
ADVANCING ANALYTICS

# ADVANCING DATABRICKS

## NEXT LEVEL ETL

Simon Whiteley
@MrSiWhiteley

DATA:Scotland

Microsoft

DATA MOVEMENTS
BRINGING DATA INTO FOCUS

adatis

cathcart associates
TECHNICAL RECRUITMENT SPECIALISTS

Arnold Clark

quorum

POWER BI SENTINEL™
Documentation · Data Lineage · Change Tracking · Backup

SentryOne®

F1F9

Incremental Group

dbWatch DATABASE CONTROL

DLM Consultants

redgate

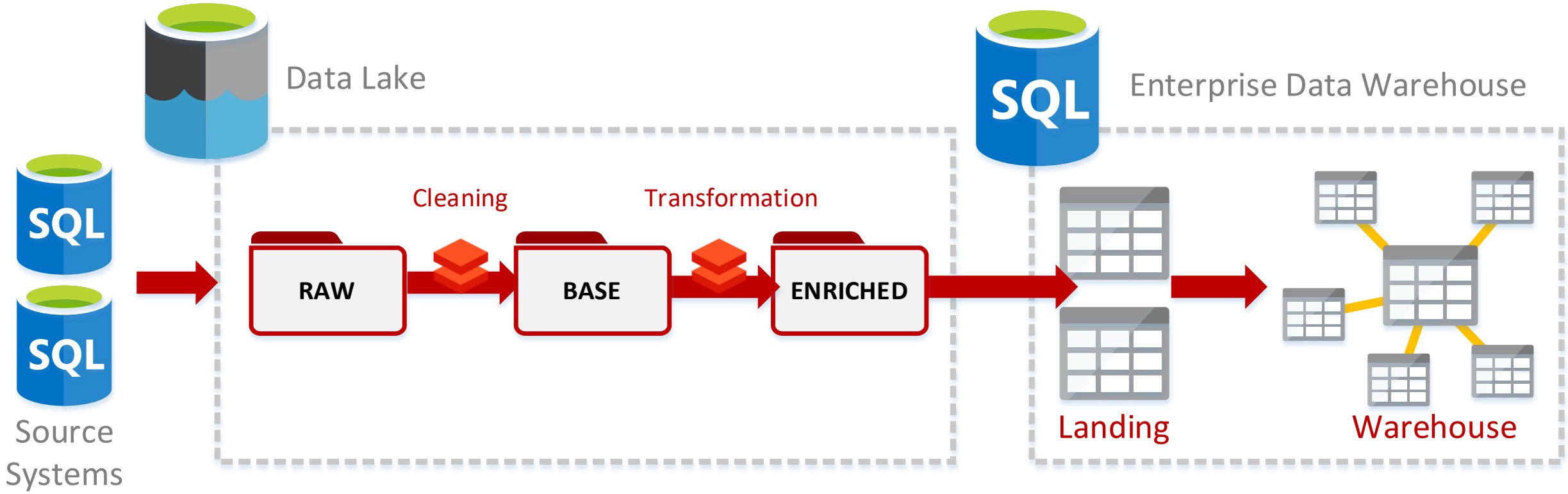
https://github.com/SiWhiteley/AdvancingDatabricks

ADVANCING DATABRICKS

# Agenda

- Schemas & Inference
- Validating Data
- Orchestration
- Spark Internals / Deep Questions

ADVANCING DATABRICKS




ADVANCING DATABRICKS

# THE DATA FRAME

## DataFrame

- Schema ← Parameter
- Format ← Parameter
- Location ← Parameter

```
df = (spark
        .read
        .schema(newSchema)
        .format(fileFormat)
        .load(dataLocation)
      )
```

# APPLYING STRUCTURE

# SCHEMA ON READ – INFER SCHEMA

Cmd 3

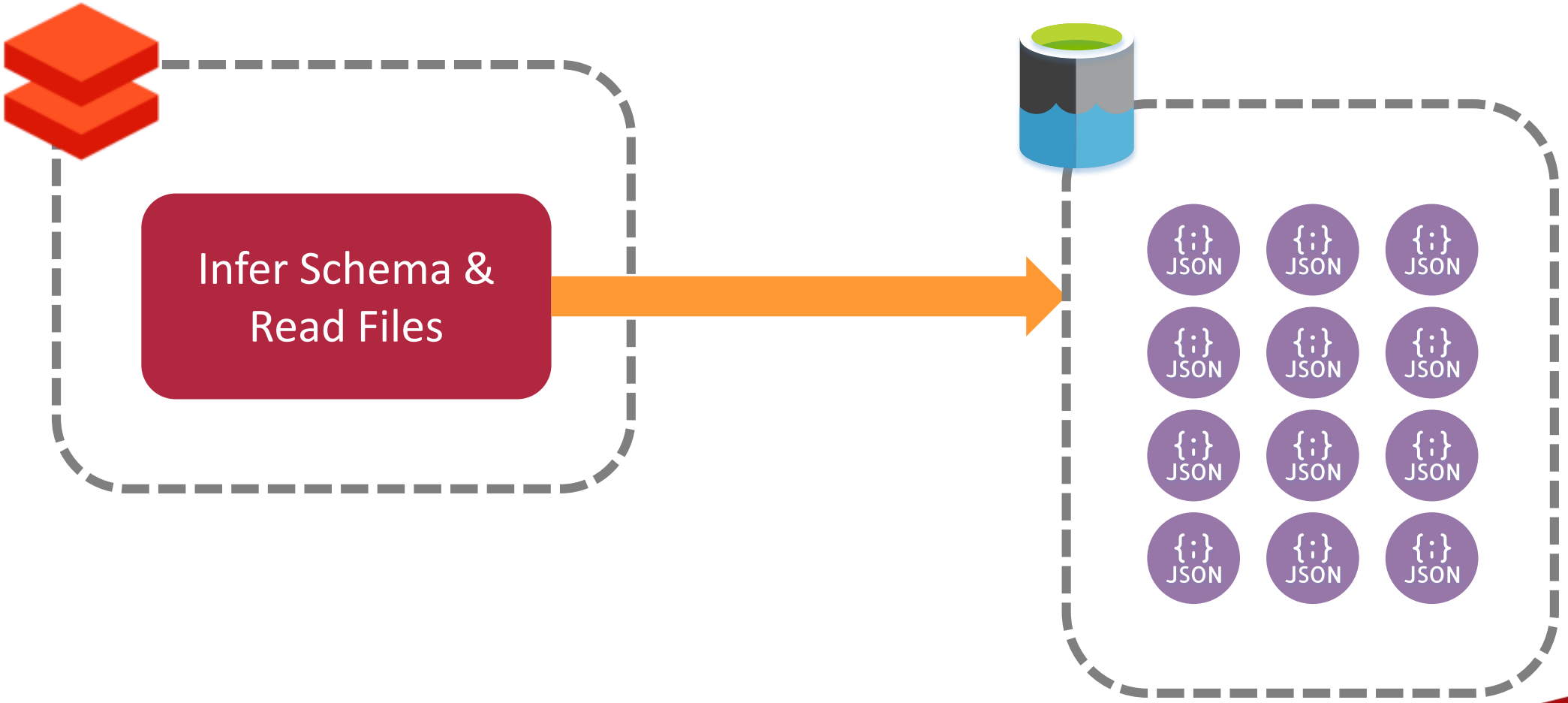
```
df = sqlContext.read.format("csv") \
  .option("header", "true") \
  .option("inferSchema", "true") \
  .load("abfss://root@dblake.dfs.core.windows.net/RAW/Public/Taxi/v1/SmallSlice.csv")
```

