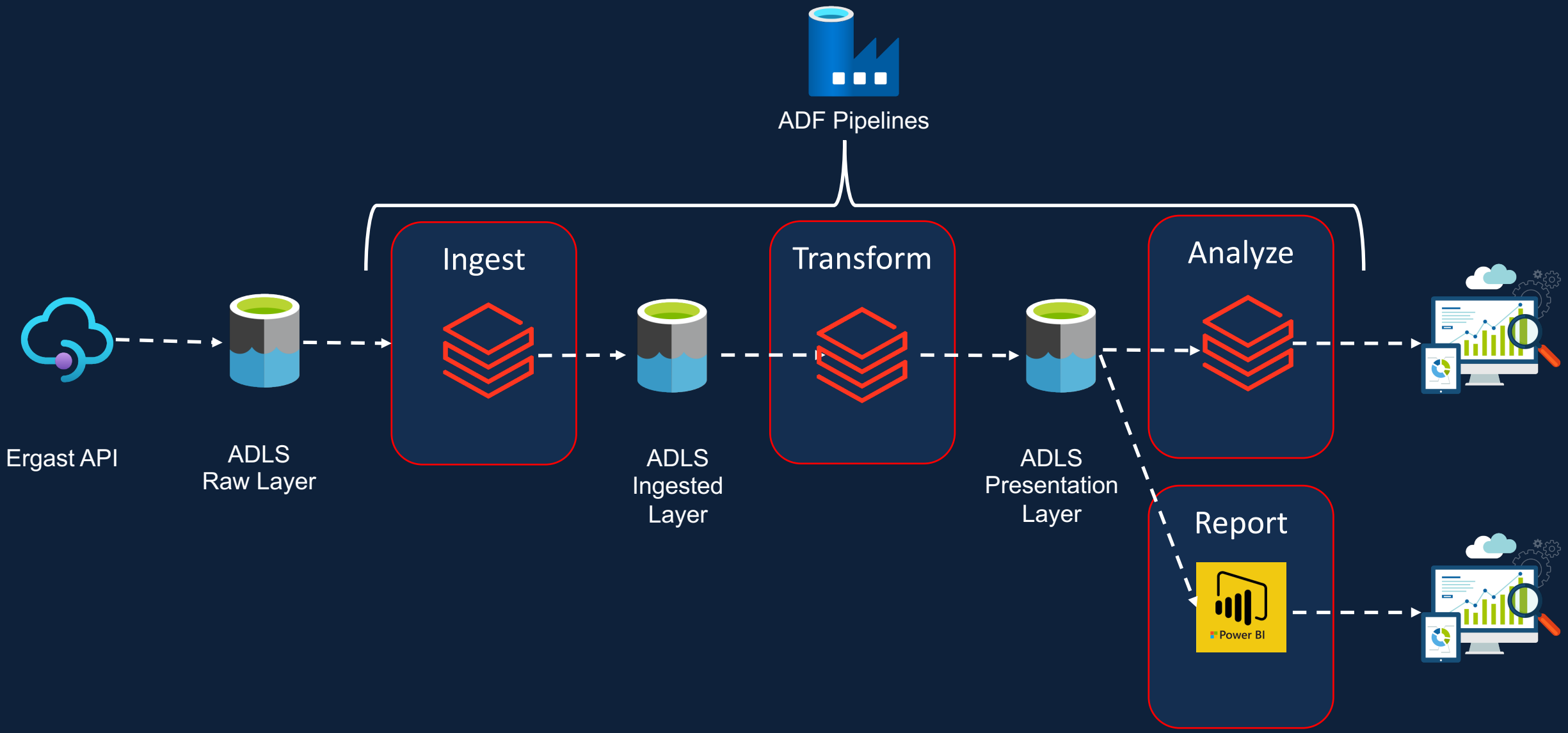
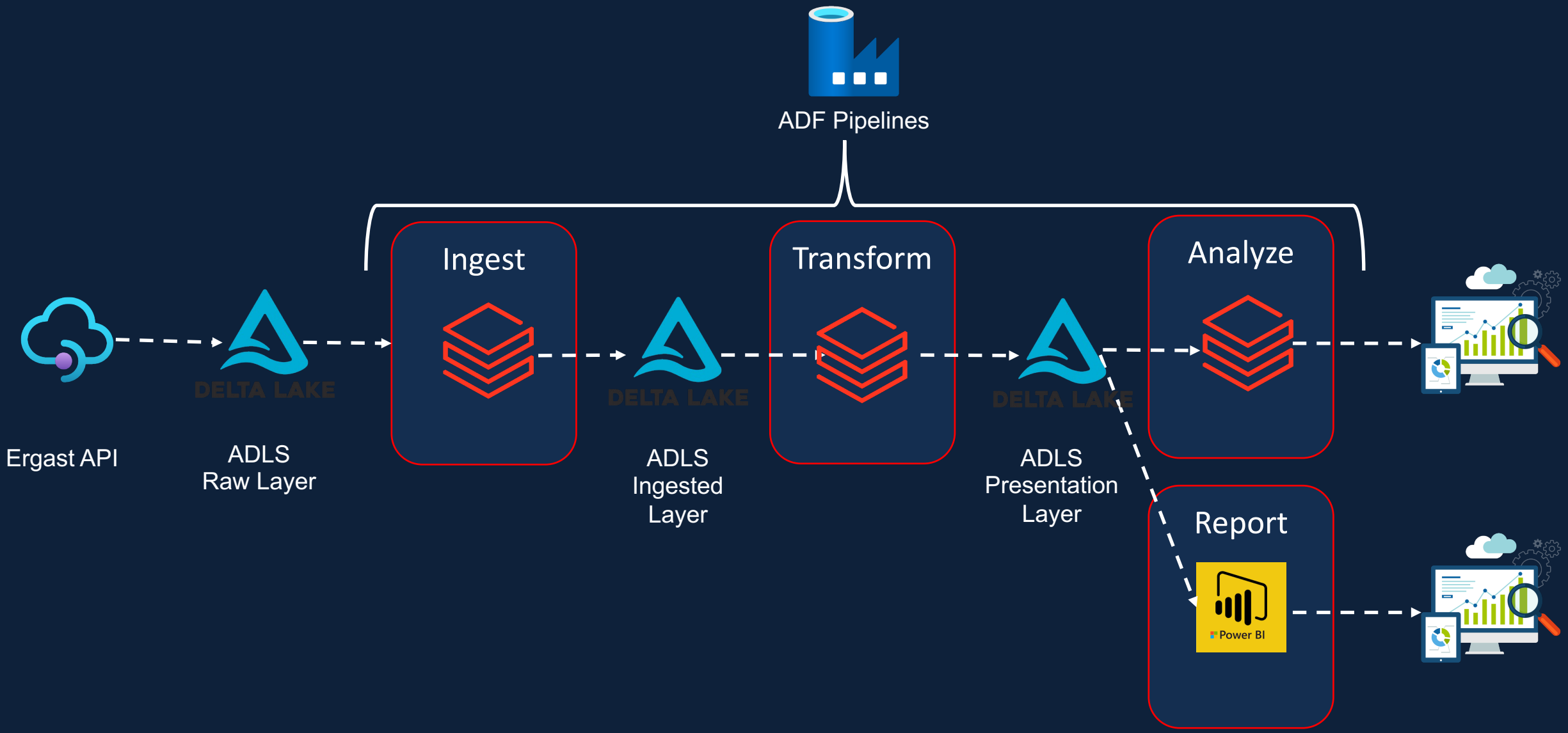
Formula1 Cloud Data Platform

