AUTOMATED ETL WITH DLT

Microsoft® MVP Most Valuable Professional

Databricks BEACONS

ETL FRAMEWORKS

A HISTORY OF ETL AUTOMATION

Table 1

Table 2

Table 3

Table 4

Table 5

Table 6

Table 7

Table 8

Table 9

Table 10

ADVANCING ANALYTICS

CODE GENERATION

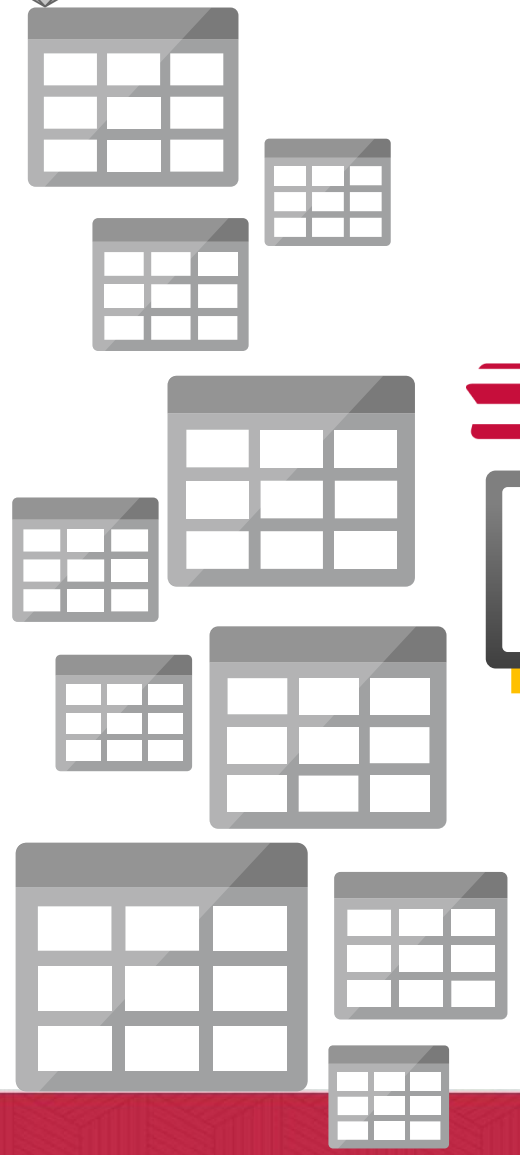


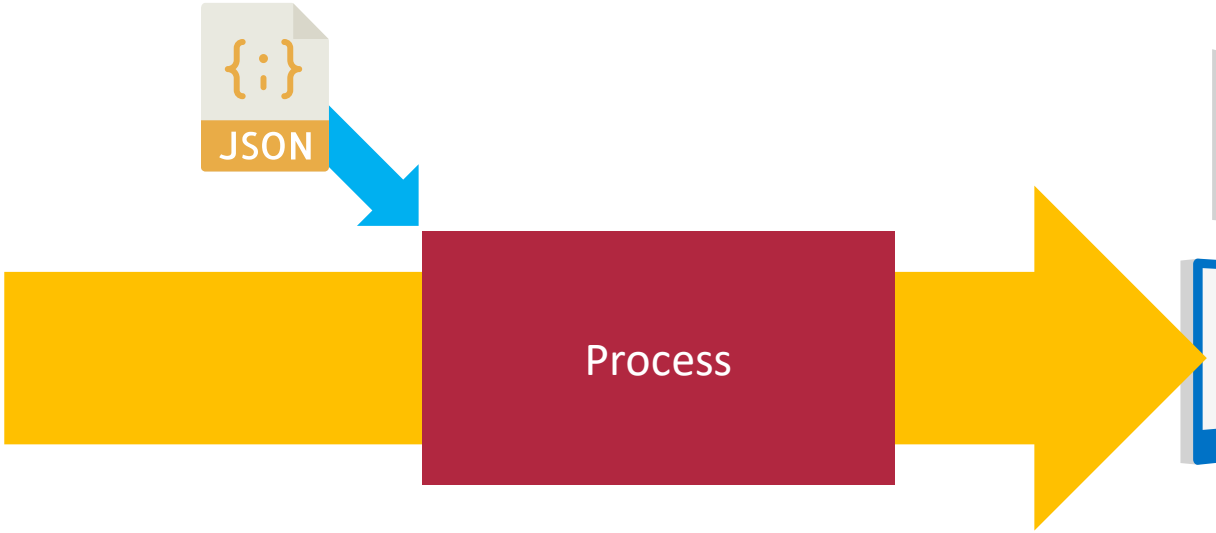
Table 1
Table 2
Table 3
Table 4
Table 5
Table 6
Table 7
Table 8
Table 9
Table 10

Considerations:

