near-zero maintenance. Data can also be accessed and shared directly using secure data sharing capabilities with internal teams, customers, partners, and even more data providers and consumers through the Snowflake Marketplace. Data doesn't move with Snowflake's modern data sharing technology. Instead a data provider grants a data consumer near-instant access to live, read-only copies of the data. This approach reduces latency, removes the need to copy and move stale data, and dramatically reduces the governance challenges of managing multiple copies of the same data.

Snowflake was designed to give data engineers access to all data at speed with performance and reliability at scale to build radically simple data pipelines. With innovative pipeline automation and data programmability, data engineers can simplify their workflows and eliminate what's unnecessary, so they can focus their effort on their most impactful work.

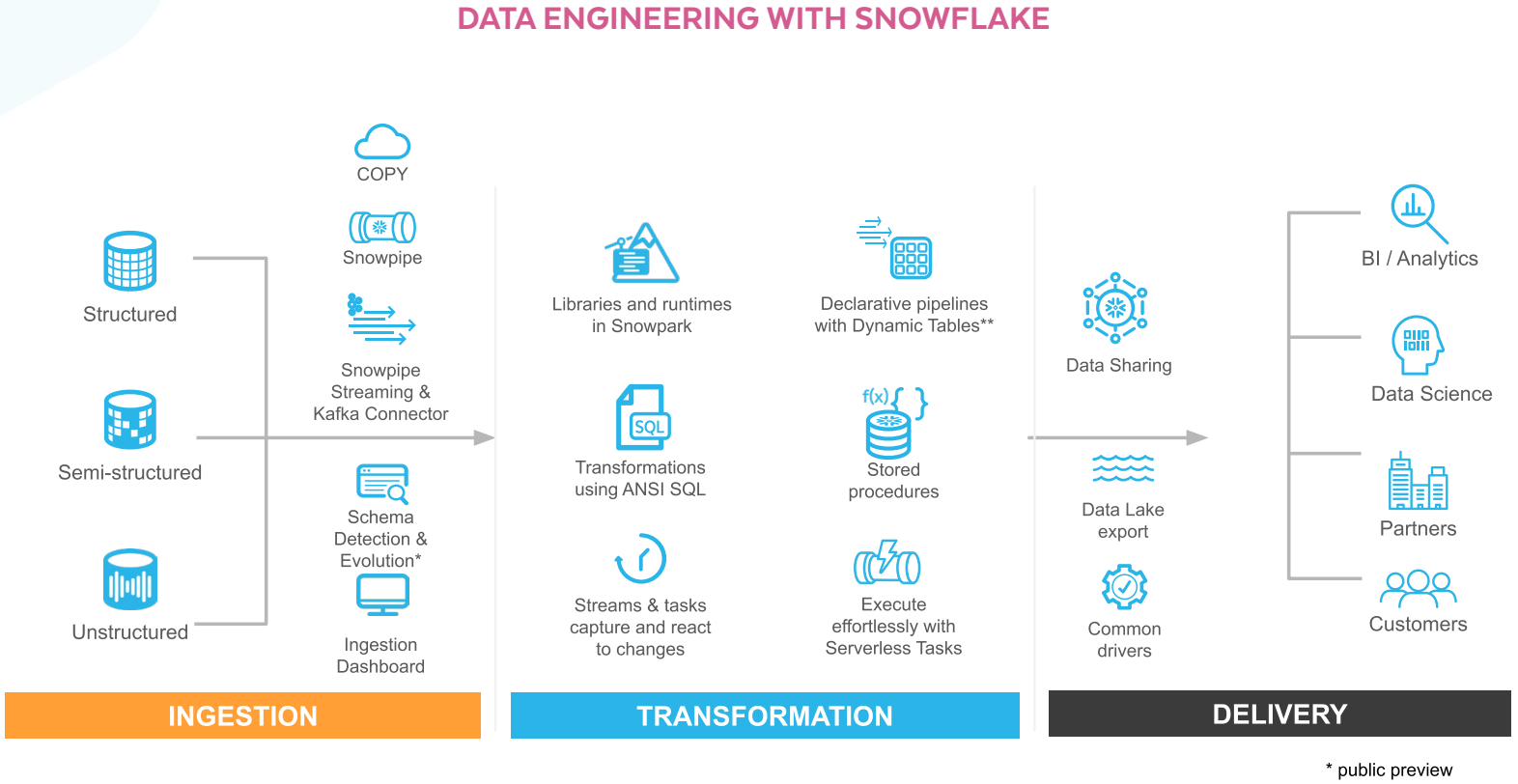

Figure 4: Snowflake supports ingestion of unstructured, semi-structured, and structured data while automated workflows facilitate transformation and delivery.

GETTING STARTED WITH SNOWPARK

To develop and deploy code with Snowpark, developers have always had the flexibility to work from their favorite integrated development environment (IDE) or notebook.

Data engineers can easily get started in Snowpark, beginning development anywhere that can run a Python kernel. Minimizing learning curves by eliminating the need for a new tool, data engineers simply install the Snowpark DataFrame API and establish a connection to their Snowflake account.

Snowpark aims to give developers flexibility. It supports many development interfaces, including:

- **Code editors and IDEs:** Many data engineers prefer to build using code editors and IDEs. These offer capabilities such as local debugging, autocomplete, and integration with source control. Snowpark works well in VS Code, IntelliJ, PyCharm, and other tools. VS Code works with a Jupyter extension that provides a notebook experience within the editor, bringing in breakpoints and debugging to the notebook experience, without requiring separate management of the Jupyter container or runtime. Code editors and IDEs are a great choice for rich development and testing experience for building pipelines.