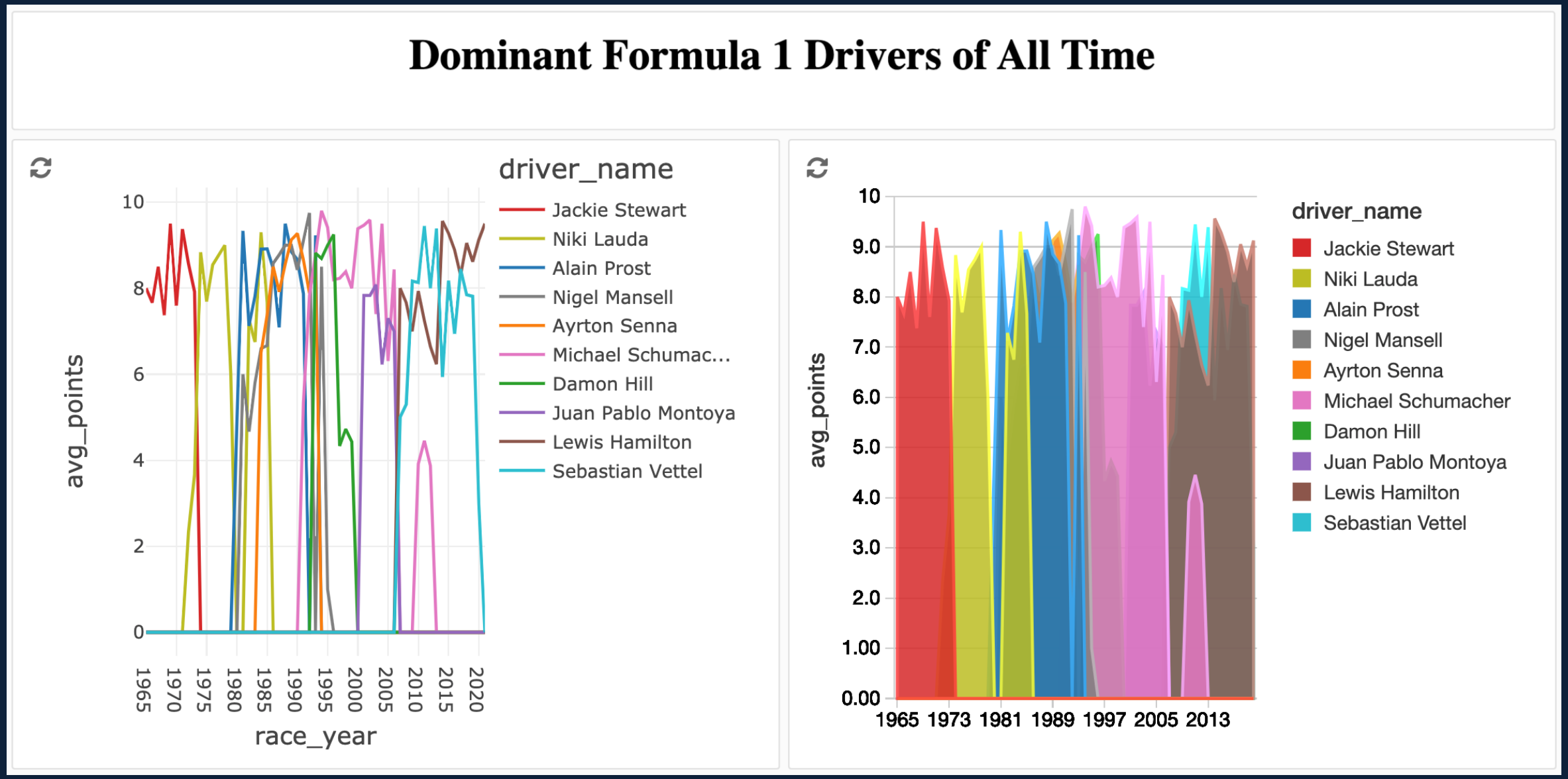
Formula1 Cloud Data Platform



Formula1 Cloud Data Platform

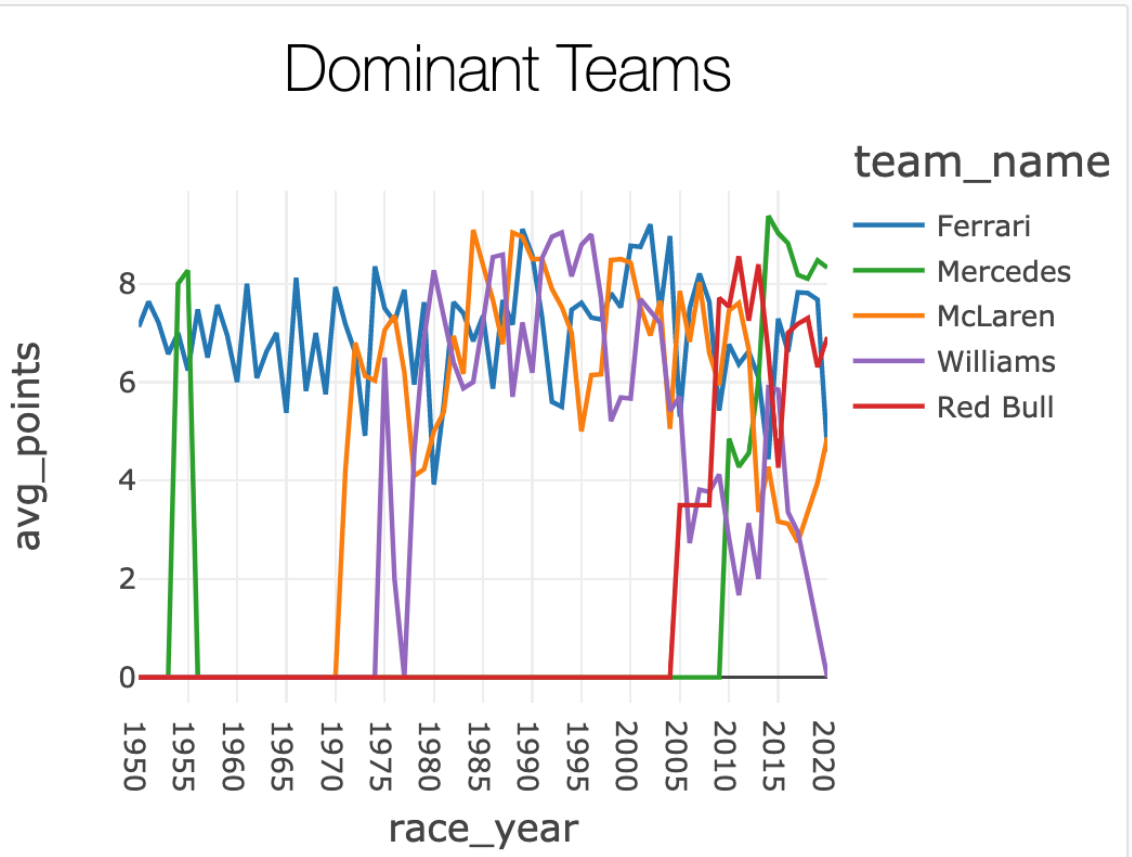


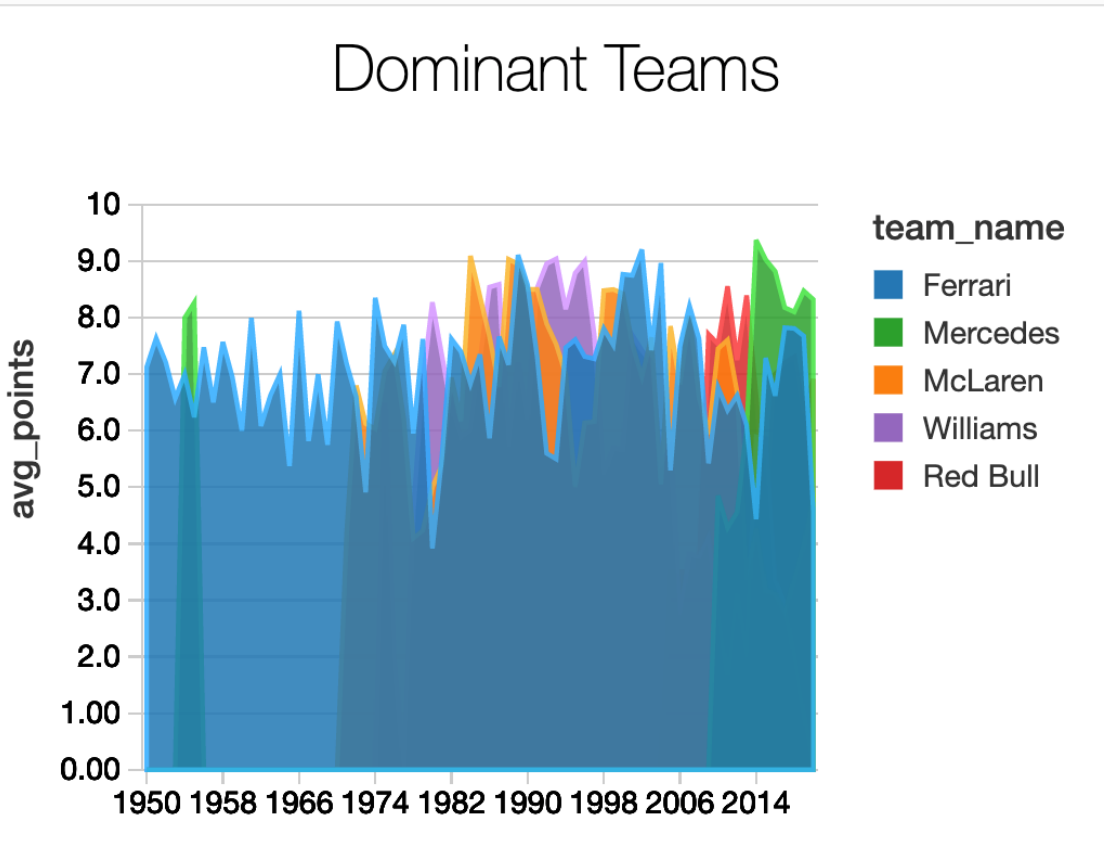
Formula1 Cloud Data Platform



Formula1 Cloud Data Platform

Dominant Formula 1 Teams of All Time





Pre-requisites

- All code and step-by-step instructions provided
- Basic Python programming knowledge required
- Basic SQL knowledge required
- Cloud fundamentals will be beneficial, but not mandatory
- Azure Account