- Code Footprint
- Automation Language
- Licence Costs
- Original ETL Problems

ADVANCING ANALYTICS

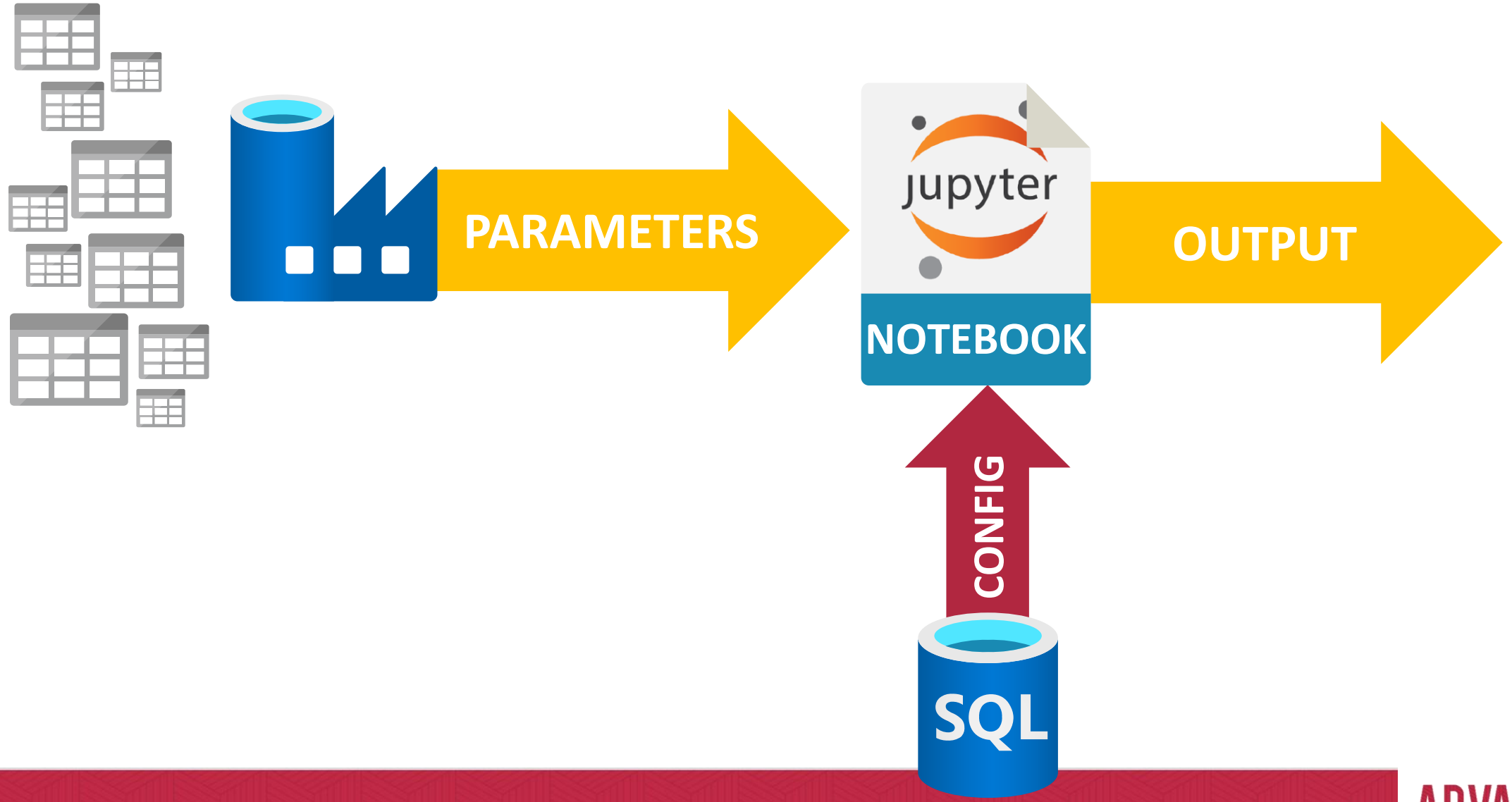
CODE AUTOMATION



Considerations:

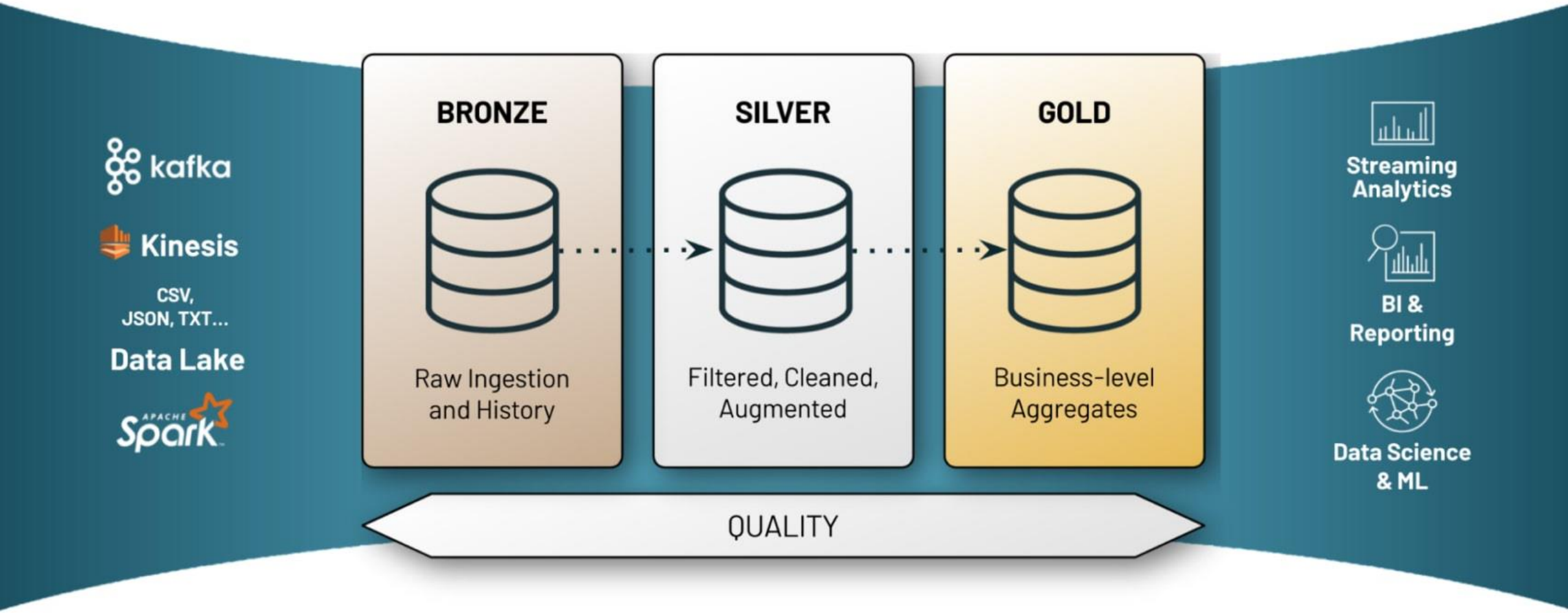
- Code Complexity
- Shift in mindset
- Metadata Management

