data toolkit company
POWERED BY RED OLIVE

RED OLIVE

ROSIE

data science in a box

powered by a
Streamlined Integration Engine

Common
Business Problems

Difficulty migrating from on-premises to cloud.

Issues with scaling ML models and operationalising models into an ‘always-on’/self learning framework.

Data quality causes problems when building and implementing ML models.

Wrangling data consumes 80% of a data scientists time and is often not reusable.

Many ML models are built as a POC or in a sand box but never make it into production.

What is ROSIE?

ROSIE is a lightweight, containerised data transformation engine written in Python that uses SQL to query, process and transform data.

ROSIE can be run as a standalone from the command line or loaded as a Python module into another Python wrapper.

ROSIE: Navigate ML Ops Workflow

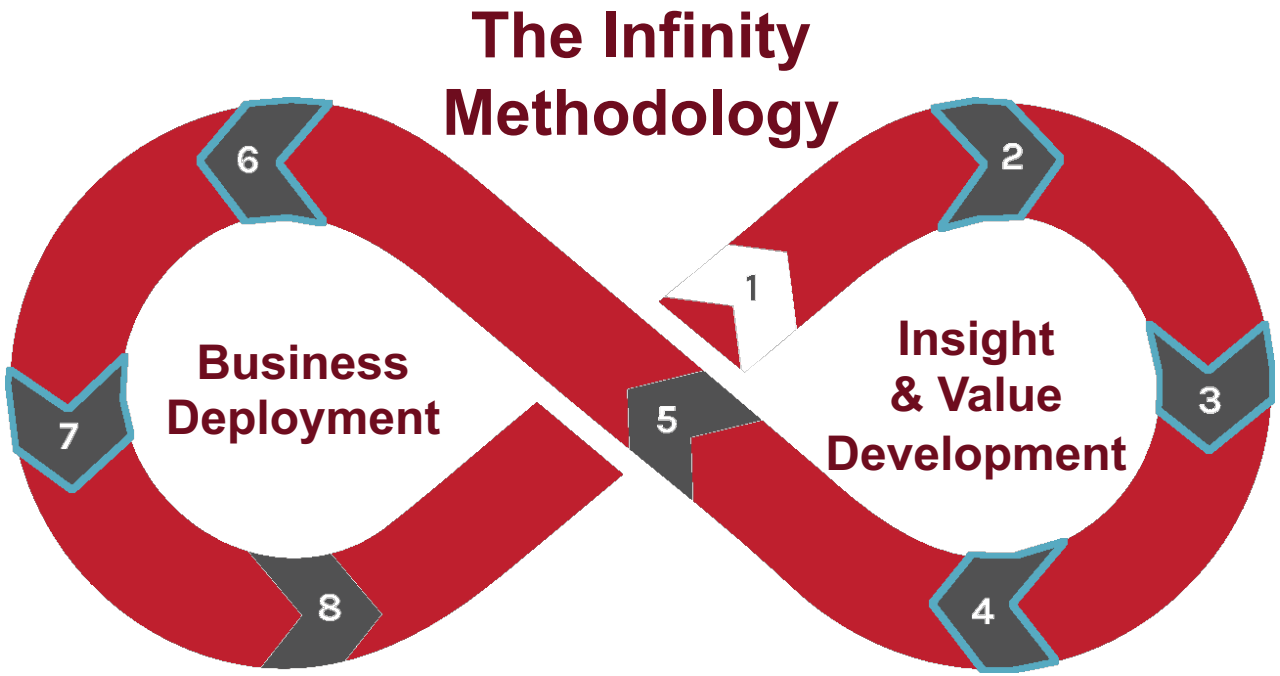
ROSIE supports the tried and tested data science methodology developed by Red Olive:

5 Evaluate
The model results.

6 Operationalise
ROSIE workflows are repeatable and flexible.

7 Embed
Run ROSIE from a scheduler and make it an embedded, automated process.

8 Self-learn
Self learning ML models can be integrated.



4 Model
Run custom built ML routines.

3 Engineer
Create a repeatable procedure to transform and manipulate data.

2 Discover
Use ROSIE to ingest data to allow for methodical discovery.

1 Identify
The business problem.

ROSIE Workflow

The basis of ROSIE is a list of tasks (a Workflow) written in a YAML file, which are performed by the engine.

Simple documentation.

Variables can be defined throughout.

A single task block.