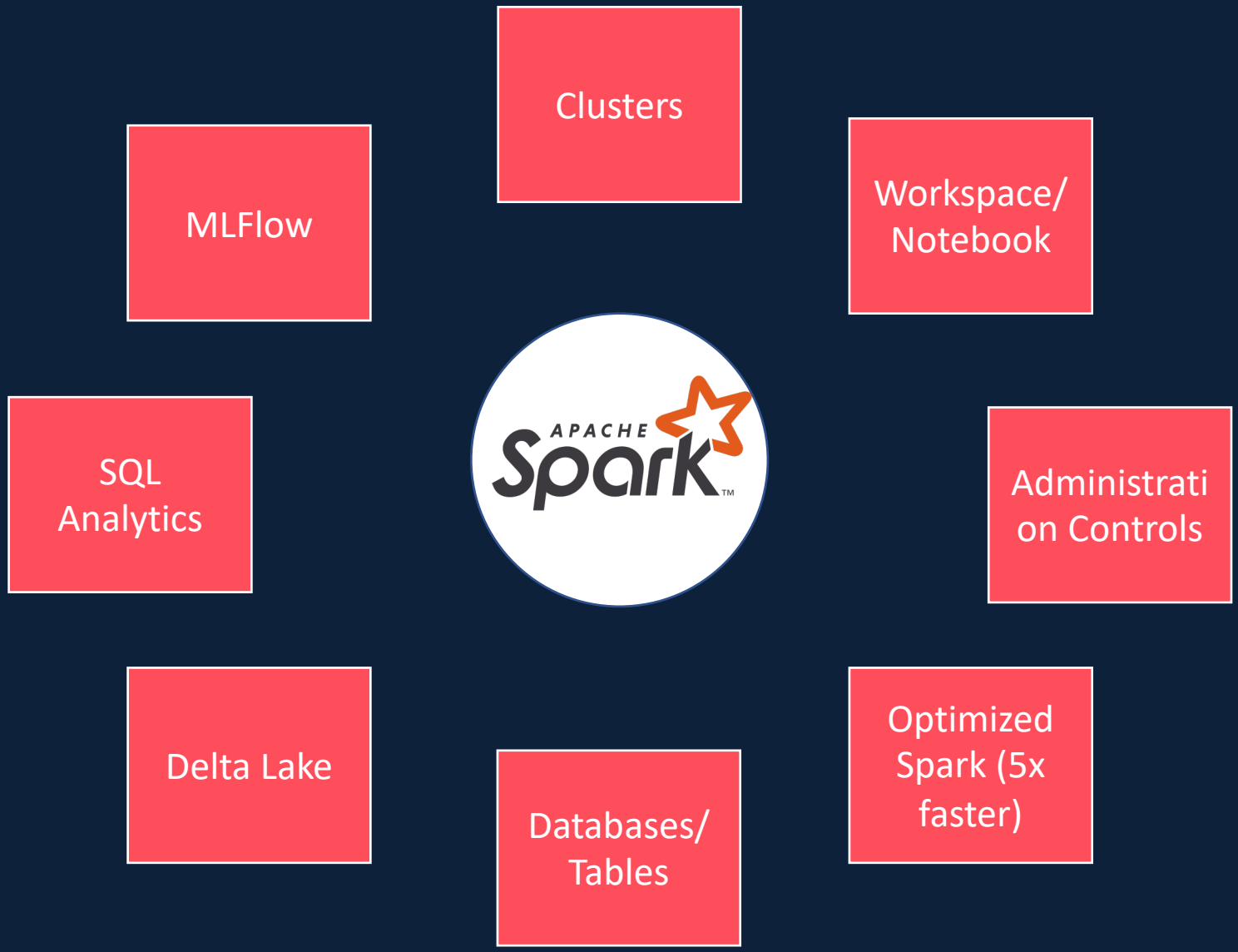
Introduction to Azure Databricks

Apache Spark

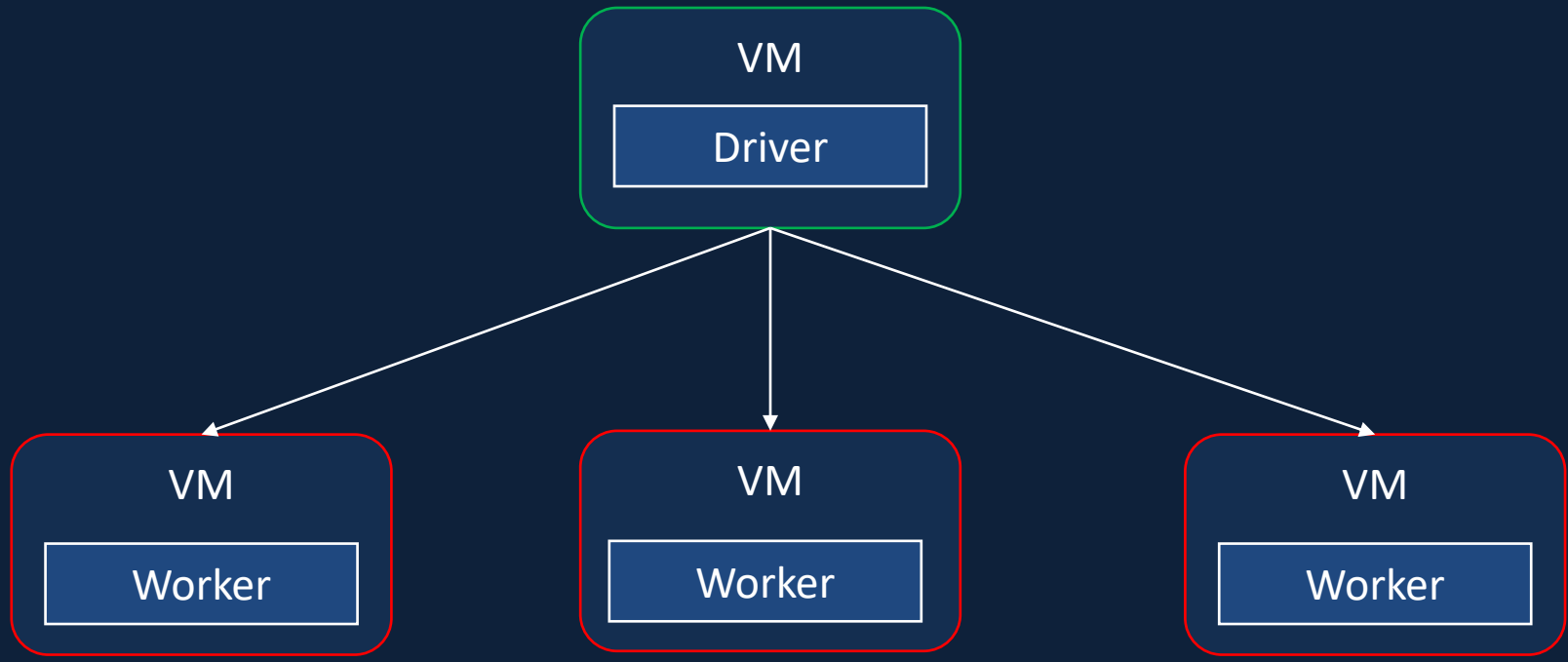
Apache Spark is a lightning-fast unified analytics engine for big data processing and machine learning

- 100% Open source under Apache License
- Simple and easy to use APIs
- In-memory processing engine
- Distributed computing Platform
- Unified engine which supports SQL, streaming, ML and graph processing

Databricks


Clusters
MLFlow
Workspace/ Notebook
APACHE Spark
SQL Analytics
Administrati on Controls
Delta Lake
Databases/ Tables
Optimized Spark (5x faster)

Databricks Cluster



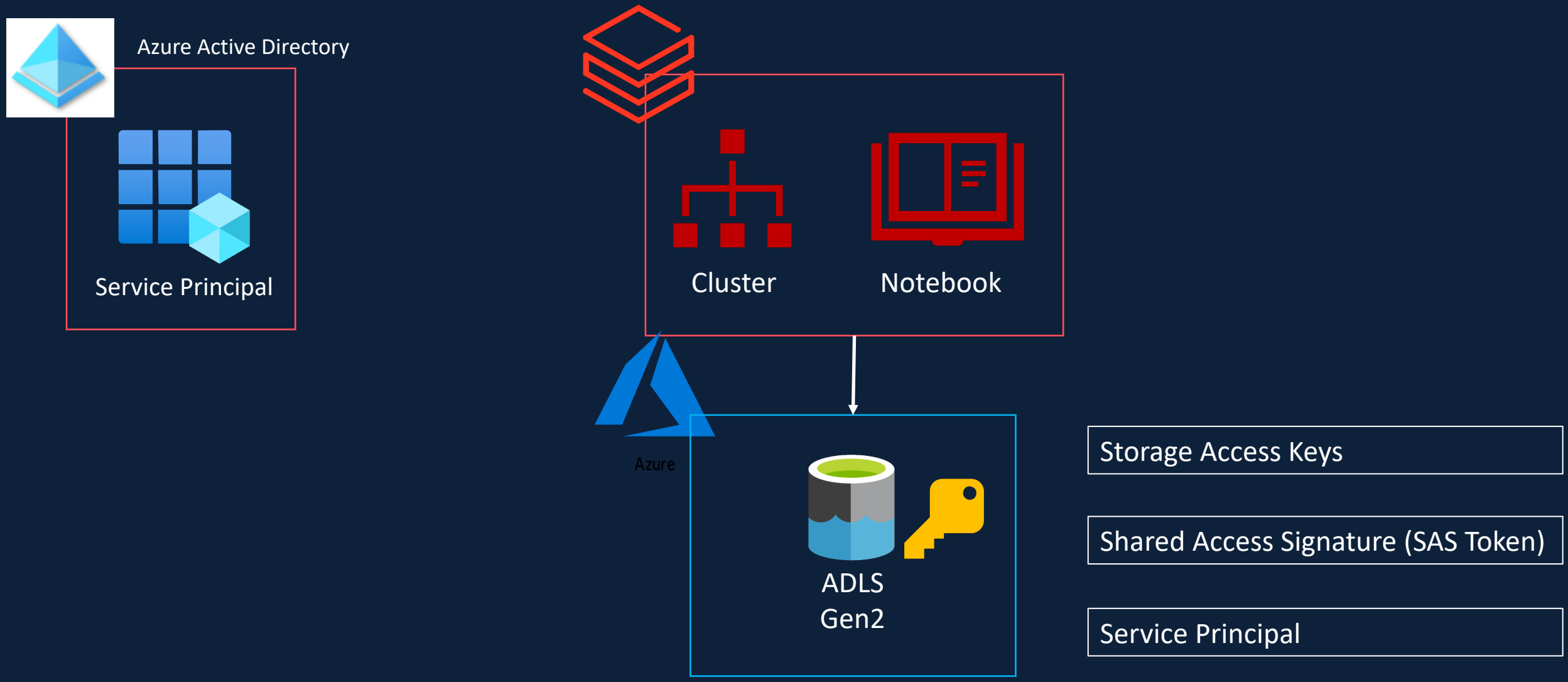
Cluster Types

All Purpose	Job Cluster
Created manually	Created by Jobs
Persistent	Terminated at the end of the job
Suitable for interactive workloads	Suitable for automated workloads
Shared among many users	Isolated just for the job
Expensive to run	Cheaper to run

Databricks Notebooks

- What’s a notebook
- Creating a notebook
- Magic Commands
- Databricks Utilities
- Import Project Solution Notebooks