df: pyspark.sql.dataframe.DataFrame

```
Dispatching_base_num: string
Pickup_DateTime: timestamp
DropOff_datetime: string
PUlocationID: integer
DOlocationID: string
```

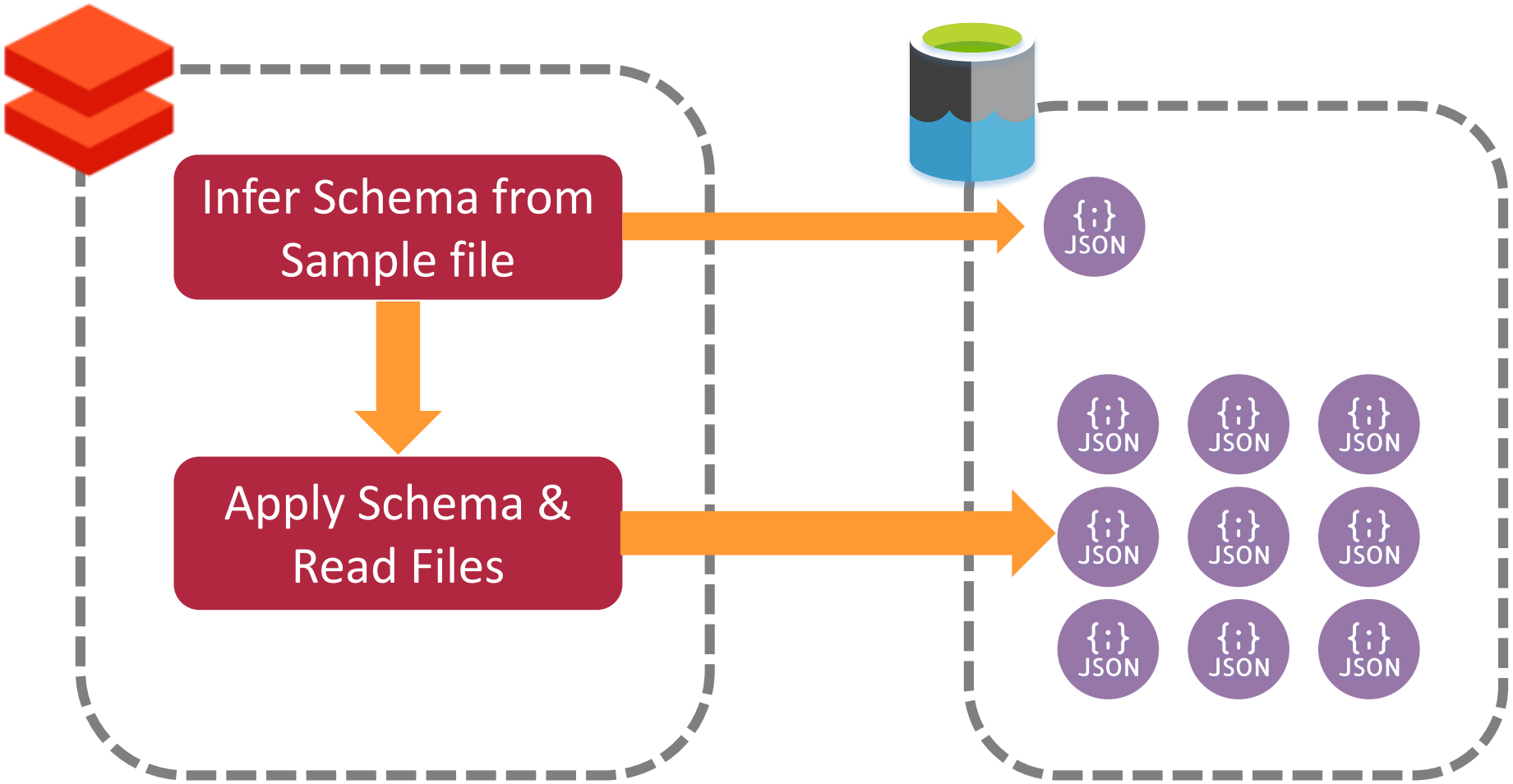