IN COMMON LAKEHOUSE SCENARIOS

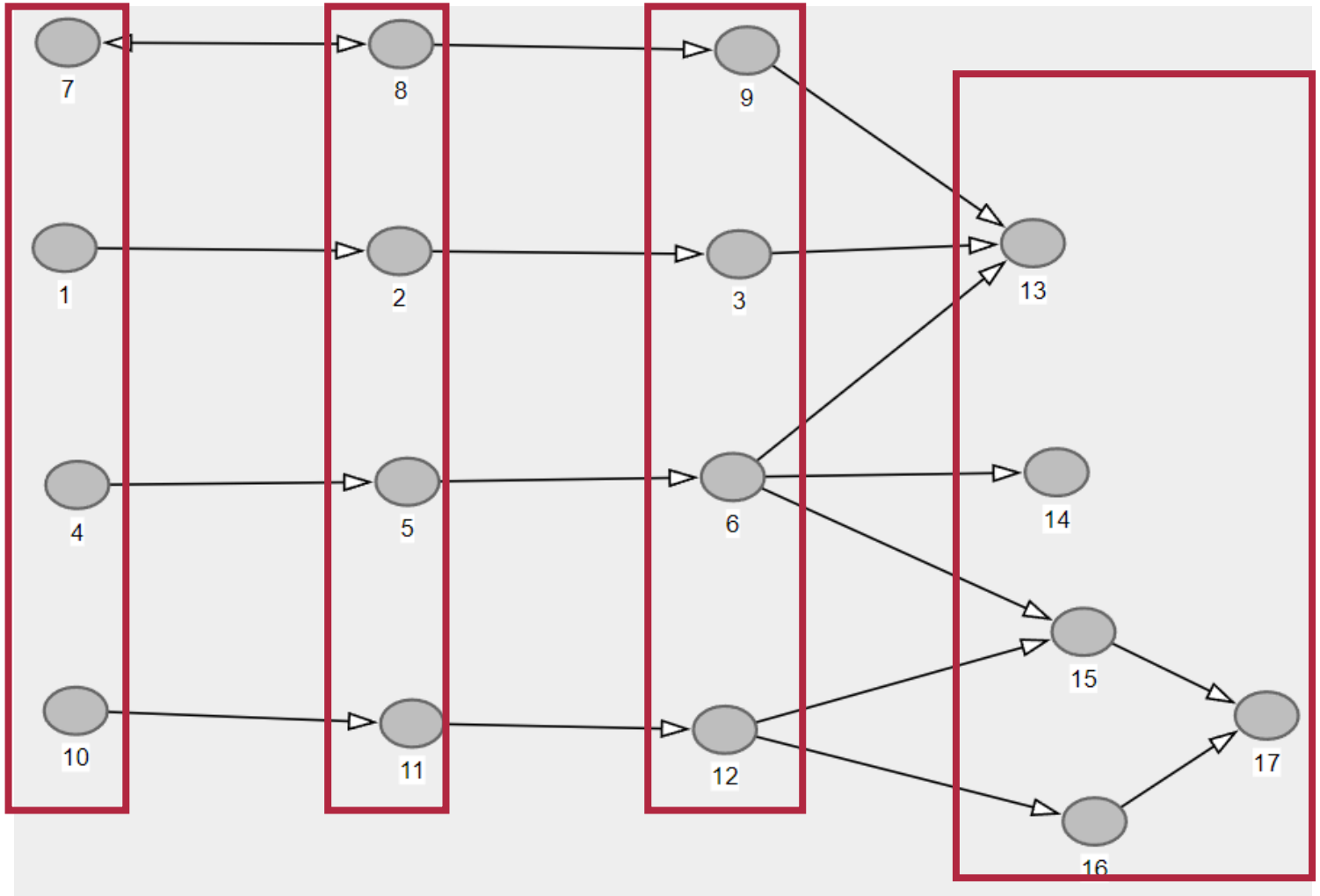


THE CHALLENGES

MEDALLION ARCHITECTURE

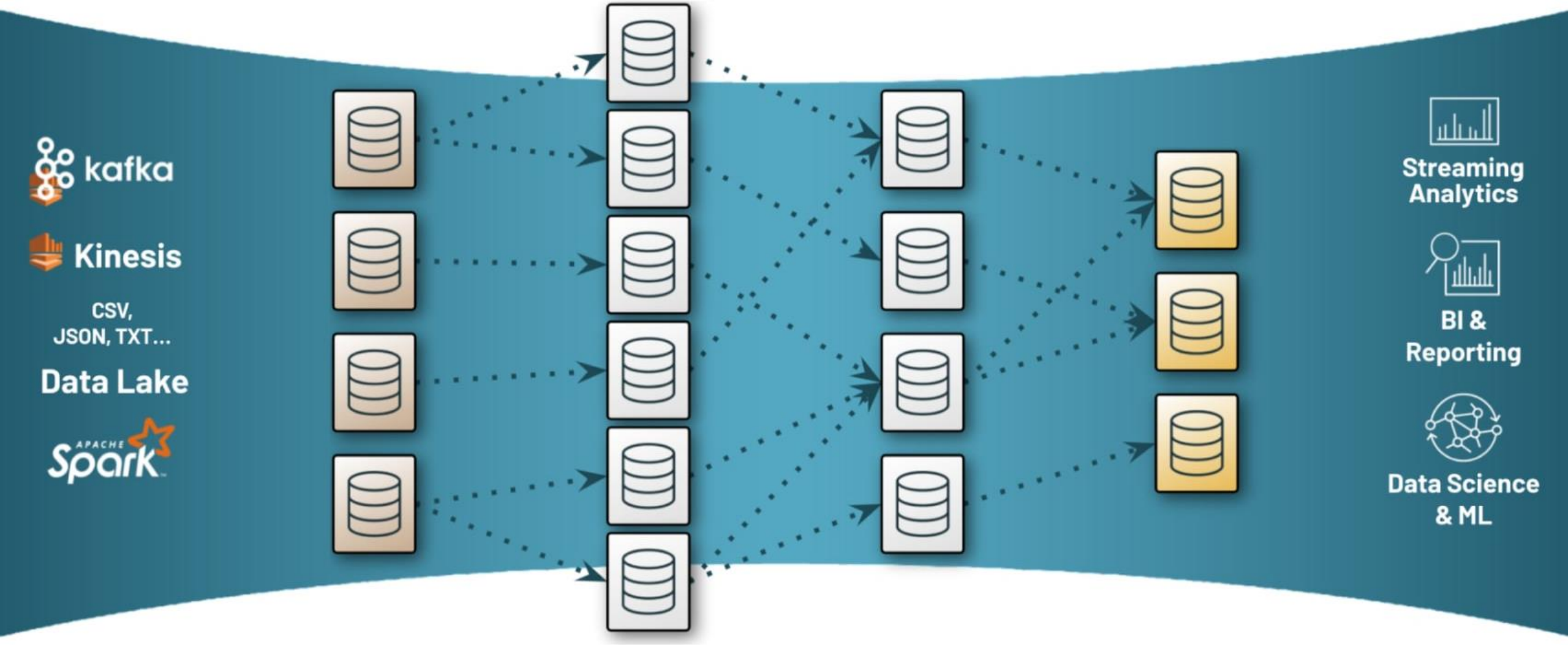


DEPENDENCY MANAGEMENT

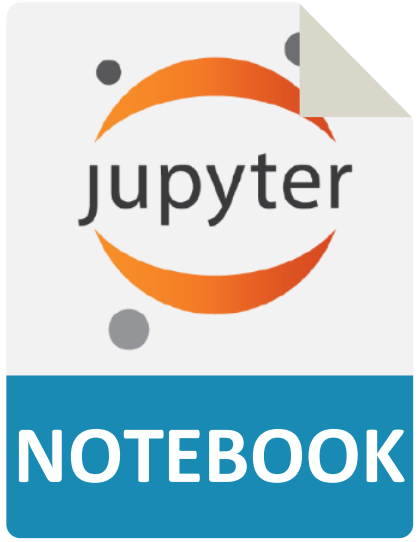


An orchestration process that can understand dependencies, react to data-driven dependencies and handle complex scenarios such as blocking and non-blocking dependencies is… not easy

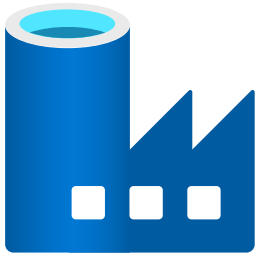
OR USING THE FANCY DATABRICKS DIAGRAM



THE BORING BITS



Logging & Telemetry Management



Process Management & Restartability

Code complexity & library management

SO HOW DO YOU IMPLEMENT AN ETL FRAMEWORK?