Access Azure Data Lake - Section Overview

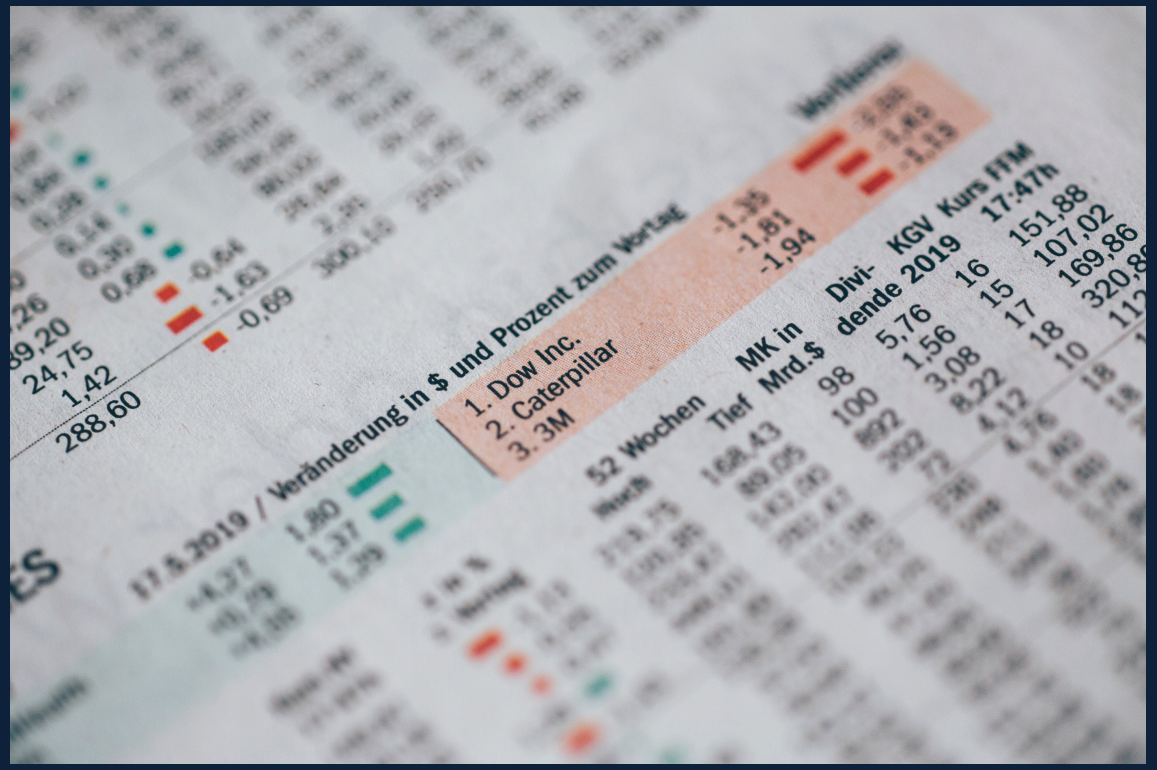


Create Azure Data Lake Gen2 Storage (ADLS Gen2)

Access Keys – Spark Configuration

spark.conf.set("fs.azure.account.key.<storage-account>.dfs.core.windows.net", "<access key>")

```
spark.conf.set(
    "fs.azure.account.key.formula1dl.dfs.core.windows.net",
    "3ORoyW+laxV39N0JZ7XWRSWSOimUGp2lKdE65nRbHrJ9UHc1fqLyJN+j+Qunhev+YL8+CwPLenWn+AStg8bfJg==")
```



Data Overview

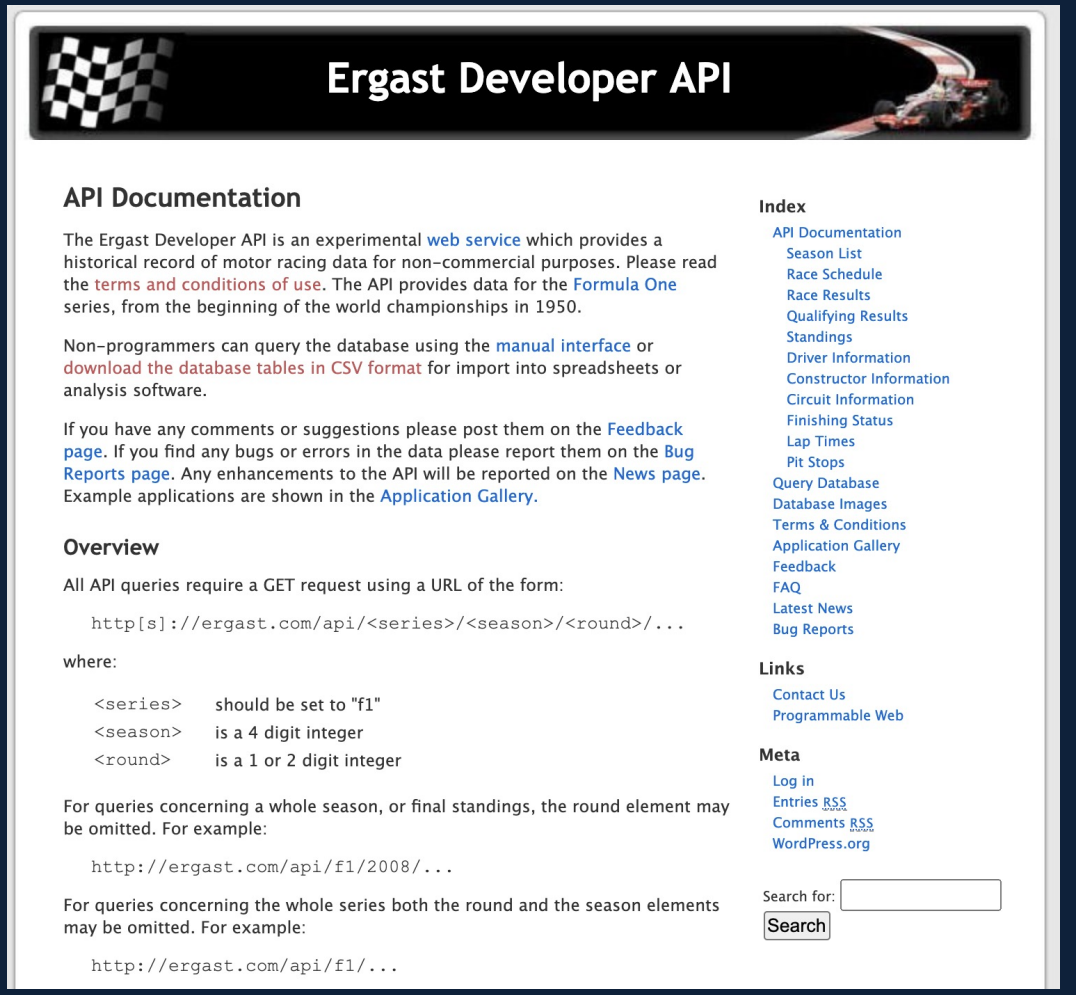

7
HUBLOT
Santander
V-Power
Santander
MAHLE
Ray-Ban
ups
infor
Santander

Formula1

Formula1 Overview

Formula1 Data Source

http://ergast.com/mrd/



Ergast Developer API

API Documentation

The Ergast Developer API is an experimental web service which provides a historical record of motor racing data for non-commercial purposes. Please read the terms and conditions of use. The API provides data for the Formula One series, from the beginning of the world championships in 1950.

Non-programmers can query the database using the manual interface or download the database tables in CSV format for import into spreadsheets or analysis software.

If you have any comments or suggestions please post them on the Feedback page. If you find any bugs or errors in the data please report them on the Bug Reports page. Any enhancements to the API will be reported on the News page. Example applications are shown in the Application Gallery.

Overview

All API queries require a GET request using a URL of the form:

```
http[s]://ergast.com/api/<series>/<season>/<round>/...
```

where:

`<series>`	should be set to "f1"
`<season>`	is a 4 digit integer
`<round>`	is a 1 or 2 digit integer

For queries concerning a whole season, or final standings, the round element may be omitted. For example:

```
http://ergast.com/api/f1/2008/...
```

For queries concerning the whole series both the round and the season elements may be omitted. For example:

```
http://ergast.com/api/f1/...
```

Index

- API Documentation
  - Season List
  - Race Schedule
  - Race Results
  - Qualifying Results
  - Standings
  - Driver Information
  - Constructor Information
  - Circuit Information
  - Finishing Status
  - Lap Times
  - Pit Stops
- Query Database
- Database Images
- Terms & Conditions
- Application Gallery
- Feedback
- FAQ
- Latest News
- Bug Reports

Links

- Contact Us
- Programmable Web

Meta

- Log in
- Entries RSS
- Comments RSS
- WordPress.org

Search for: Search

Formula1 Data Files

Circuits	CSV
Races	CSV
Constructors	Single Line JSON
Drivers	Single Line Nested JSON
Results	Single Line JSON
PitStops	Multi Line JSON
LapTimes	Split CSV Files
Qualifying	Split Multi Line JSON Files

Import Raw Data to Data Lake

Data Ingestion Requirements

- Ingest All 8 files into the data lake
- Ingested data must have the schema applied
- Ingested data must have audit columns
- Ingested data must be stored in columnar format (i.e., Parquet)
- Must be able to analyze the ingested data via SQL
- Ingestion logic must be able to handle incremental load

Data Transformation Requirements

- Join the key information required for reporting to create a new table.
- Join the key information required for Analysis to create a new table.
- Transformed tables must have audit columns
- Must be able to analyze the transformed data via SQL
- Transformed data must be stored in columnar format (i.e., Parquet)
- Transformation logic must be able to handle incremental load

