ATSCALE snowflake

AtScale + Snowflake

Bridging Business Intelligence & Data Science in the Data Cloud

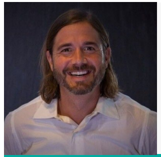
Today's Speakers

Simon Field

Snowcat Technical Director
Snowflake

Simon works in Snowflakes Customer Acceleration Team (SnowCAT), supporting customers to utilise new and advanced product capabilities within Snowflakes Data Cloud to improve the value they derive from their data.

Simon has worked in the field of Advanced Analytics, Data Warehousing, Big Data and Data Science for over 30 years, helping organisations make the transition to data-driven decision making.



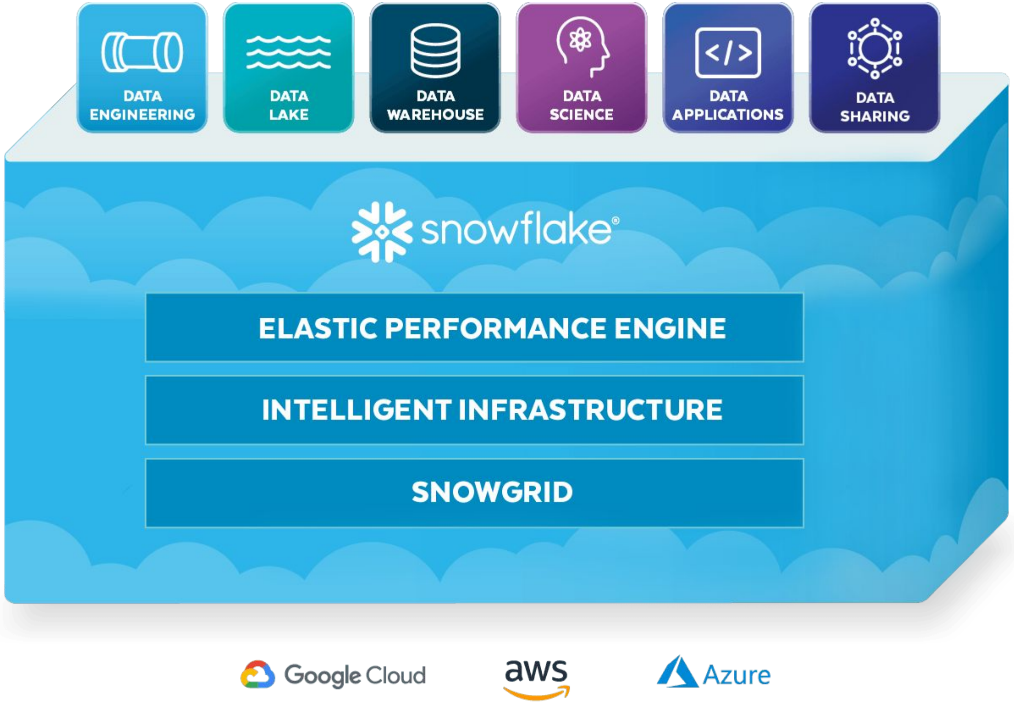
Daniel Gray

VP, Solutions Engineering
AtScale

Daniel brings rich experience in technical solutions engineering as well as software engineering to his work with global enterprise organizations.

Prior to joining AtScale to lead the Solutions Engineering team, Daniel spent many years in the analytics space including Hewlett-Packard's Advanced Technology Center, Vertica, and Domino Data Lab.

SNOWFLAKE PLATFORM



DATA ENGINEERING

DATA LAKE

DATA WAREHOUSE

DATA SCIENCE

DATA APPLICATIONS

DATA SHARING

snowflake®

ELASTIC PERFORMANCE ENGINE

INTELLIGENT INFRASTRUCTURE

SNOWGRID

DATA SOURCES

OLTP DATABASES

ENTERPRISE APPLICATIONS

THIRD-PARTY

WEB/LOG DATA

IoT

DATA CONSUMERS

DATA MONETIZATION

OPERATIONAL REPORTING

AD HOC ANALYSIS

REAL-TIME ANALYTICS

DATA SCIENCE WITH SNOWFLAKE

WORKFLOW

Collect Data ▶ Visualize, Explore & Understand ▶ Feature Engineering & Transformation ▶ Train ▶ Deploy ▶ Monitor