ADVANCING DATABRICKS

# SCHEMA ON READ – INFER SCHEMA



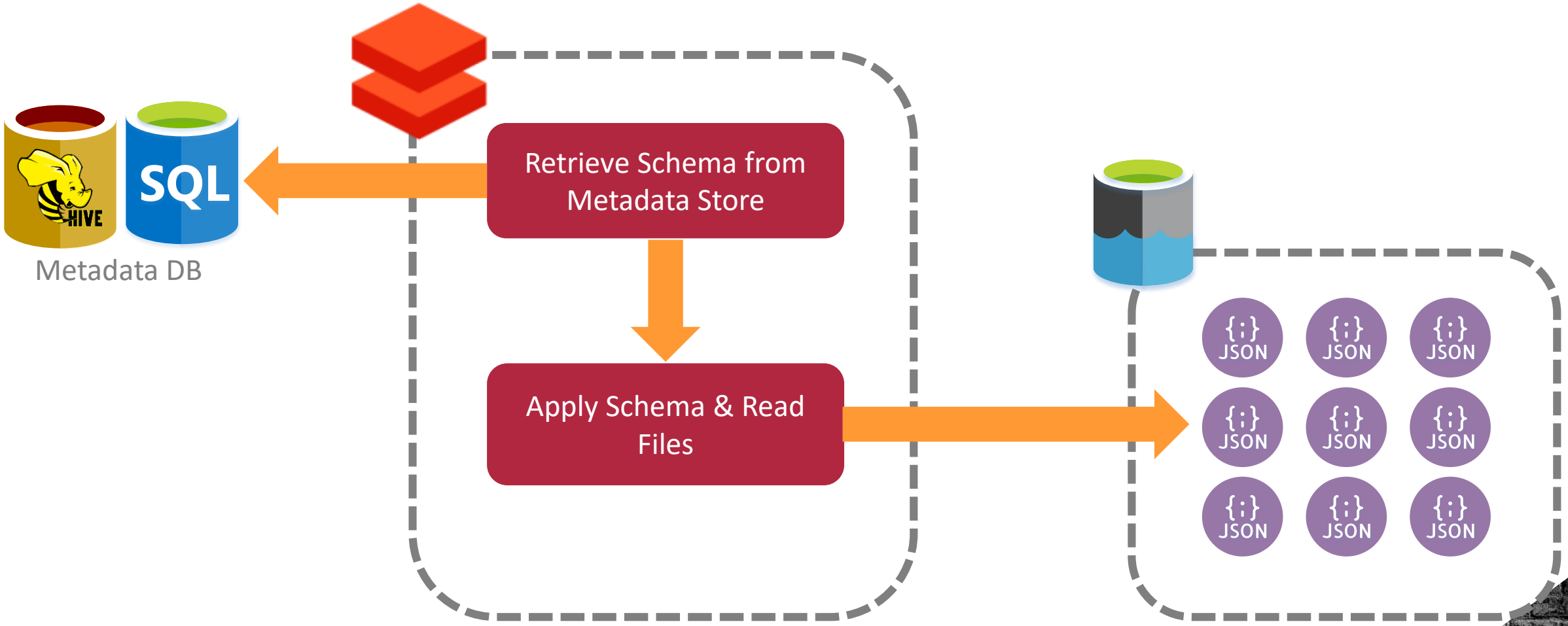

ADVANCING DATABRICKS

# SCHEMA ON READ – SAMPLE FILES




ADVANCING DATABRICKS

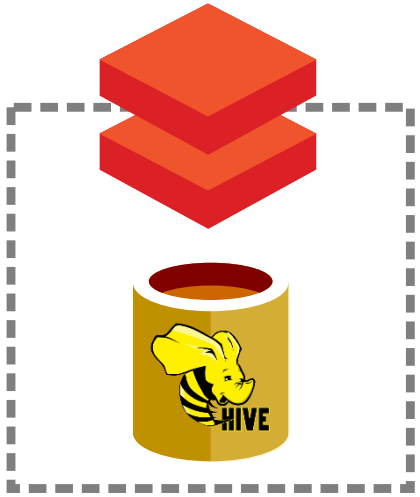