Reporting Requirements

- Driver Standings
- Constructor Standings

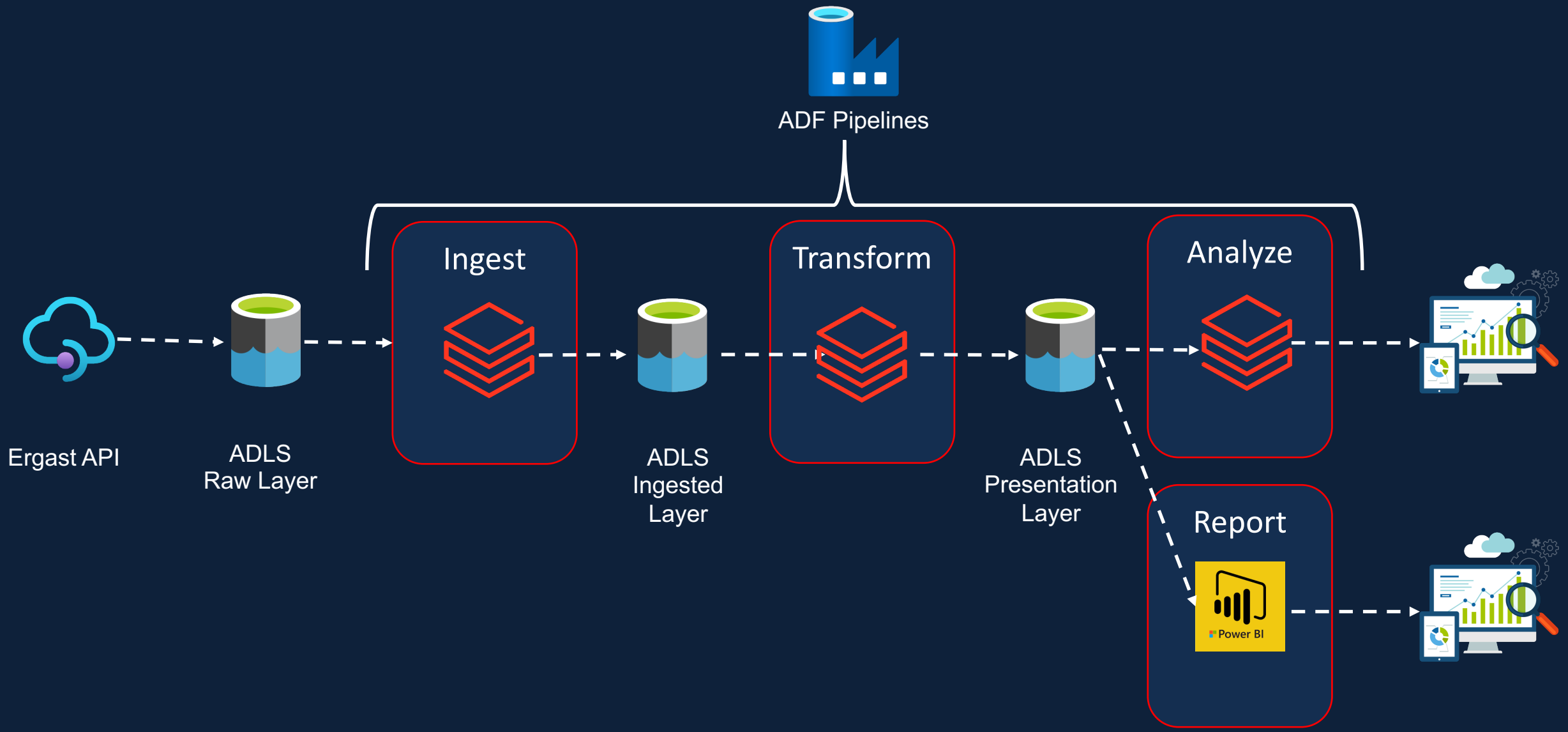
Analysis Requirements

- Dominant Drivers
- Dominant Teams
- Visualize the outputs
- Create Databricks Dashboards

Scheduling Requirements

- Scheduled to run every Sunday 10PM
- Ability to monitor pipelines
- Ability to re-run failed pipelines
- Ability to set-up alerts on failures

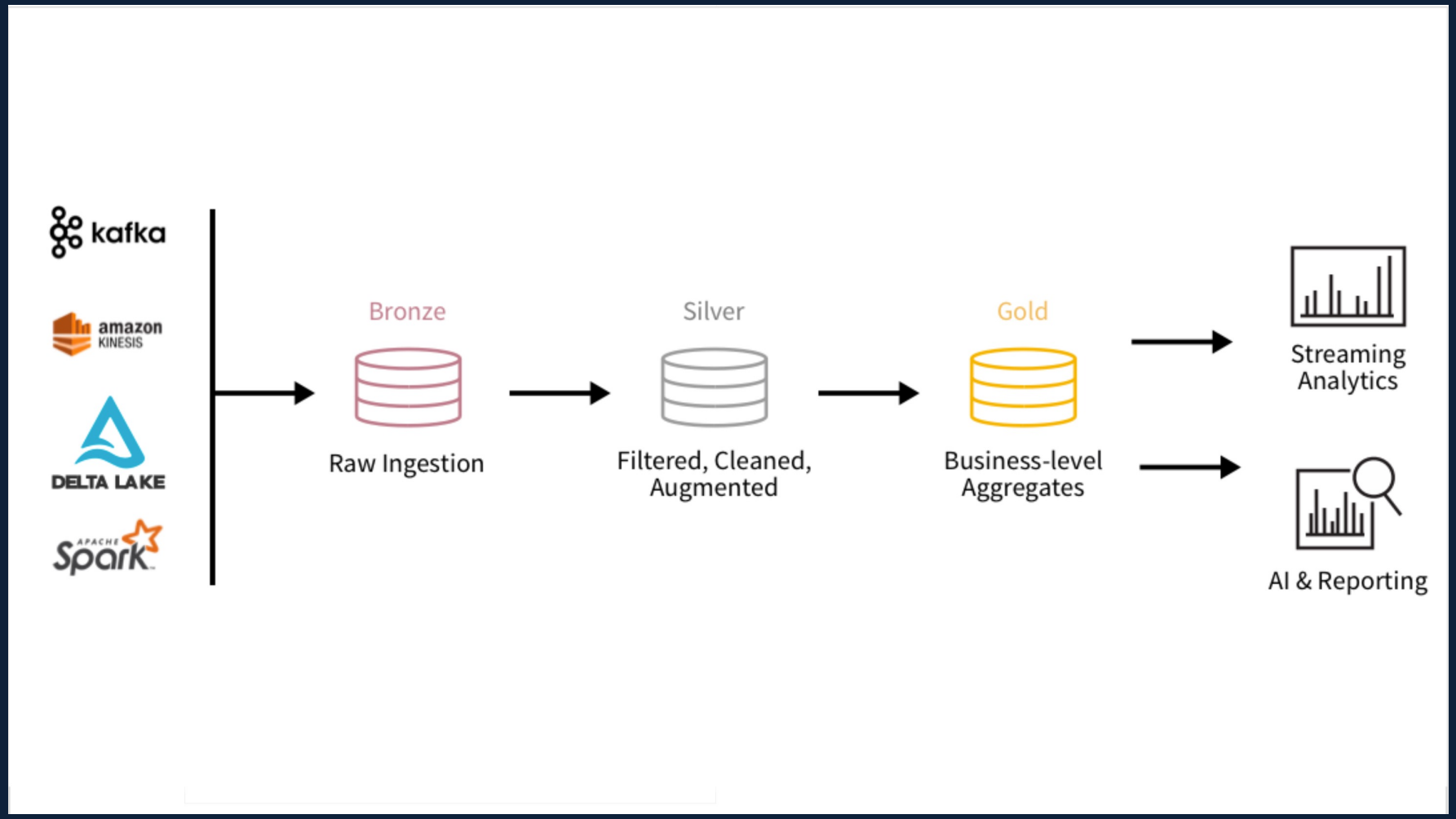
Solution Architecture



Azure Databricks Modern Analytics Architecture

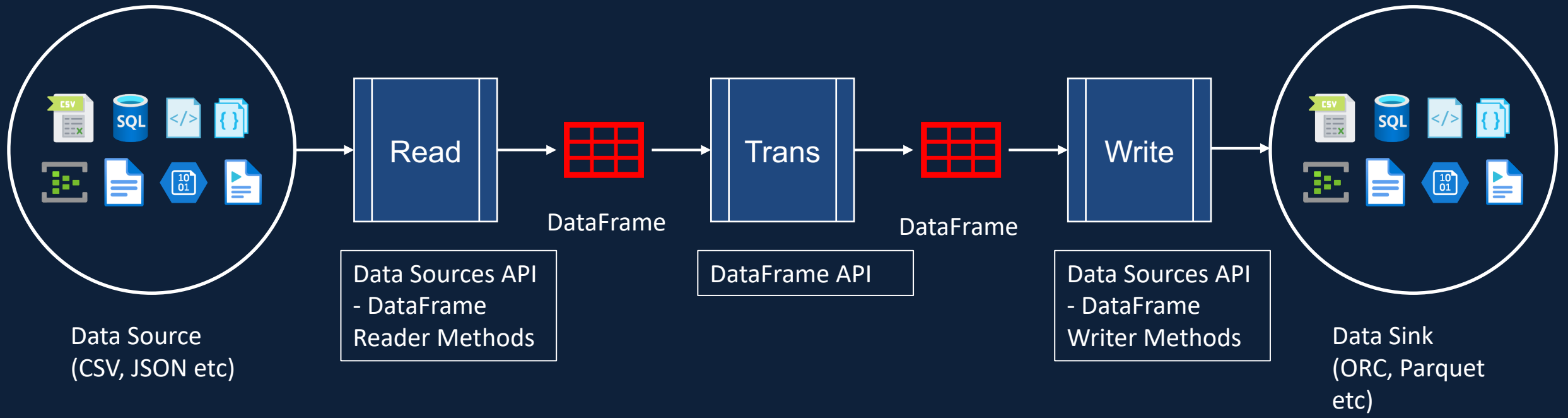
https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/azure-databricks-modern-analytics-architecture