SNOWFLAKE FEATURES

Collect Data

Data Sharing & Data Marketplace for access to external datasets

Schema-on-Read for semi-structured data (eg. JSON)

Process streaming data using **Kafka Connector** and **Snowpipe**

Visualize, Explore & Understand

Query cloud storage without loading data using **External Tables**

Dashboards & visualizations using **Snowsight** & BI partners

ATSCALE

Feature Engineering & Transformation

Quick & easy feature engineering using **ANSI SQL Views**

Functional data engineering in Scala/ Python via **Snowpark***

Data enrichment & pipeline orchestration via **External Functions**

Transformation using **Streams and Tasks**

Private sandboxes without duplicating data with **Zero-Copy Cloning**

Train / Deploy

Extensive Partner Ecosystem

Amazon SageMaker, DataRobot, H2O.ai, dataiku, zepl, alteryx The Thrill of Solving

Bulk ML inference using **Java UDFs***

Built-in support for common drivers (ODBC, JDBC, Python, and more)

Easily Connect Your ML Toolchain

TensorFlow, PyTorch, Keras, scikit learn, Apache Spark, RStudio, XGBoost

Monitor

Model Management

Amazon SageMaker, DataRobot, H2O.ai, dataiku, zepl, alteryx The Thrill of Solving

Persist predictions and ground-truth in Snowflake for easy evaluation

SNOWFLAKE PLATFORM

One platform, one copy of data, many workloads

Secure and governed access to all data

Near-zero maintenance, as a service

Unlimited performance and scale

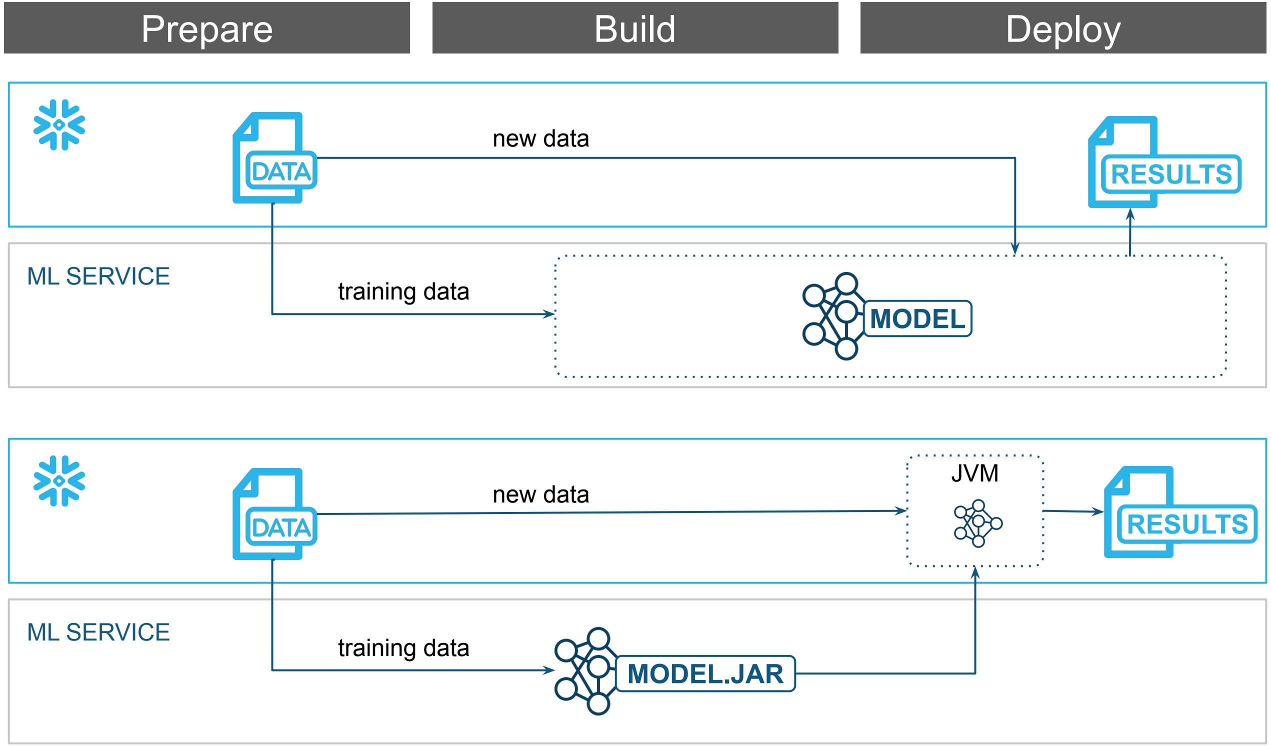
Model Inference : External Functions or Java UDFs