Build One

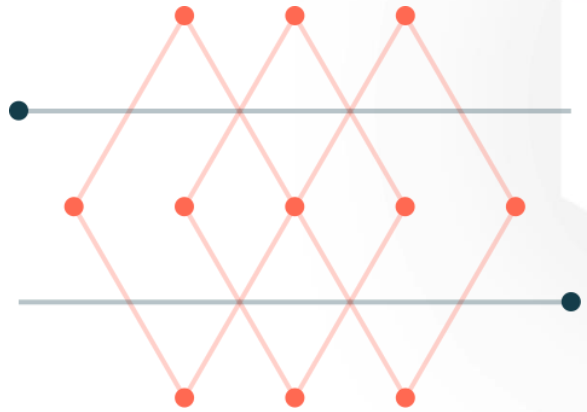
Buy One

Use Free One

Databricks DLT

Use one that's baked into a product you're paying for

DELTA LIVE TABLES

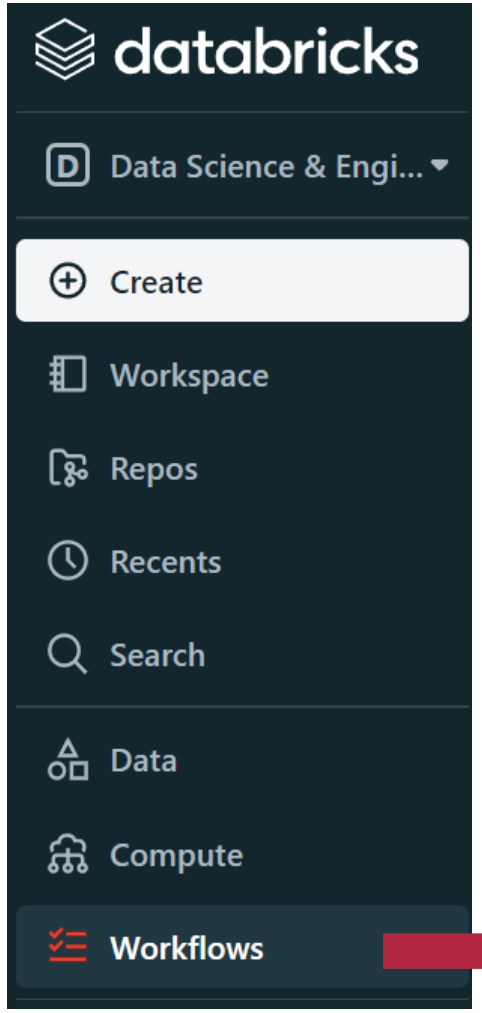


WHAT ARE DELTA LIVE TABLES?

DLT is a framework built into **Databricks workflows** that allow you to define target schemas, dependencies and transformations in Python or SQL.

DLT manages the dependencies and management of the pipeline for you and also contains many useful abstractions over common ETL tasks such as data quality, incremental loading, streaming management and event slowly changing dimensions.

WORKFLOWS > DELTA LIVE TABLES



Workflows

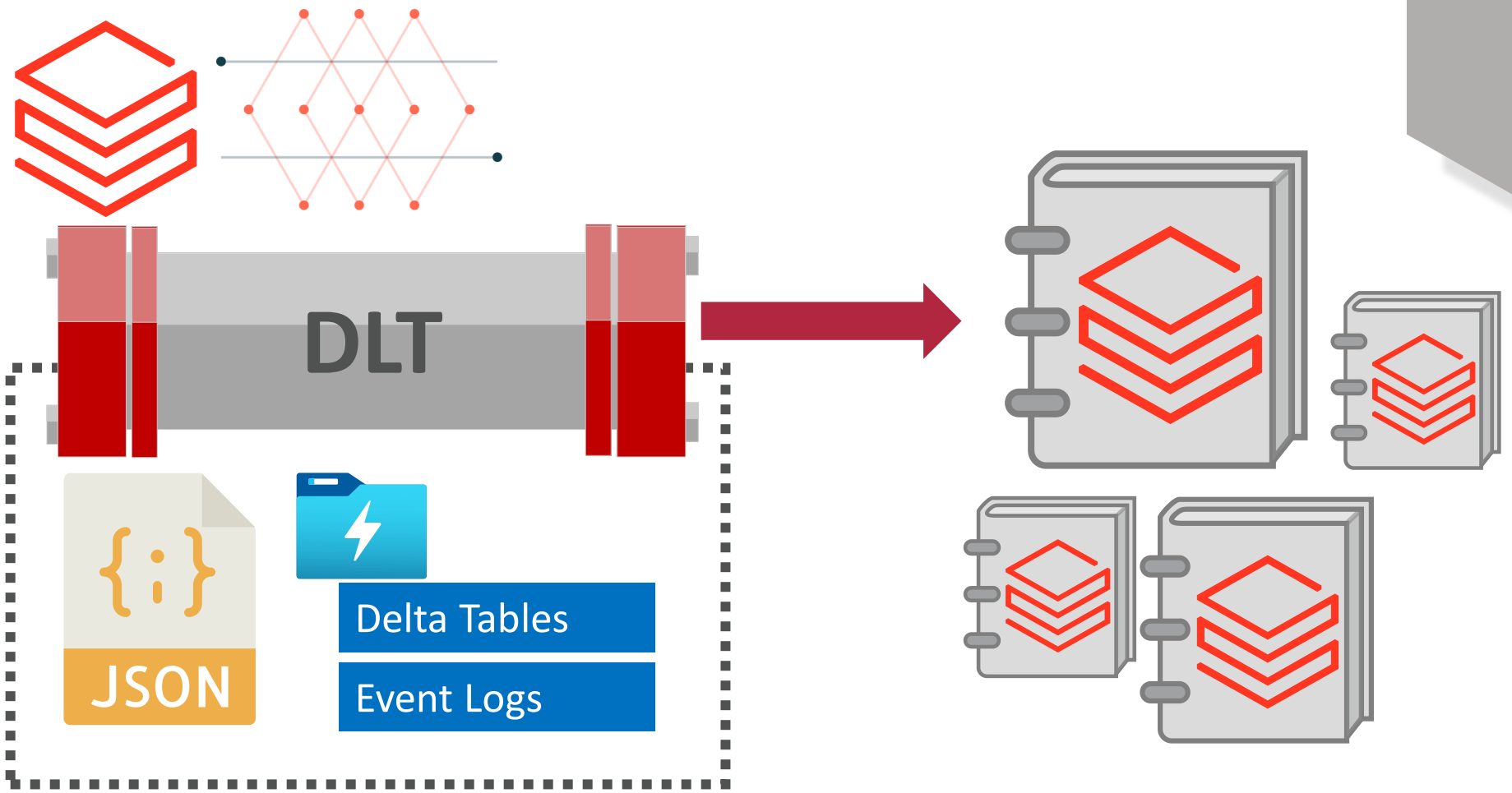
Jobs | Job runs | Delta Live Tables

Create Pipeline | All | Owned by me

Name	Recent updates	ID	Owner	Actions
AdventureworksDW	✓	aa6c52d2-e9eb-4d6b-b656-f8bb81474f8c	simon@advancinganalytics.co.uk	
DLT Example		149bbe8b-01da-46c1-8a88-9de1ec61638f	simon@advancinganalytics.co.uk	