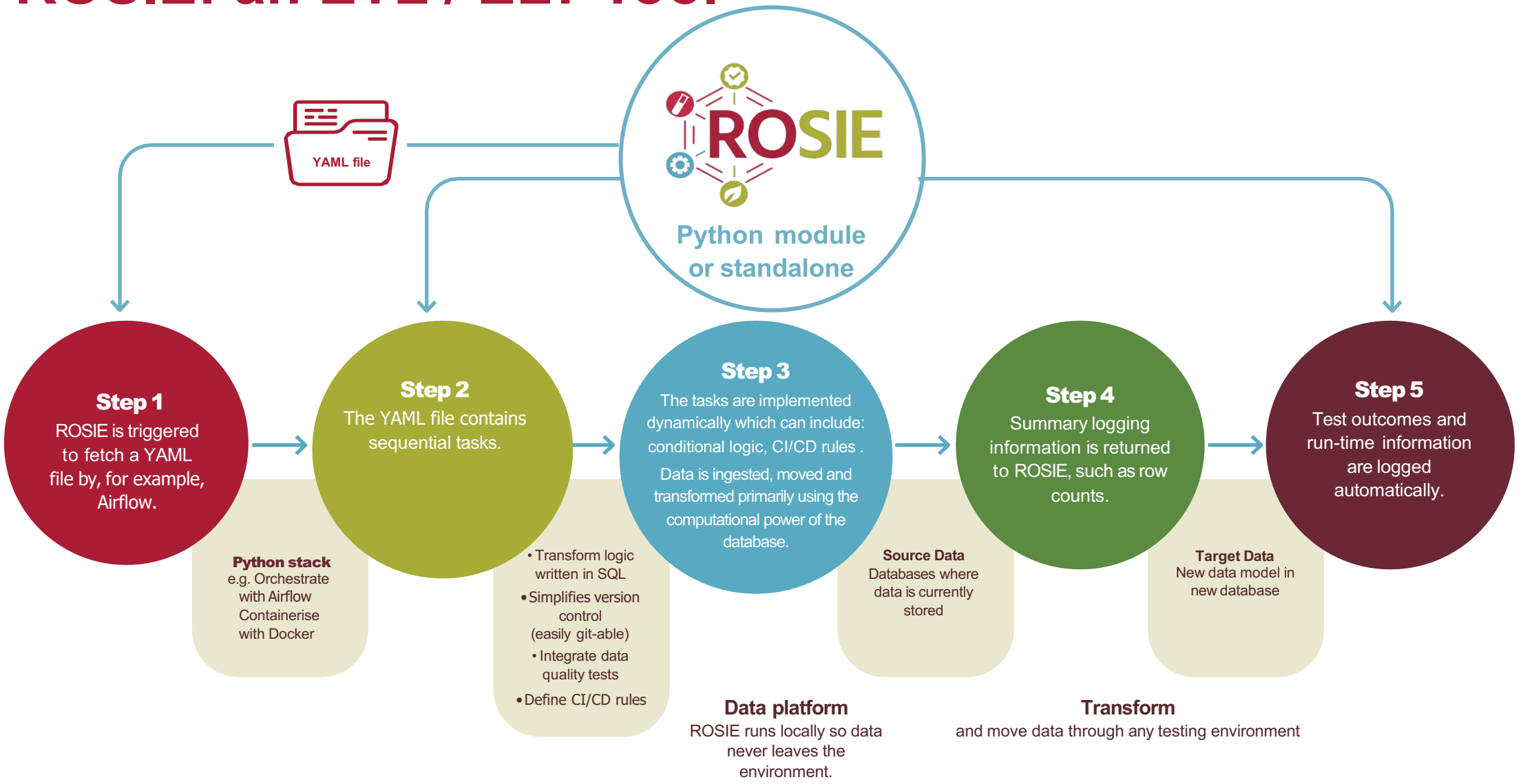
Additional files (SQL, python, etc.) can be referenced.

ROSIE also supports for loops.

```
name  BDL_2023_Demo
comment  Outline of a ROSIE workflow.

# Initialise task list
tasks
    assign
      variable  variable_1
      value  '"test_table'"

  # Define a simple task
  - id  task_1
    type  sql
    statement  "SELECT COUNT(*) FROM '{{ variable_1 }}'"
    assign_to  query_result
    log_after  "Result of query is {{query_result}}"

    id  execute_sql_query
    type  sql
    file  complex_sql_query.sql

    id  initialise_for_loop
    for  variable_name
    in  "'table_1', 'table_2', 'table_3'"
    .
    .
    .
```