# SCHEMA ON READ – METADATA STORE



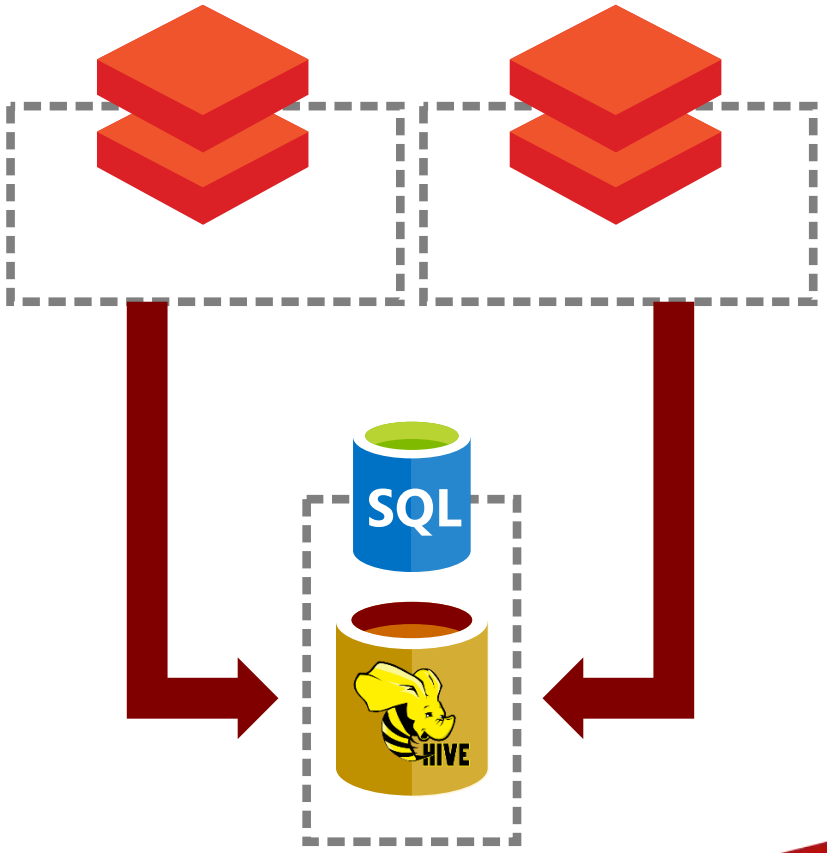

ADVANCING DATABRICKS

# USING THE HIVE METASTORE

Internal
(Cross-Cluster)



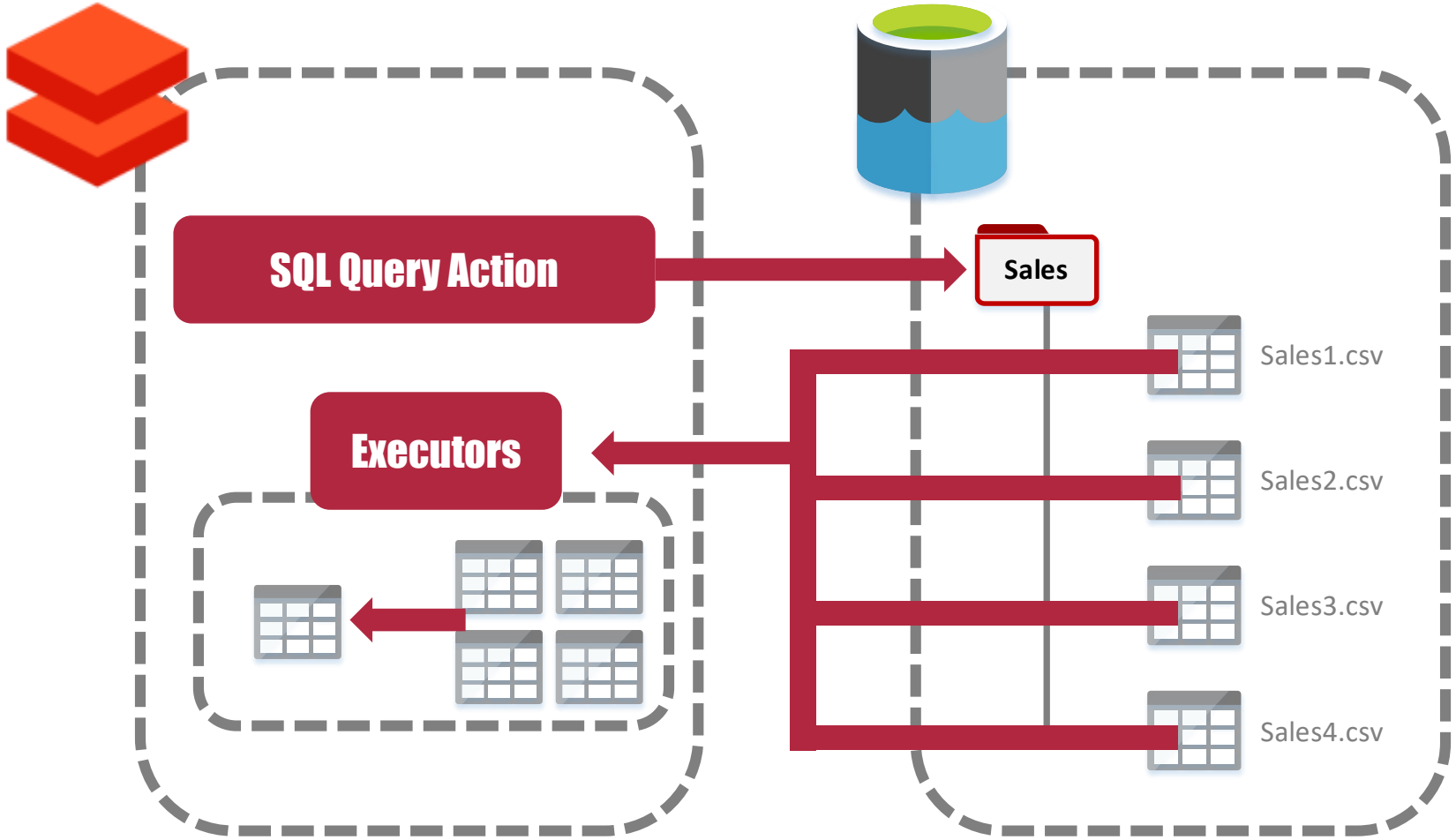

External
(Cross-Workspace)