EXTERNAL SERVICE

Data continuously travels to externally hosted model (REST API)
E.g. AWS Lambda

WITH JAVA UDF

Model packaged as java file (.jar) runs where data lives



Prepare

Build

Deploy

DATA

new data

RESULTS

ML SERVICE

training data

MODEL

DATA

new data

JVM

RESULTS

ML SERVICE

training data

MODEL.JAR

ML partners with .JAR models: DataRobot, Dataiku, H2O or bring your own

PRIVATE | **PUBLIC** | GA

SNOWPARK

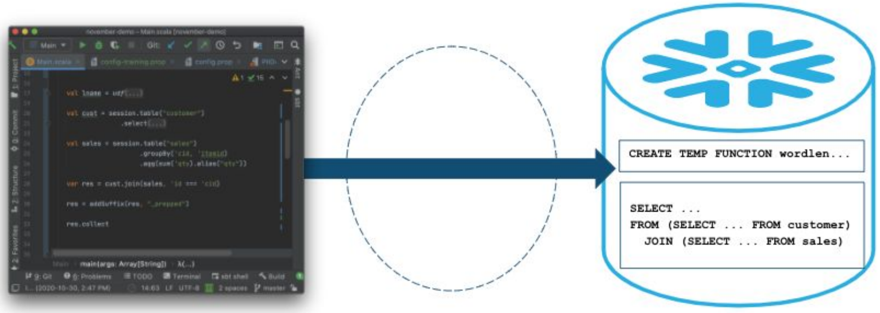
A new developer experience that allows you to write functional code and execute it directly within Snowflake

Example Use Cases:

- Data transformation
- Data preparation and feature engineering
- ML Scoring / Inference to operationalize ML models in data pipelines
- ELT systems
- Data apps

Allows coders to:

- Write in their preferred language and tools
- Easily complete and debug data pipelines with familiar constructs such as DataFrames, functions and use third-party libraries.
- Pushes all processing into Snowflake and eliminates the need to have other processing systems



```
val lname = udf(...)

val cust = session.table("customer")
              .select(...)

val sales = session.table("sales")
               .groupBy('cid, 'itemid)
               .agg(sum('qty).alias("qty"))

var res = cust.join(sales, 'id === 'cid)

res = addSuffix(res, "_prepped")

res.collect
```

```
CREATE TEMP FUNCTION wordlen...
```

```
SELECT ...
FROM (SELECT ... FROM customer)
  JOIN (SELECT ... FROM sales)
```

november-demo – Main.scala [november-demo]
Main
Git:
Main.scala
config-training.prop
config.prop
main(args: Array[String])
9: Git
6: Problems
TODO
Terminal
sbt shell
Build
14:63 LF UTF-8 2 spaces master

Snowpark pushes all of its operations directly to Snowflake without the need for Spark or any other intermediary.