ADVANCING ANALYTICS

DLT COMPONENTS



Pipeline Data

Process Notebooks

THE DLT NOTEBOOK - SQL

We can define our DLT workflows in either SQL or Python. We'll look at SQL first – here's the script to read from a json file into a new table called "clickstream_raw"

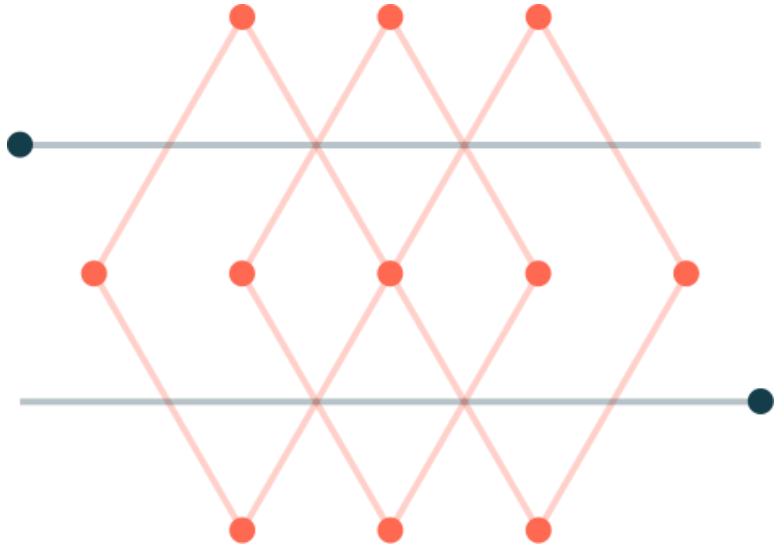
```
--SQL Syntax is deliberately simple, just create a "Live" SQL table from a source
CREATE LIVE TABLE clickstream_raw
COMMENT "The raw wikipedia clickstream dataset, ingested from /databricks-datasets."
AS SELECT * FROM
json.`/databricks-datasets/wikipedia-datasets/data-001/clickstream/raw-uncompressed-json/2015_2_clickstream.json`;
```

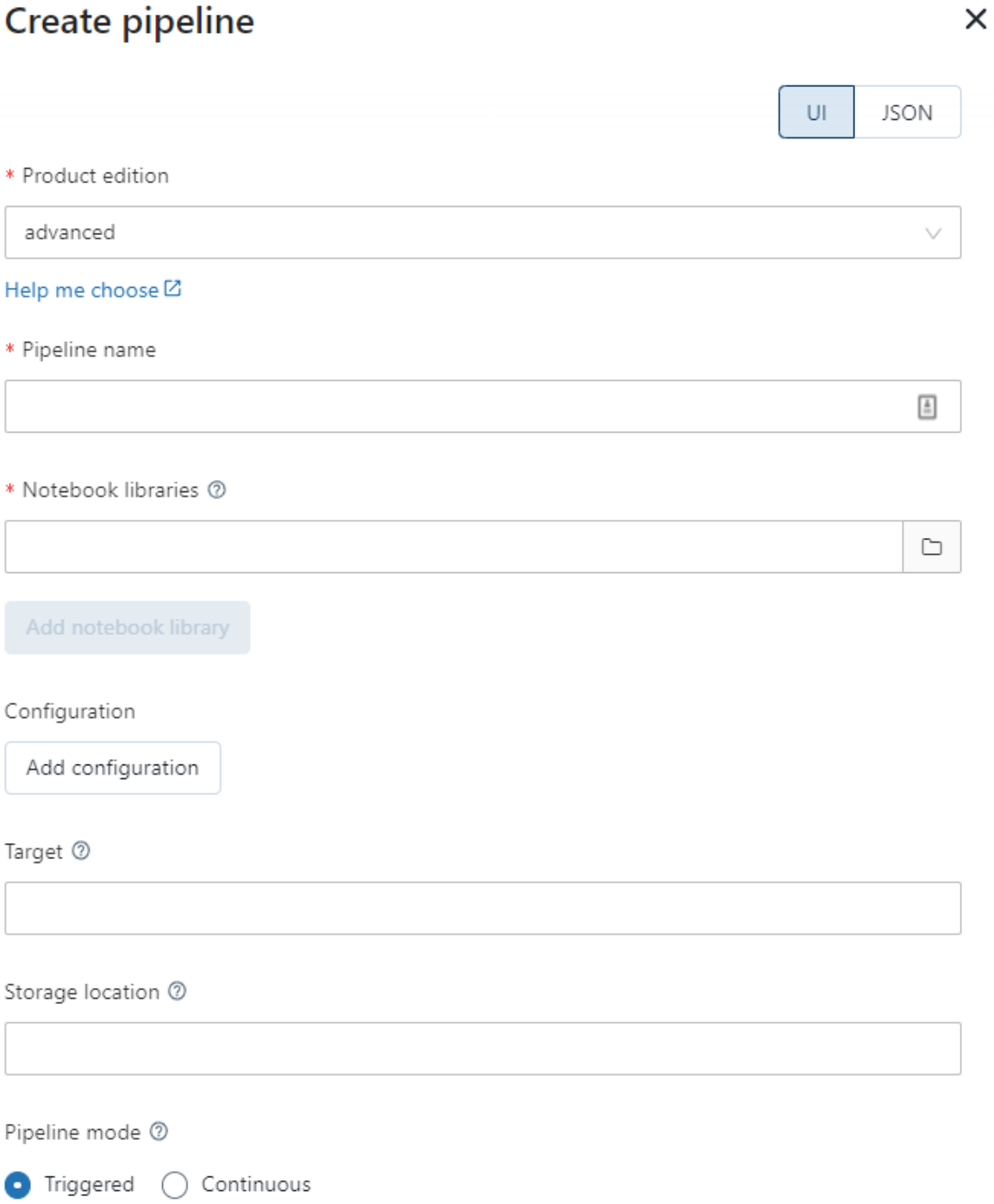
We can create dependant tables just be referring to that live table by name

```
--SQL Syntax is deliberately simple, just create a "Live" SQL table from a source
CREATE LIVE TABLE clickstream_next
AS SELECT * FROM live.clickstream_raw
```

PIPELINE CREATION

Pipelines can be defined using either a simple GUI, or by inputting the JSON manually. Some attributes can only be configured through the JSON config file.