# PARTITIONING

# READING FILES – NO PARTITIONS

*SELECT * FROM MyFiles WHERE Year = 2019 AND Month = 3*

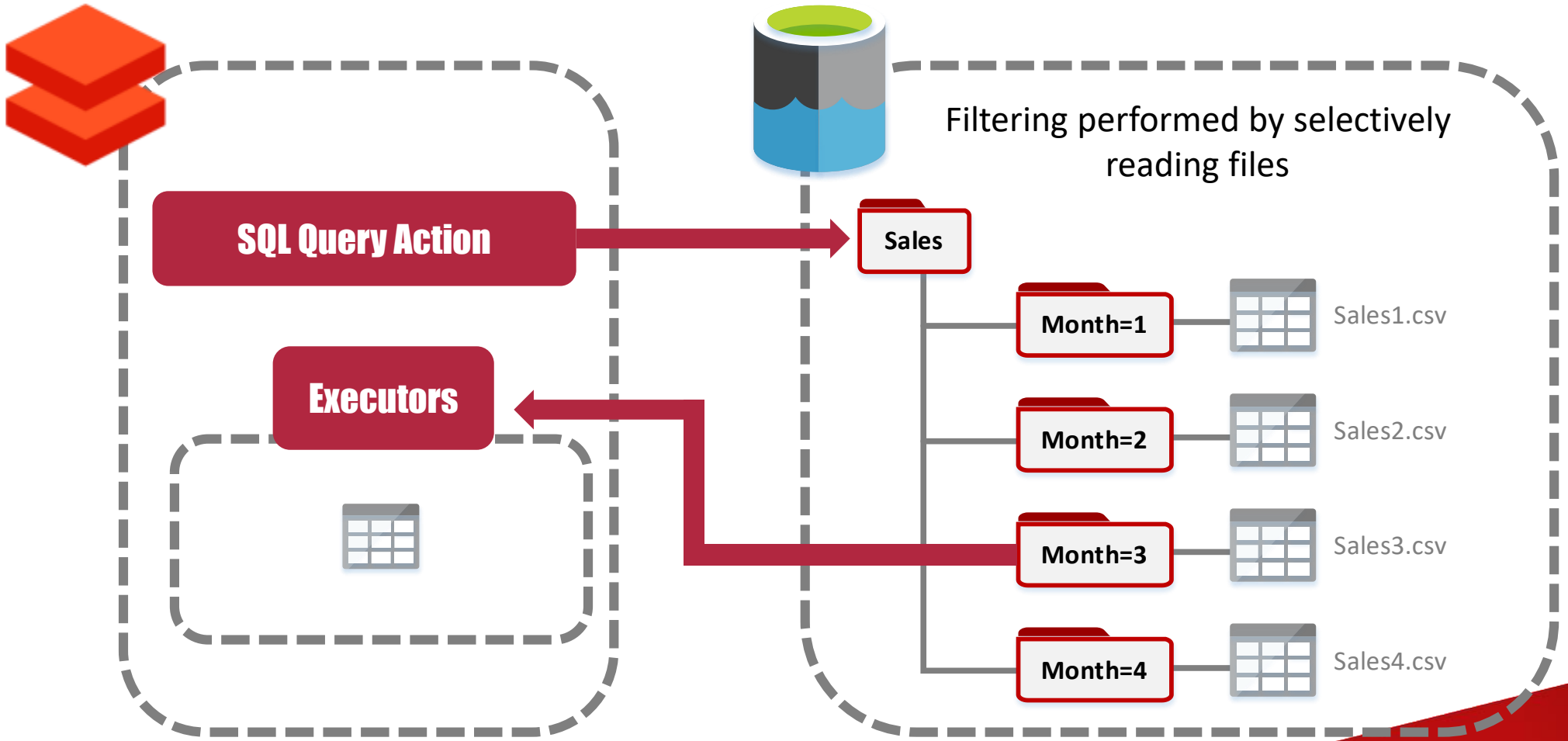



Filtering performed on executors after reading all files

DATA ENGINEERING VS DATA SCIENCE

# READING FILES – PARTITIONED

*SELECT * FROM MyFiles WHERE Year = 2019 AND Month = 3*



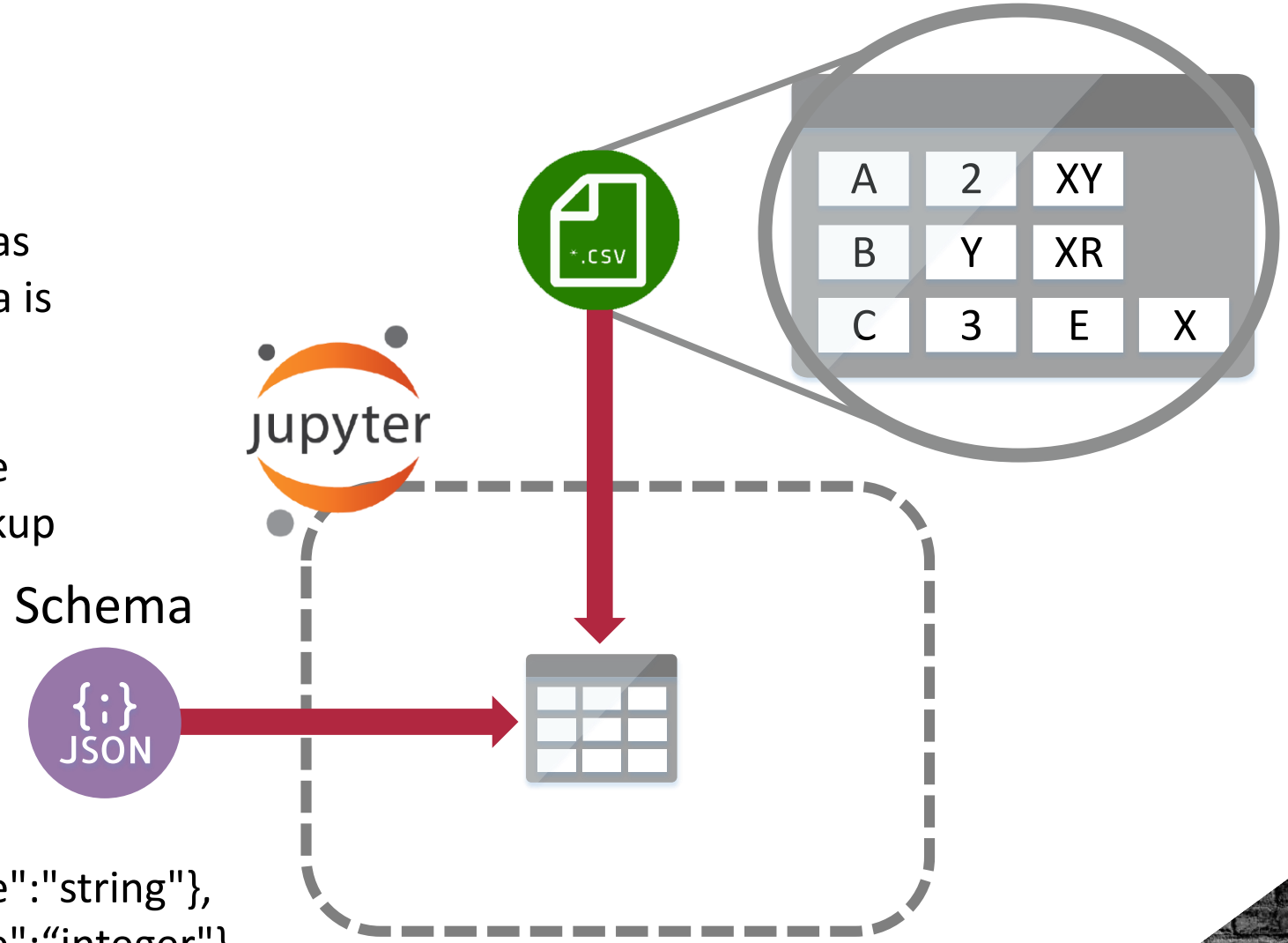

DATA ENGINEERING VS DATA SCIENCE

# DATA VALIDATION

# DATA VALIDATION

With Schema-On-Read file types such as CSV, we need to make sure that the data is in the format that we expect it to be.

Let's assume that we're specifying the schema ourselves, from an external lookup




```
{"fields":[
{"name":"Col1","nullable":true,"type":"string"},
{"name":"Col2","nullable":true,"type":"integer"},
{"name":"Col3","nullable":true,"type":"string"}
]}
```

DATA ENGINEERING VS DATA SCIENCE

# DATA VALIDATION

We have three different methods for handling failed parsing of a DataFrame when accessing text datasets such as csv:

| string | int | string | |
|---|---|---|---|
| A | 2 | XY | |