Databricks Architecture



https://databricks.com/solutions/data-pipelines

Spark DataFrame

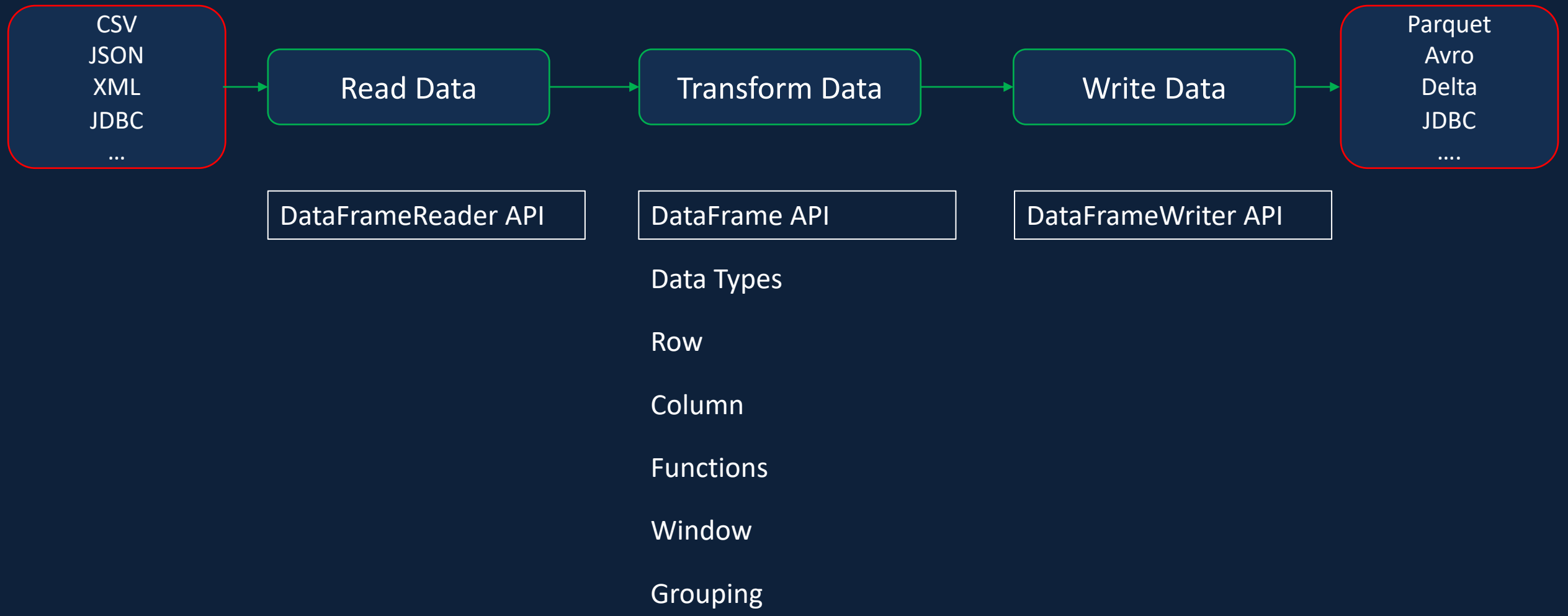


Data Ingestion Overview

Data Ingestion Requirements

- Ingest All 8 files into the data lake
- Ingested data must have the schema applied
- Ingested data must have audit columns
- Ingested data must be stored in columnar format (i.e., Parquet)
- Must be able to analyze the ingested data via SQL
- Ingestion logic must be able to handle incremental load

Data Ingestion Overview



Data Ingestion Overview

File Name	File Type	Assignment/ Class work
Circuits	CSV	Class work
Races	CSV	Assignment
Constructors	Single Line JSON	Class work
Results	Single Line JSON	Assignment
Drivers	Single Line Nested JSON	Class work
PitStops	Multi Line JSON	Class work
LapTimes	Split CSV Files	Class work
Qualifying	Split Multi Line JSON Files	Assignment

Data Ingestion - Circuits

Column Name
circuitId
circuitRef
name
location
country
lat
long
alt
url (dropped)

Column Name	Data Type
circuit_id (Renamed)	Integer
circuit_ref (Renamed)	String
name	String
location	String
country	String
latitude (Renamed)	Double
longitude (Renamed)	Double
altitude (Renamed)	Integer
ingestion_date (new)	Timestamp

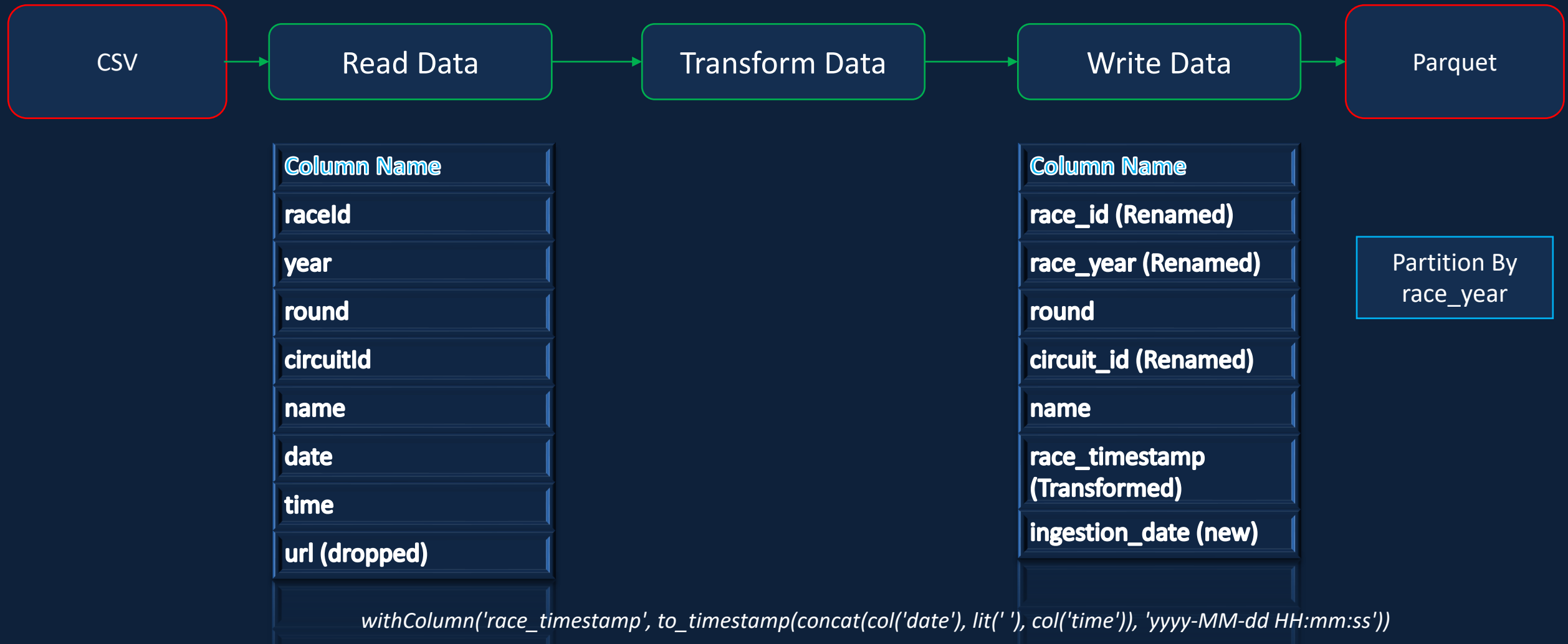
Data Ingestion – Races (Assignment)

Column Name
raceId
year
round
circuitId
name
date
time
url (dropped)

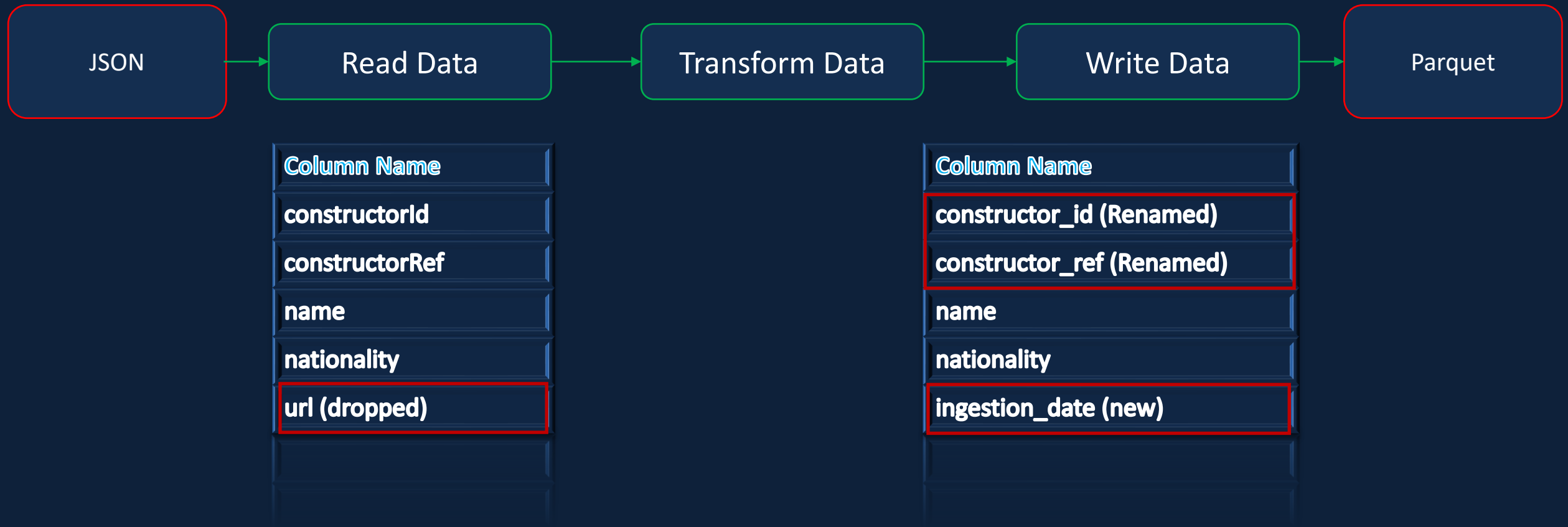
Column Name	Data Type
race_id (Renamed)	Integer
race_year (Renamed)	Integer
round	Integer
circuit_id (Renamed)	Integer
name	String
race_timestamp (Transformed)	Timestamp
ingestion_date (new)	Timestamp

withColumn('race_timestamp', to_timestamp(concat(col('date'), lit(' '), col('time')), 'yyyy-MM-dd HH:mm:ss'))