**Support for Scala is in Public Preview. Plan to add other languages in future.*

PRIVATE | **PUBLIC** | GA

SNOWFLAKE JAVA FUNCTIONS

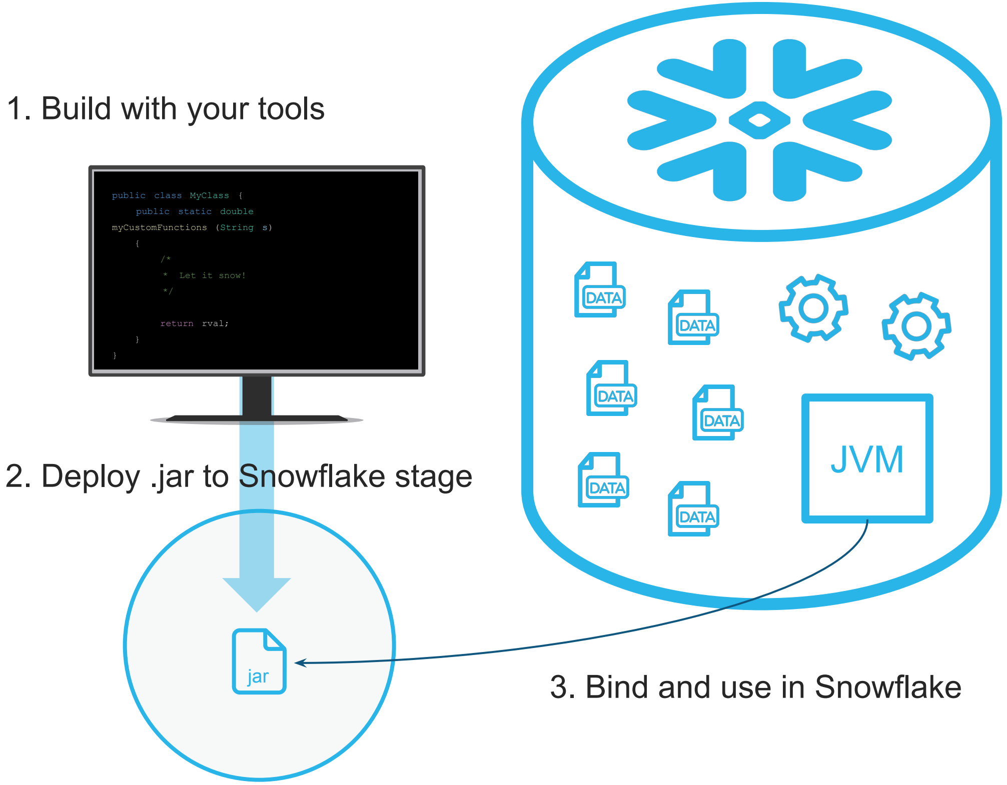
Transform and augment your data using custom logic running right next to your data, with no need to manage a separate service.

Example Scenarios:

- ML Scoring
- Apply custom code
- Use third-party libraries

Benefits:

- Developers can build custom functionality in Snowflake using the JVM languages and popular libraries.
- Snowpark ‘publishes’ functions developed in Scala as UDFs for execution in Snowflake via SQL or the Snowpark API.
- Users can access this functionality as if it were built in functions in Snowflake.
- Administrators can rest easy: data never leaves Snowflake and access and execution permissions for functions can be controlled.