| B | Y | XR | |
| C | 3 | E | X |

| | | |
|---|---|---|
| A | 2 | XY |
| null | null | null |
| C | 3 | E |

1. **PERMISSIVE** – Extra columns are simply ignored, if any column fails to parse the entire row is nullified

| | | | |
|---|---|---|---|
| A | 2 | XY | |
| B | Y | XR | |
| C | 3 | E | X |

2. **FAILFAST** – If any attributes are different to the specified schema, whether by failing to parse datatypes or attributes added/missing, the entire DataFrame will fail to load

| | | |
|---|---|---|
| A | 2 | XY |

3. **DROPMALFORMED** – Any rows that differ from the schema will be silently dropped from the dataset, also known as the "Nothing to see here" approach to ETL…

DATA ENGINEERING VS DATA SCIENCE

# DEMO: DATA VALIDATION APPROACHES

- PERMISSIVE
- FAILFAST
- DROP MALFORMED

- A BETTER WAY

# ORCHESTRATION

# DATABRICKS WIDGETS

Cmd 3

```
#dbutils.widgets.removeAll()
dbutils.widgets.text("fileName", "Product","AdventureWorks Table")
dbutils.widgets.dropdown("entity_name", "Taxi",["Taxi","TaxiZones"] ,"Entity Name")
```