Data Ingestion – Races (Partition By)



Data Ingestion - Constructors

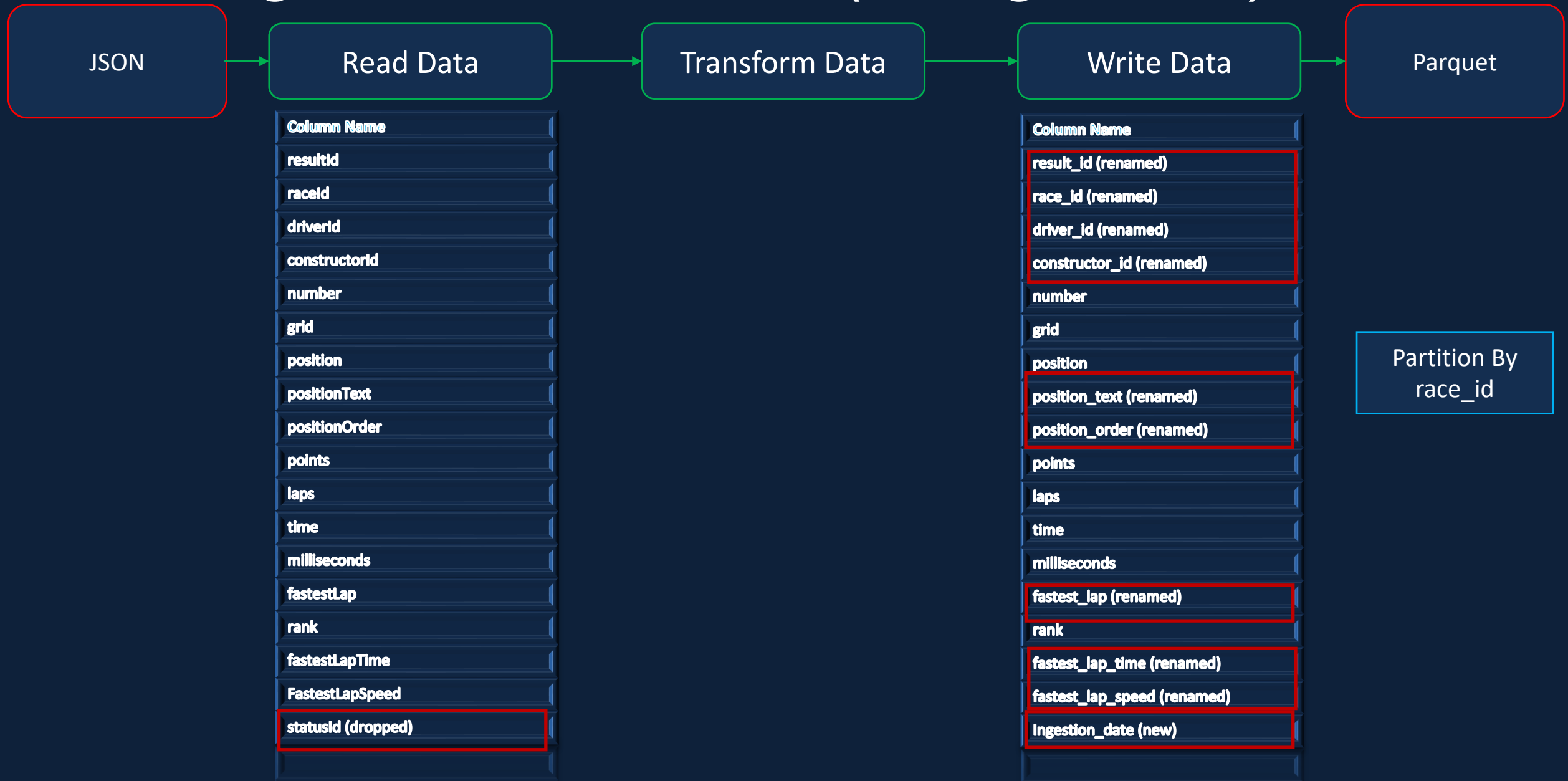


Data Ingestion - Drivers

Column Name
driverId
driverRef
number
code
name.forename
name.surname
dob
nationality
url (dropped)

Column Name
driver_id (Renamed)
driver_ref (Renamed)
number
code
name(transformed)
dob
nationality
ingestion_date (new)

Data Ingestion – Results (Assignment)

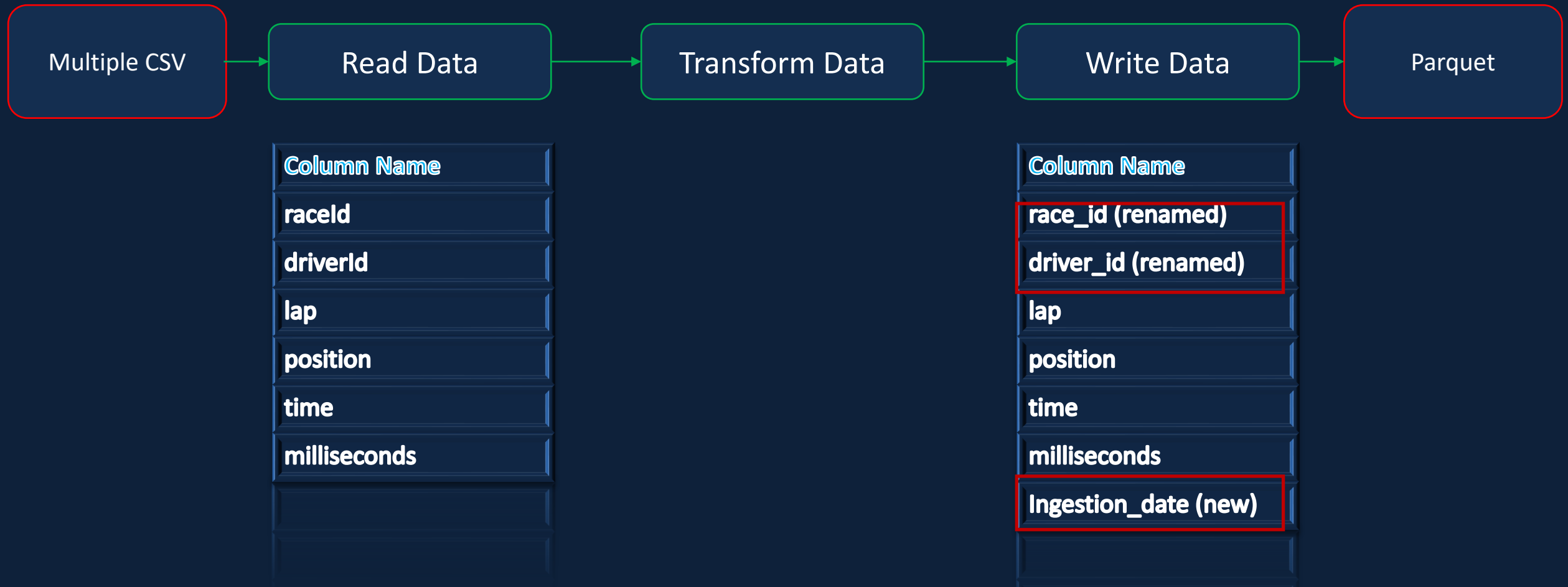


Data Ingestion - Pitstops

Column Name
raceId
driverId
stop
lap
time
duration
milliseconds

Column Name
race_id (renamed)
driver_id (renamed)
stop
lap
time
duration
milliseconds
Ingestion_date (new)

Data Ingestion - Laptimes

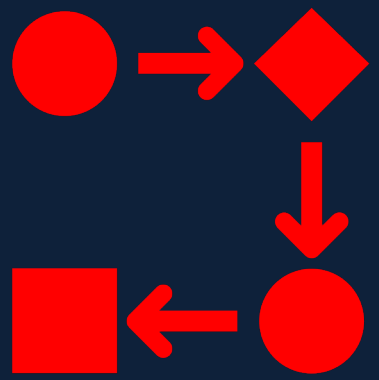


Data Ingestion – Qualifying (Assignment)

Column Name
qualifyingId
raceId
driverId
constructorId
number
position
q1
q2
q3

Column Name
qualifying_id(renamed)
race_id(renamed)
driver_id(renamed)
constructor_id(renamed)
number
position
q1
q2
q3
Ingestion_date(new)

Databricks Workflows



- Include notebook
- Defining notebook parameters
- Notebook workflow
- Databricks Jobs

Include notebook (%run)