ADVANCING ANALYTICS

DLT VERSION

	DLT Core	DLT Pro	DLT Advanced
SQL API	●	●	●
Python API	●	●	●
Streaming Tables	●	●	●
Continuous Pipelines	●	●	●
Auto-loader integration	●	●	●
Observability Metrics with Event Log	●	●	●
Table Lineage	●	●	●
Access Controls (ACL)	●	●	●
Pipeline multi-cluster partitioning	●	●	●
Enhanced Autoscaling (preview)	●	●	●
Change Data Capture (CDC) - type 1		●	●
Change Data Capture (CDC) - type 2 (preview)		●	●
Data Quality Expectation Rules			●
Data Quality Expectation Policies			●
Data Quality Observability			●

BUILDING A BASIC PIPELINE

- Writing a SQL DLT Notebook
- Creating a DLT Pipeline
- Loading some Data

THE DLT NOTEBOOK - PYSPARK

At first glance, the pyspark script looks more complicated. But we're doing the exact same things, but now with all of the flexibility of a spark dataframe

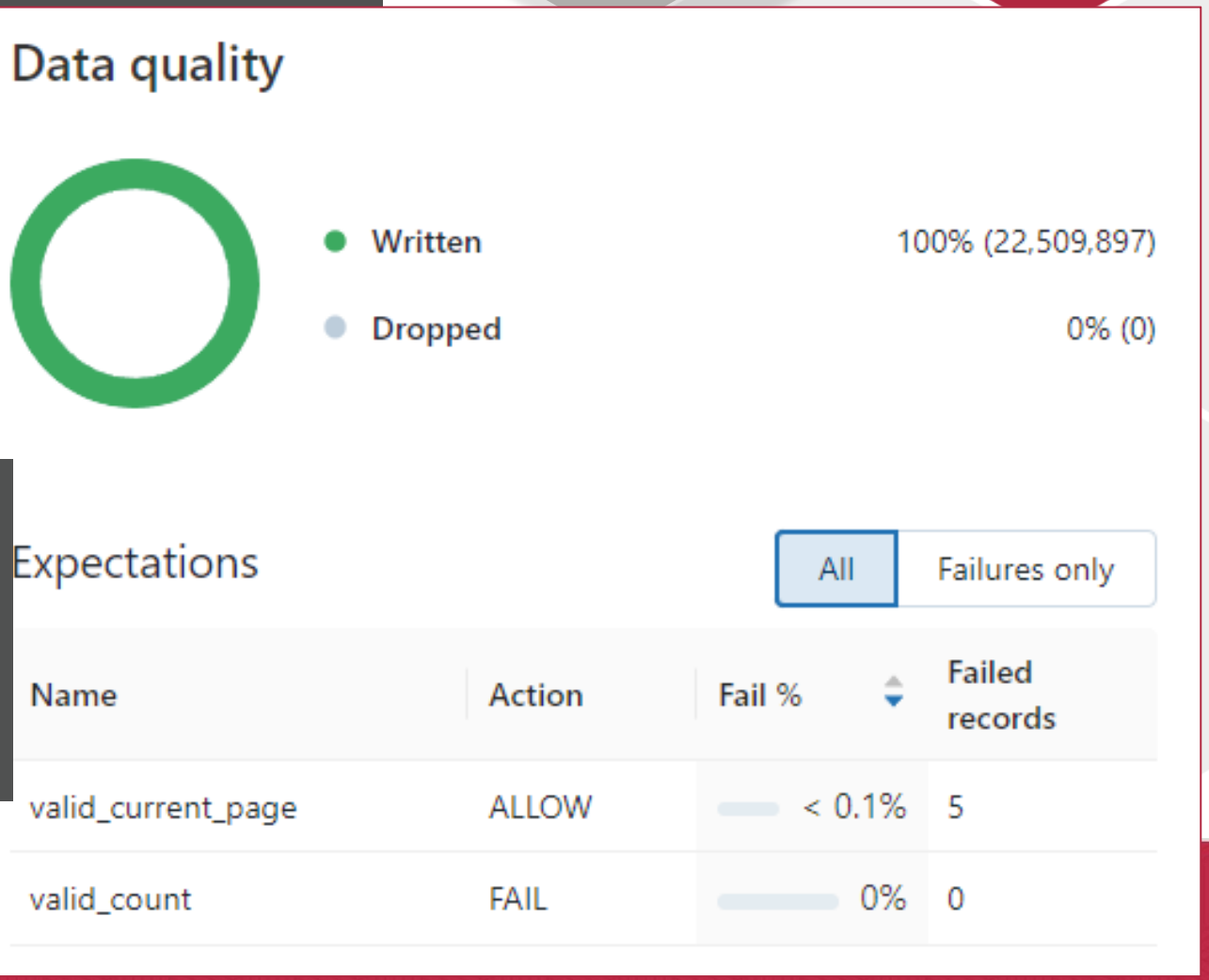
```
#SQL Syntax is deliberately simple, just create a "Live" SQL table from a source
Import dlt

@dlt.table(comment="The raw wikipedia clickstream dataset, ingested from /databricks-datasets.")
def clickstream_raw():
    return (spark.read.json("/databricks-datasets/wikipedia-datasets/data-001/clickstream/raw-uncompressed-
    json/2015_2_clickstream.json"))
```

EXPECTATIONS

GREAT EXPECTATIONS

```
--IN SQL
CREATE LIVE TABLE clickstream_prepared(
  CONSTRAINT valid_current_page EXPECT (current_page_title IS NOT NULL),
  CONSTRAINT valid_count EXPECT (click_count > 0) ON VIOL
```

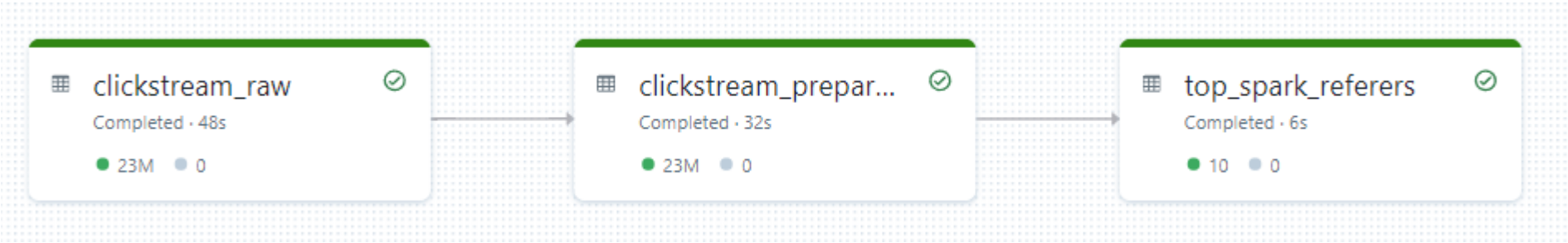
```
#Same but Pyspark
@dlt.table()
@dlt.expect("valid_current_page_title", "current_page_title
IS NOT NULL")
@dlt.expect_or_fail("valid_count", "click_count > 0")
```



ADVANCING ANALYTICS

DLT MANAGEMENT

AUTOMATED OBSERVABILITY



All | Info | Warning | Error | Filter...