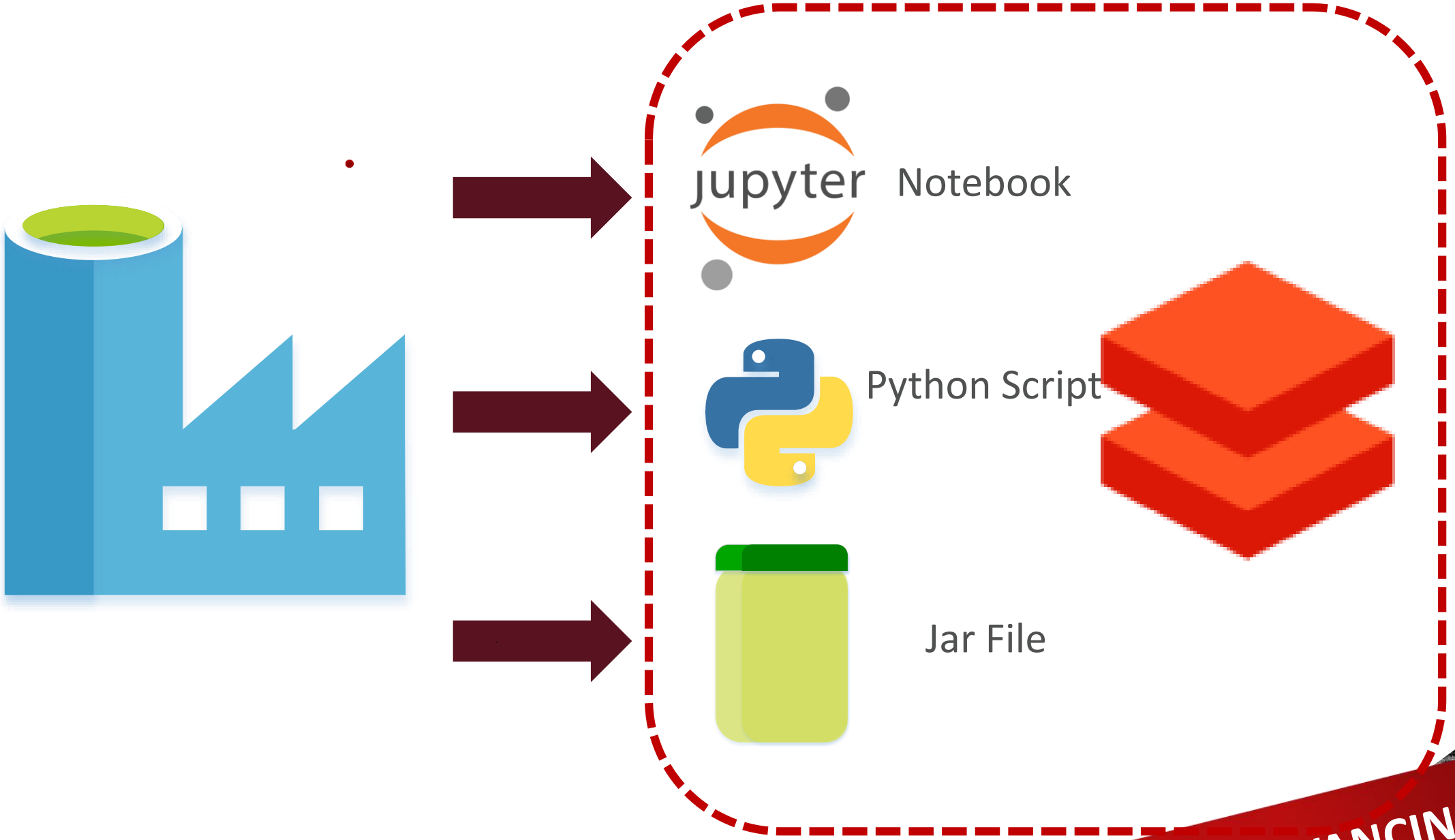
AdventureWorks Table : ProductCategory

Entity Name : Taxi

Cmd 8

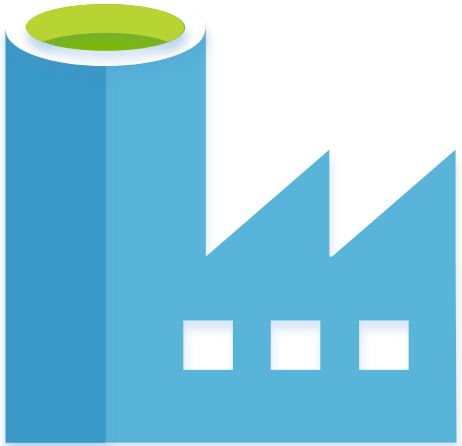
```
fileName = dbutils.widgets.get("fileName")
```

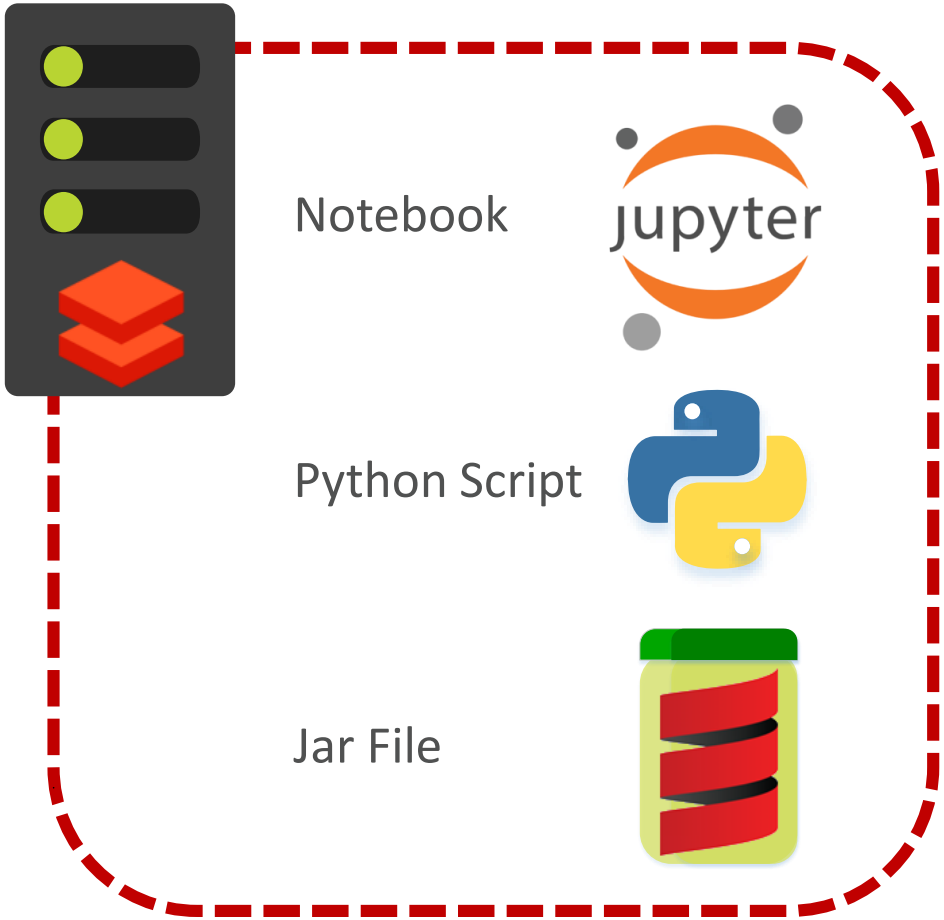
ADVANCING DATABRICKS




ADVANCING DATABRICKS

# AZURE DATA FACTORY






ADF uses Databricks Jobs behind the scenes, but takes away some of the work, and means you can orchestrate your notebooks with other tasks!