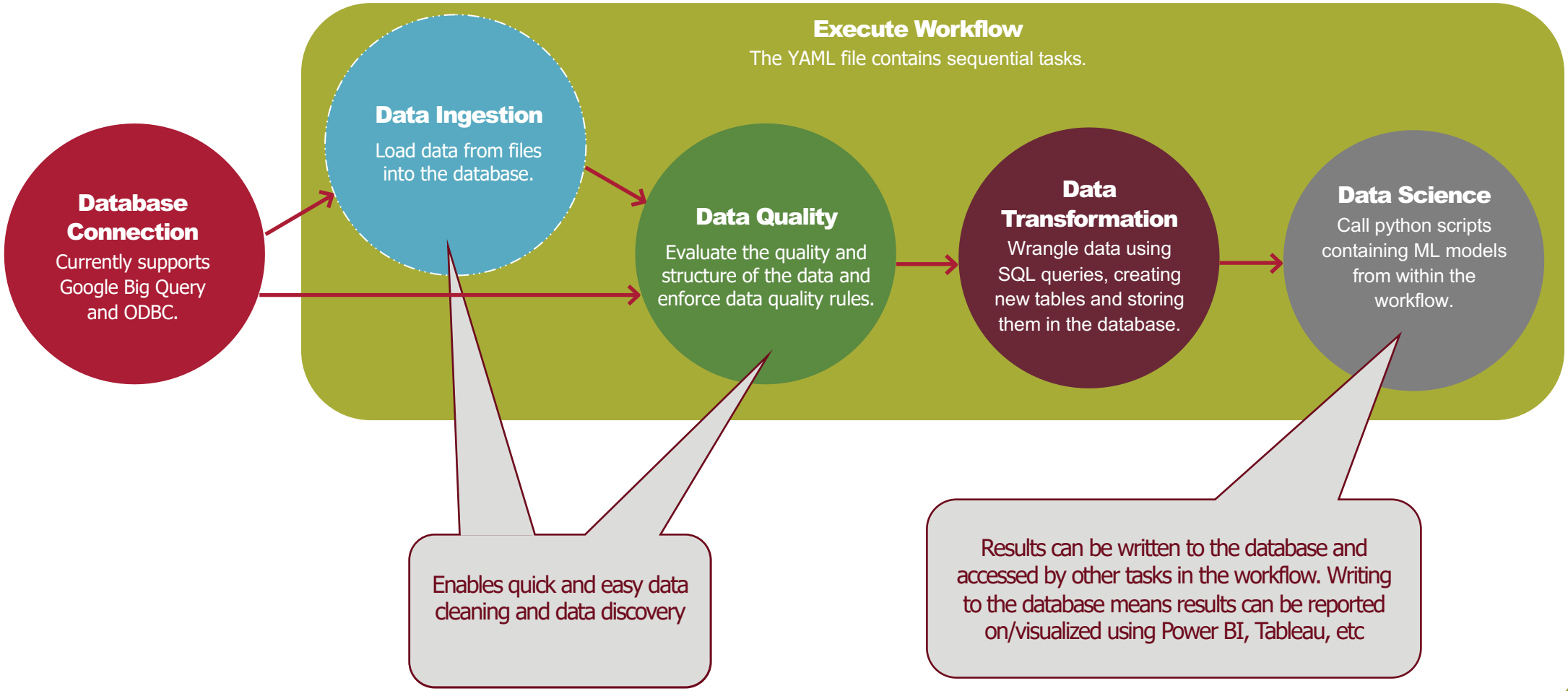
Provides a simple but effective way of continuously developing a Workflow