18 minutes ago	flow_progress	Flow 'clickstream_prepared' is STARTING.
18 minutes ago	flow_progress	Flow 'clickstream_prepared' is RUNNING.
17 minutes ago	flow_progress	Flow 'clickstream_prepared' has COMPLETED.
17 minutes ago	flow_progress	Flow 'top_spark_referers' is STARTING.
17 minutes ago	flow_progress	Flow 'top_spark_referers' is RUNNING.


```
"timestamp": "2022-08-24T08:12:18.992Z",
"message": "Flow 'clickstream_prepared' has COMPLETED.",
"level": "INFO",
"details": {
    "flow_progress": {
        "status": "COMPLETED",
        "metrics": {
            "num_output_rows": 22509897
        },
        "data_quality": {
            "dropped_records": 0,
            "expectations": [
                {
                    "name": "valid_current_page",
                    "dataset": "clickstream_prepared",
                    "passed_records": 22509892,
                    "failed_records": 5
                }
            ]
        }
    }
```

SCHEDULING

DLT Pipelines can be scheduled via their own schedule workflow

Job name

Schedule: Manual / Scheduled

Every Day at 09 : 30 (UTC+01:00) Dublin, Edin...

Alerts: simon@advancinganalytics.c Start Success Failure Add

Or you can create a Databricks Job manually

Task name *

DLT_Scheduled_Run

Type *: Delta Live Tables pipeline

Pipeline *: DLT with SQL

Advanced options

Cancel Create

ADVANCING ANALYTICS

DEEPER LOOK AT DLT PIPELINES

- Managing & Maintaining
- Logs & Data Stores
- Scheduling & Jobs

ADVANCED ETL

CHANGE DATA CAPTURE (MERGE)


```
APPLY CHANGES INTO LIVE.clickstream_silver
FROM clickstream_raw
KEYS (current_page_title)
SEQUENCE BY import_date
```

NOTE: The DLT table must already exist to be able to merge data into it

```
APPLY CHANGES INTO LIVE.table_name
FROM source
KEYS (keys)
[WHERE condition]
[IGNORE NULL UPDATES]
[APPLY AS DELETE WHEN condition]
[APPLY AS TRUNCATE WHEN condition]
SEQUENCE BY orderByColumn
[COLUMNS {columnList | * EXCEPT (exceptColumnList)}]
[STORED AS {SCD TYPE 1 | SCD TYPE 2}]
```

CHANGE DATA CAPTURE (MERGE) - PYSPARK


```
dlt.apply_changes(
target = "<target-table>",
source = "<data-source>",
keys = ["key1", "key2", "keyN"],
sequence_by = "<sequence-column>",
ignore_null_updates = False,
apply_as_deletes = None,
apply_as_truncates = None,
column_list = None,
except_column_list = None,
stored_as_scd_type = <type> )
```

Ensure you create the DLT table before the merge statement:

```
#Tables can be created using:
dlt.create_steaming_live_table(“tableName")
```

EXAMPLE MERGE PIPELINE FLOW

```python
@dlt.view(name=f"bronze_SalesLT_SalesOrderDetail")
def incremental_bronze():
  df = (spark
      .readStream
      .format("cloudFiles")
      .options(**cloudfile)
      .options(**driftConf)
      .load(dataPath)
       )

  return df

dlt.create_steaming_live_table("silver_SalesLT_SalesOrderDetail")

dlt.apply_changes(
  target = "silver_SalesLT_SalesOrderDetail",
  source = "bronze_SalesLT_SalesOrderDetail",
  keys = ["SalesOrderDetailID"],
  sequence_by = col("ModifiedDate"),
)
```

SLOWLY CHANGING DIMENSIONS

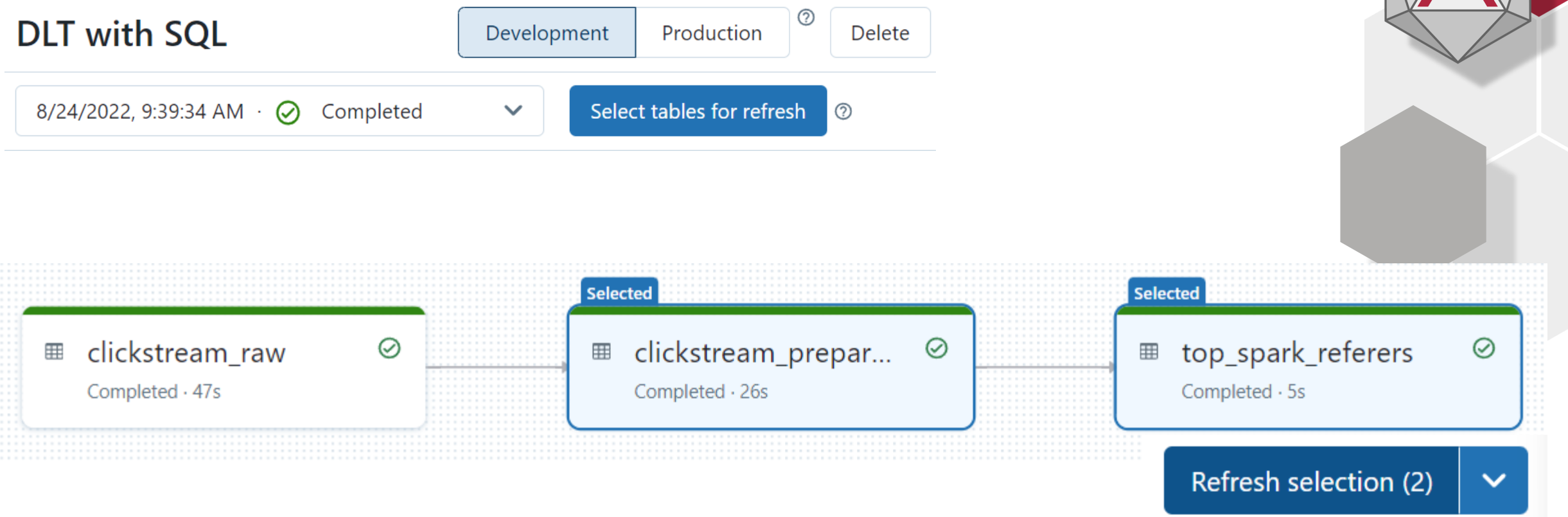
Slowly changing dimensions are baked into the Merge functionality. To define an SCD Type 2 table, simply add the “Stored as SCD Type 2” clause to the merge statement. This automatically appends the _START_AT and _END_AT effective date columns

```
APPLY CHANGES INTO LIVE.clickstream_silver
FROM clickstream_raw
KEYS (current_page_title)
SEQUENCE BY import_date
STORED AS SCD TYPE 2
```