1. Build with your tools

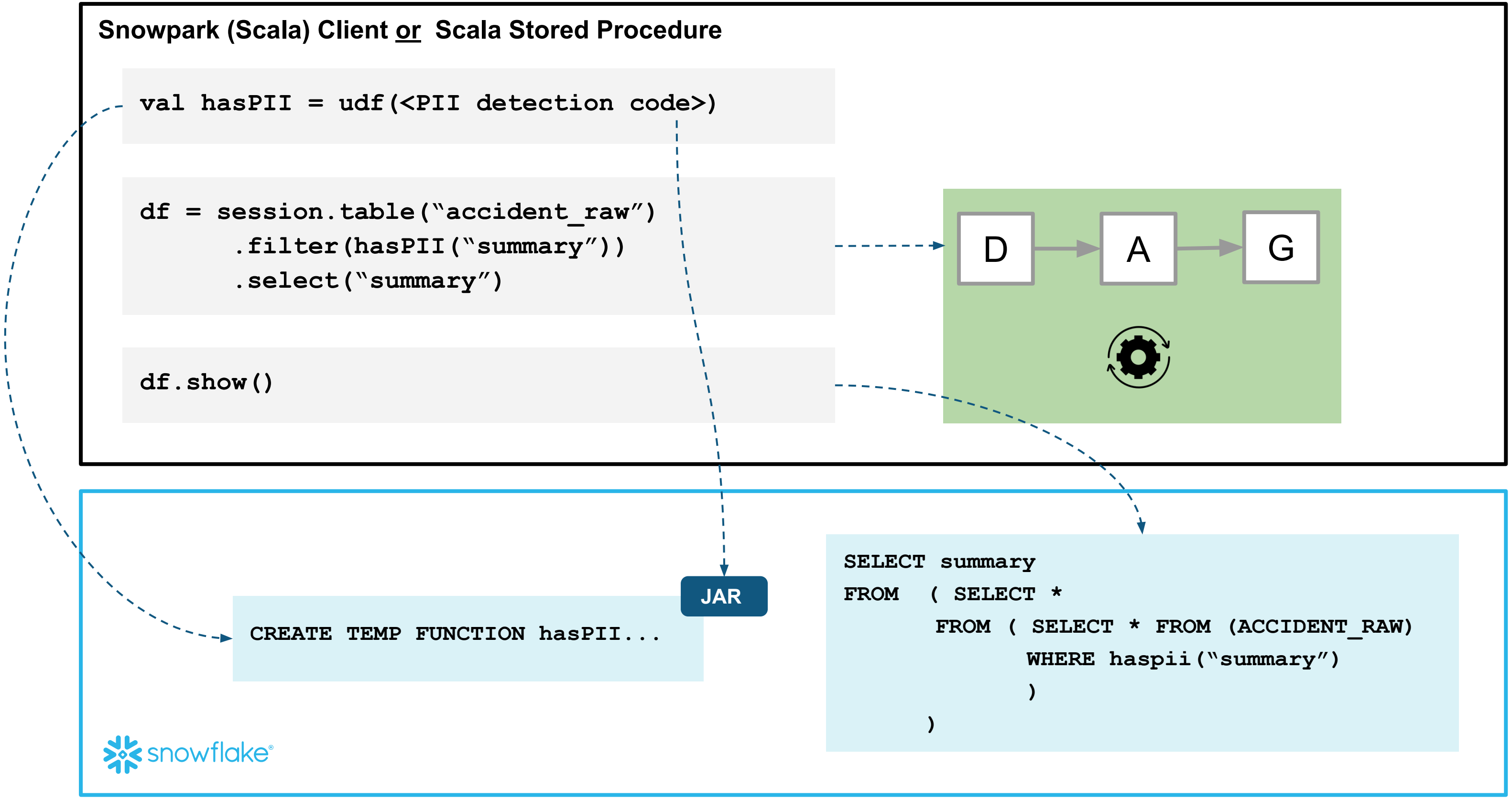
```
public class MyClass {
    public static double
myCustomFunctions (String s)
    {
        /*
         * Let it snow!
         */

        return rval;
    }
}
```

2. Deploy .jar to Snowflake stage

3. Bind and use in Snowflake

SNOWPARK + UDFs



Snowpark (Scala) Client <u>or</u> Scala Stored Procedure

```
val hasPII = udf(<PII detection code>)
```