ROSIE: an ETL / ELT Tool

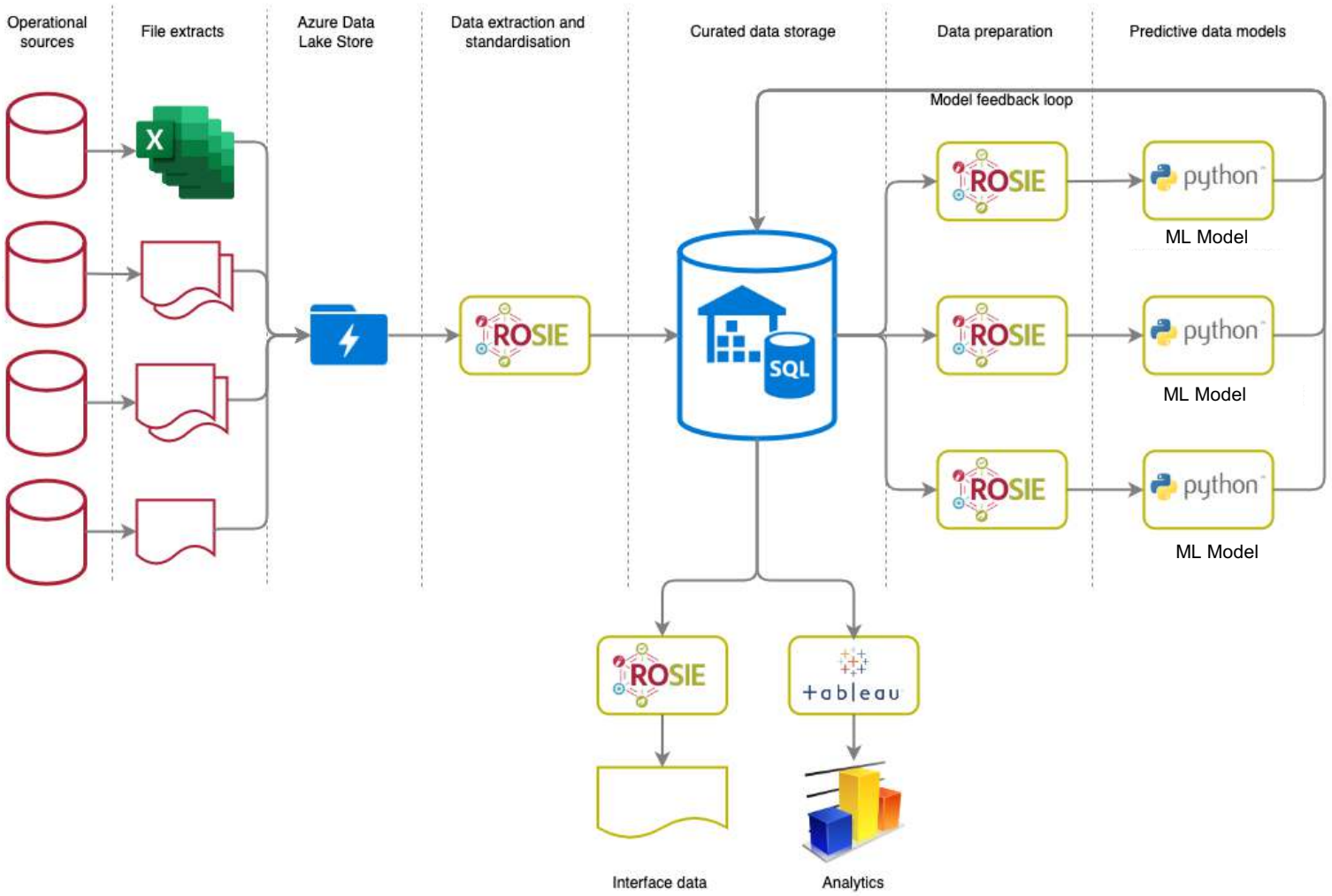


ROSIE: Data Science in a Box

Combining ELT, data quality and ML models



Example of a ROSIE Data Science Solution



ROSIE in action

What does ROSIE do?

Provides a quick and reliable ETL/ELT tool with no need for any platform specific coding languages, only SQL.

Allows data quality to be evaluated and rules enforced as an independent process or part of a larger Workflow.

Combines functionality to seamlessly manage and integrate MLOPs.

Why ROSIE?

ROSIE can help:

Conduct rapid and agile iterations of data experiments to support new ML models.

Reduces the deployment time of new models to live by enabling active performance monitoring.

Improve ROI while reducing the total cost of data ownership.

Enable faster, more streamlined data solutions using real-time data.

Integrate with self-learning ML models resulting in an always-on, evolving solution.

Please come and chat to our friendly team to find out more.

Get in touch to find out more about ROSIE or to arrange a demo

www.datatoolkit.co,

or email hello@datatoolkit.co

ROSIE: a Data Quality Tool

Build and deploy custom data quality rules that can be:

used to understand and report on the quality of existing data.

imposed on existing/new data using a combination of SQL queries and the built-in assertion verbs.

implemented as a specific data quality process or as tasks in a larger workflow.

Additionally, ROSIE contains pre-built data quality workflows that can be dynamically applied and adapted to profile existing data.

ROSIE:DQ

Supported Assertion Verbs:

- Assert Equal
- Assert Not-Equal
- Assert in List
- Assert not in List
- Assert RegEx
- Assert Pass
- Assert Fail

How does ROSIE work?

RED OLIVE

ROSIE is triggered to fetch a YAML file, for example, by Airflow.

The YAML file contains sequential tasks written using SQL or shell-based commands which are returned to **ROSIE**.

The YAML file defines a series of tasks, as either primitive assignments, SQL, shell or API directives. There is support for case statements and conditional steps, logging and the setting of an exit status.

The tasks are implemented dynamically which can include conditional logic and CI/CD rules. Data is moved and transformed from the source to the target location and is not routed via **ROSIE** – all data processing happens in the database.

Summary logging information is returned to **ROSIE**, such as row counts.

Test outcomes and runtime information is logged automatically.