# AZURE DATA FACTORY

Notebook path * /Demonstrations/Dynamic Transformation/D Browse

Base Parameters

+ New | Delete

| NAME | VALUE |
| --- | --- |
| entity_name | Taxi |

Entity Name : Taxi

Cmd 1

## Configure Widgets

```
#dbutils.widgets.removeAll()
dbutils.widgets.dropdown("entity_name", "Taxi",["Taxi","TaxiZones"] ,"Entity Name")
```

Cmd 10

## Return Results to Caller

```
import json
dbutils.notebook.exit(json.dumps({"processedRows":processedRows, "status":"Succeeded",
```

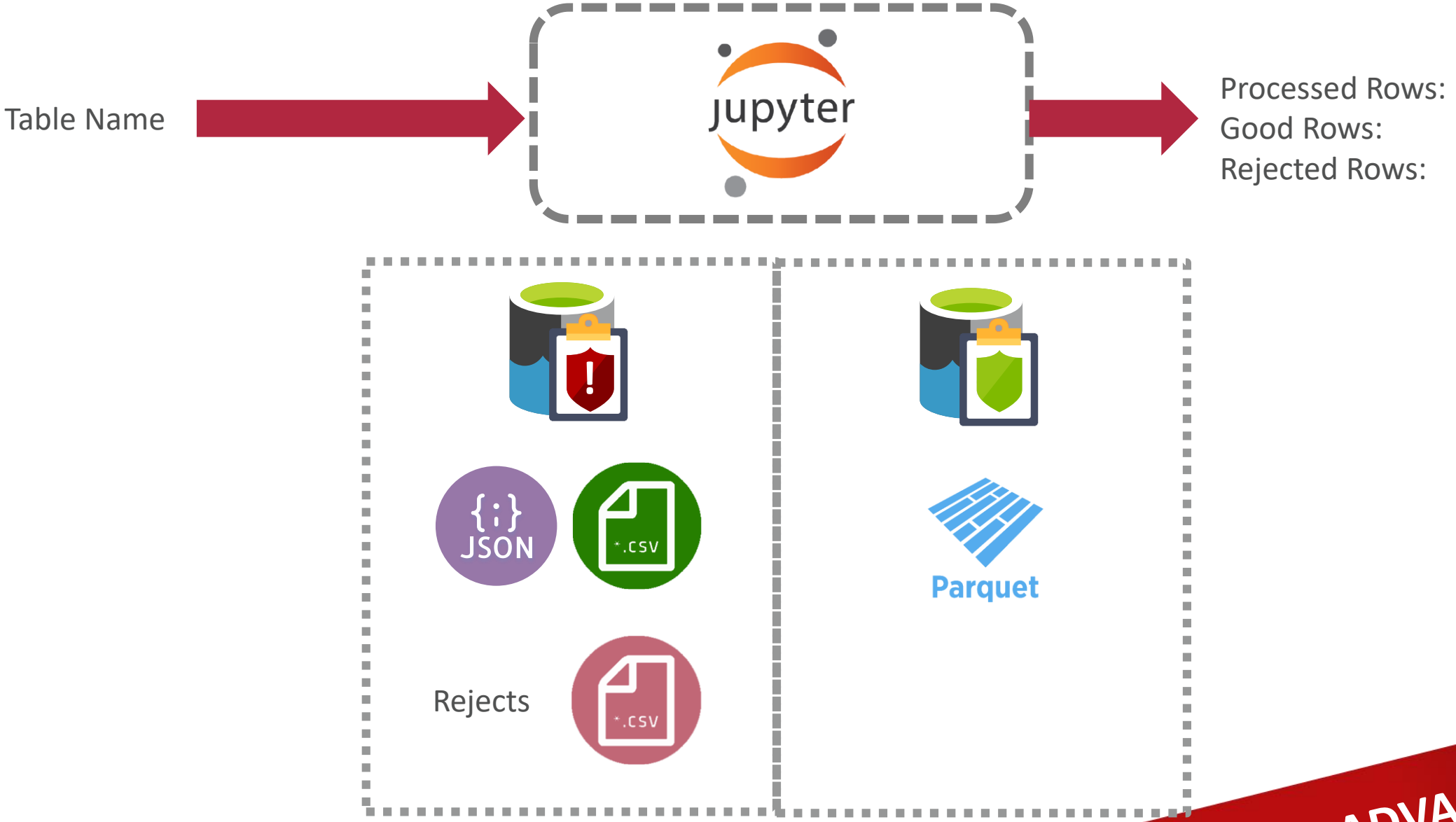
```
Notebook exited: {"Status": "Succeeded", "ProcessedRows": 66141344, "TransformationsApplie
```


```
"runOutput": {
    "TransformationsApplied": 2,
    "Status": "Succeeded",
    "ProcessedRows": 66141344
},
```


ADVA
DATABRICKS

# ONTO THE DEMO…

# OUR GOAL:




ADVANCING DATABRICKS

# DEMO: DYNAMIC VALIDATION

- USING WIDGETS
- PARSING SCHEMAS
- DYNAMIC DATAFRAMES
- WRITING DATA
- OUTPUT PARAMETERS

# WHY WOULD WE DO THIS?

TIME (COST)

| Old Days | Newer Days | Now |
| --- | --- | --- |
| Design | Design | Design Metadata |
| Build SSIS | Build SSIS | Support One Thing |
| Build Orchestration | Build Orchestration | |
| Test Everything | DevOps | |
| Deploy Everything | Support Everything | |
| Support Everything | | |

Session Feedback
bit.ly/2019sfeedback

Event Feedback
bit.ly/2019efeedback

DATA:Scotland

# DATA:Scotland Speaker Scholarship


BIT.LY/DS-SPEAKER

ADVANCING ANALYTICS

# QUESTIONS?

Simon Whiteley
@MrSiWhiteley