- **Snowsight worksheets:** Snowsight is Snowflake's web interface that provides SQL and Python (currently in public preview) support in a unified, easy-to-use experience. These worksheets provide autocomplete for the Snowpark session and can run directly from the browser as a stored procedure. Snowsight is a good option for teams looking for a zero-install editor for writing and running Snowpark and quickly turning that code into Stored Procedures that can be orchestrated as part of an automated pipeline.
- **Open-source notebook solutions:** One popular option for building pipelines in Snowpark is to leverage notebooks. Notebooks enable rapid experimentation using cells. With Snowpark, you can run a variety of notebook solutions such as Jupyter Notebooks, which can be run locally while connected securely to Snowflake to execute data operations. Any machine running containers or Python can build and execute Snowpark pipelines. A similar approach can be used for working with Snowpark in other notebook solutions, including Apache Zeppelin. Open source notebook solutions are a great choice for data exploration.
- **Partner integrated solutions:** Many Snowpark Accelerated partners offer either hosted open source notebooks or their own integrated experiences. Their solutions include out-of-the-box Snowpark APIs preinstalled and offer secure data connections. These deeply integrated experiences speed up the building and deploying of pipelines, models, and apps. More information on partner integrations can be found on the Snowpark Accelerated page.

RESOURCES

Start harnessing the power of Snowflake with Snowpark for data engineering, and get started with the resources below:

- FREE TRIAL
- QUICKSTART
- DEVELOPER DOCUMENTATION
- MEDIUM BLOG
- SNOWFLAKE FORUMS

ABOUT SNOWFLAKE

Snowflake enables every organization to mobilize their data with Snowflake's Data Cloud. Customers use the Data Cloud to unite siloed data, discover and securely share data, power data applications, and execute diverse AI/ML and analytic workloads. Wherever data or users live, Snowflake delivers a single data experience that spans multiple clouds and geographies. Thousands of customers across many industries, including 590 of the 2022 Forbes Global 2000 (G2K) as of April 30, 2023, use Snowflake Data Cloud to power their businesses.

Learn more at **snowflake.com**

CITATIONS

[1] https://insights.stackoverflow.com/survey