userId	name	city
125	Mercedes	Guadalajara

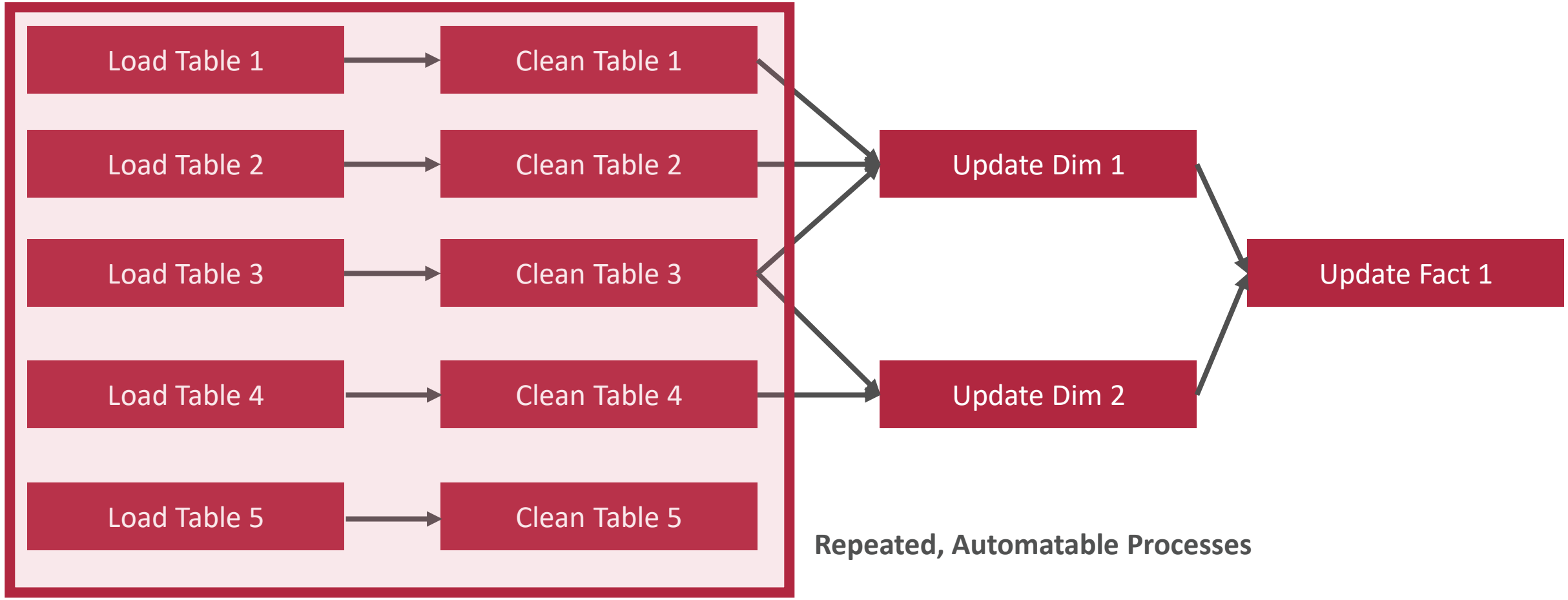
userId	name	city	__START_AT	__END_AT
123	Isabel	Monterrey	1	5
123	Isabel	Chihuahua	5	6
124	Raul	Oaxaca	1	null

SELECTIVE TABLE REFRESH



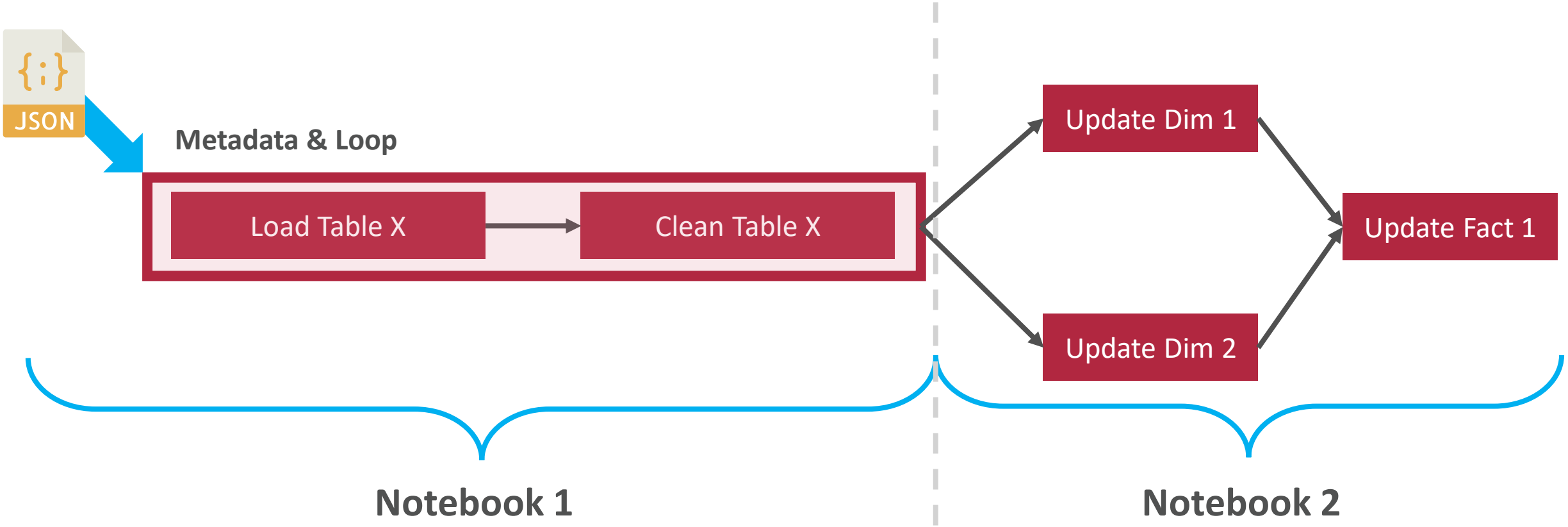
FRAMEWORKS AROUND FRAMEWORKS

AVOIDING THE TRAP OF MANUAL TABLE DEFINITIONS



Repeated, Automatable Processes

METADATA-DRIVEN DLT



FRAMEWORKS AROUND DLT

- Boilerplate DLT Notebooks
- Mix & Match Approaches

LEARN MORE ABOUT DLT

YouTube

ADVANCING ANALYTICS

ADVANCING ANALYTICS

READ THE DOCS!

https://www.databricks.com/discover/pages/getting-started-with-delta-live-tables

Thanks & Questions?

Twitter: @MrSiWhiteley

youtube.com/c/AdvancingAnalytics

AdvancingAnalytics.co.uk

ADVANCING ANALYTICS