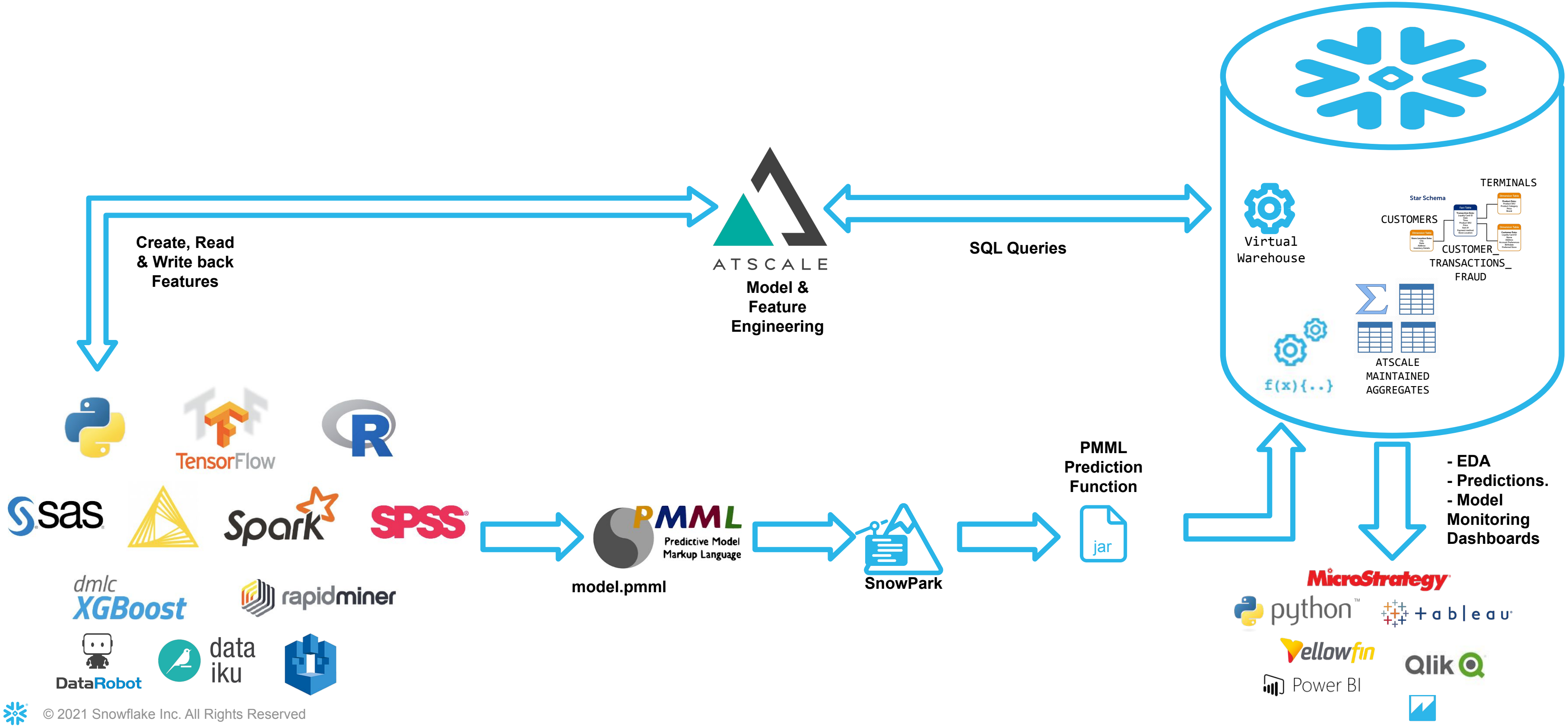
```
df = session.table(“accident_raw”)
       .filter(hasPII(“summary”))
       .select(“summary”)
```

```
df.show()
```

```
CREATE TEMP FUNCTION hasPII...
```

```
SELECT summary
FROM  ( SELECT *
        FROM ( SELECT * FROM (ACCIDENT_RAW)
               WHERE haspii(“summary”)
             )
      )
```

Model-Portability standards enable model inference & MLOps in Snowflake



Create, Read & Write back Features

ATSCALE

Model & Feature Engineering

SQL Queries

Virtual Warehouse

Star Schema

TERMINALS

CUSTOMERS

CUSTOMER_ TRANSACTIONS_ FRAUD

f(x){..}

ATSCALE MAINTAINED AGGREGATES

model.pmml

SnowPark

PMML Prediction Function

- EDA
- Predictions.
- Model Monitoring Dashboards

DATA SCIENCE WITH SNOWFLAKE
BEST PRACTICES

Enrich datasets using **Data Marketplace** for improved model accuracy

Use **Streams & Tasks** to build end-to-end ML pipelines

Create datasets without loading data into Snowflake via **External Tables**

Leverage **External & JAVA Functions** for training or to get predictions

Use **Zero-Copy Clones** for training snapshots

Use regular or Materialized **Views** to create repository of ML features used for training and prediction

Optimize training instance memory usage by using **Snowflake SQL** for aggregation & sampling

Use **SnowPark** for functional programming with **dataframes** running in Snowflake

SUMMARY

❏ AtScale enables data, features and relationships to be modelled over Snowflake tables.

❏ Native Data Frame support via Snowpark enables Data Engineers and Scientists to build data engineering pipelines and execute models.

❏ Model storage/persistence and interoperability via PMML (and other) open model format.

❏ Java UDF allows fast compiled custom code execution within Snowflake.

❏ Access to Java based languages and libraries directly in Snowflake.