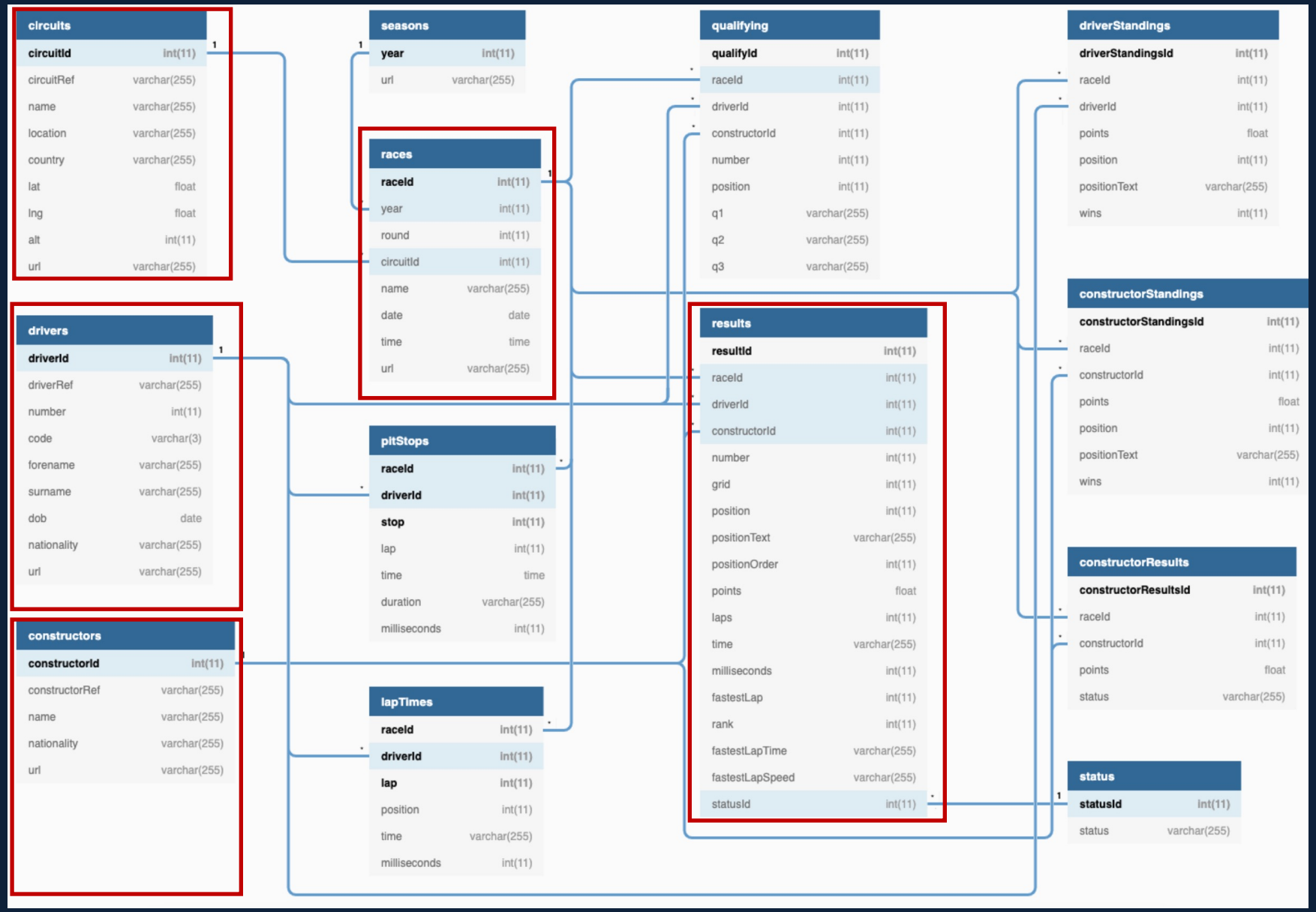
Passing Parameters (widgets)

Notebook Workflow

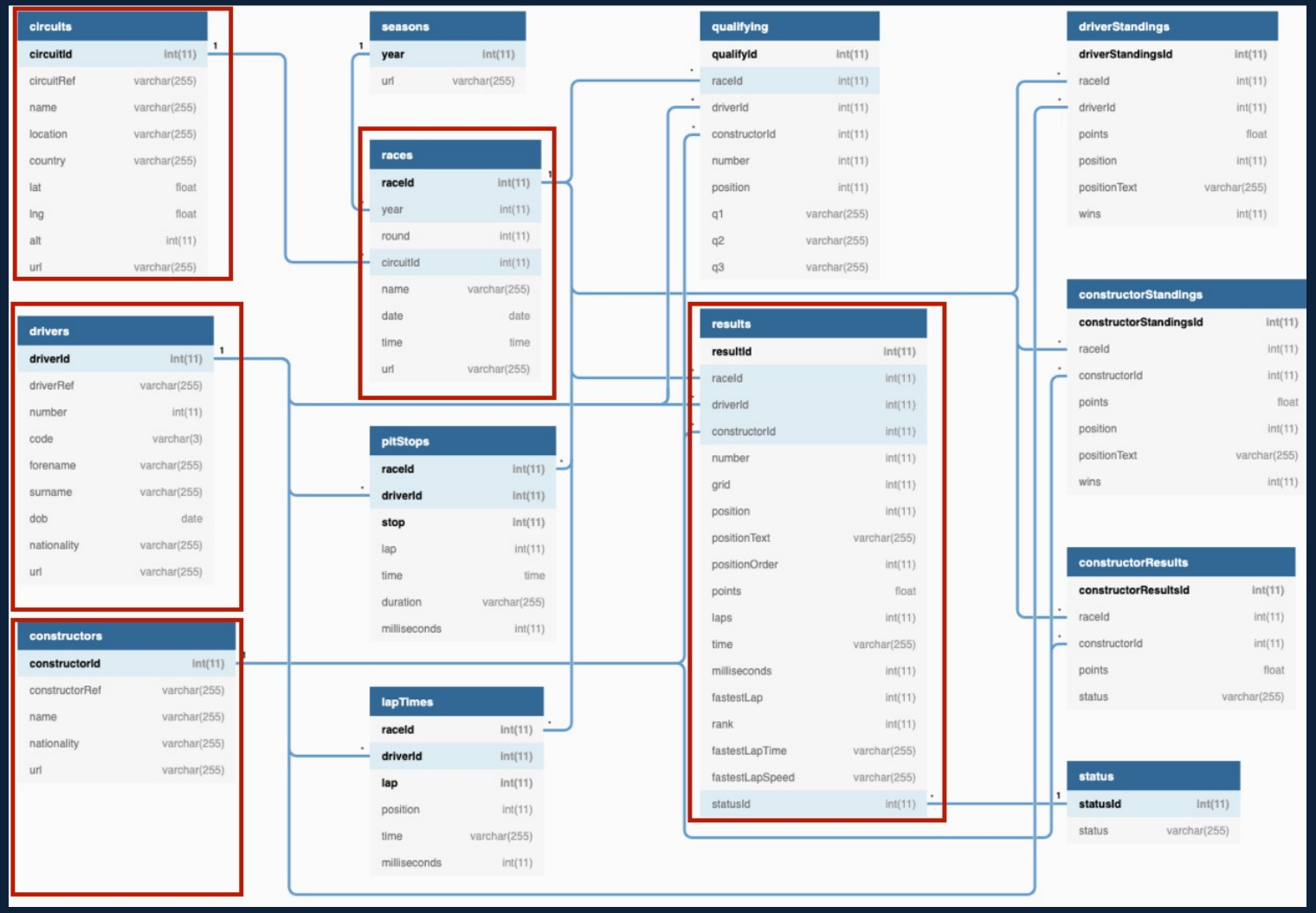
Join Transformation

Race Results

Join –Race Results

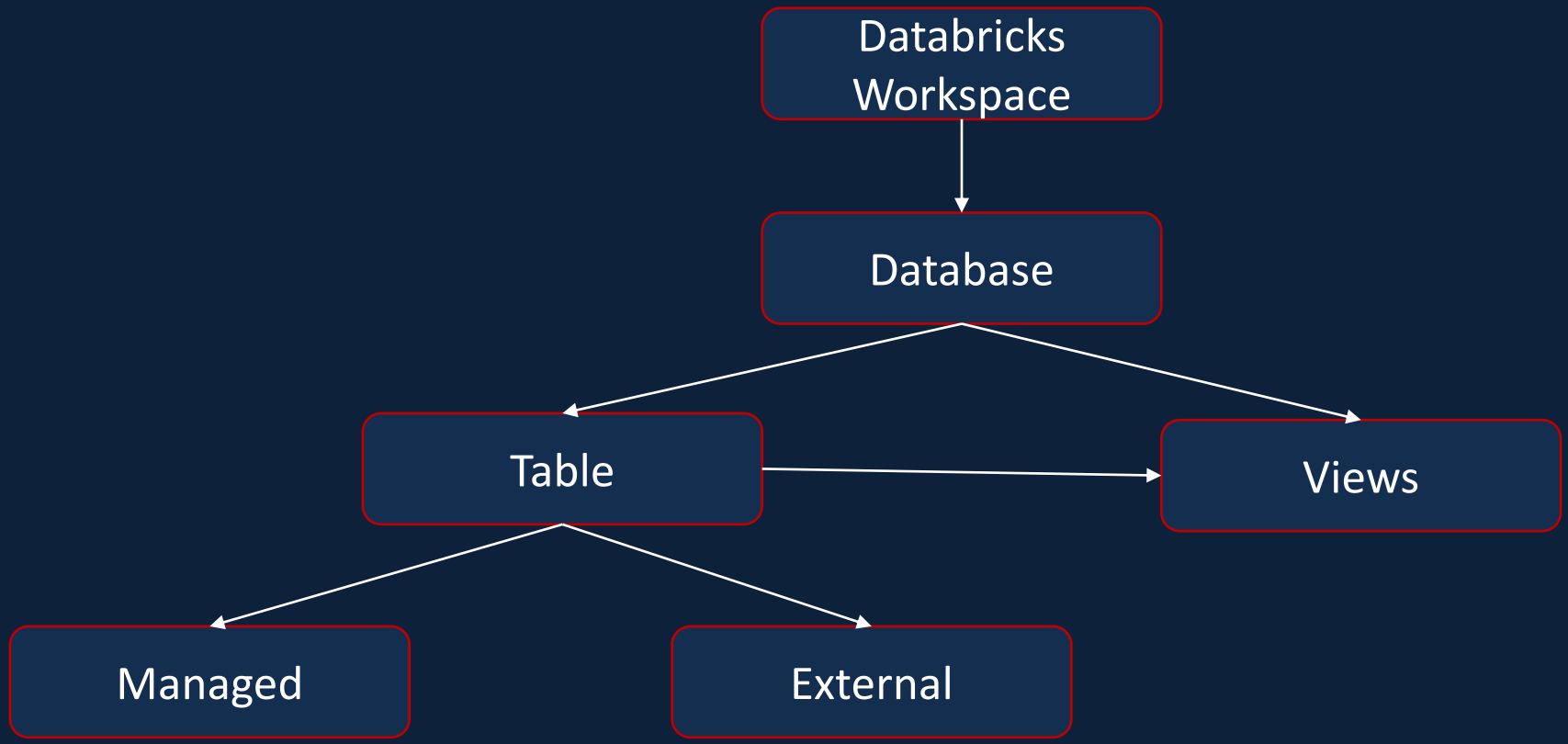


Join –Race Results

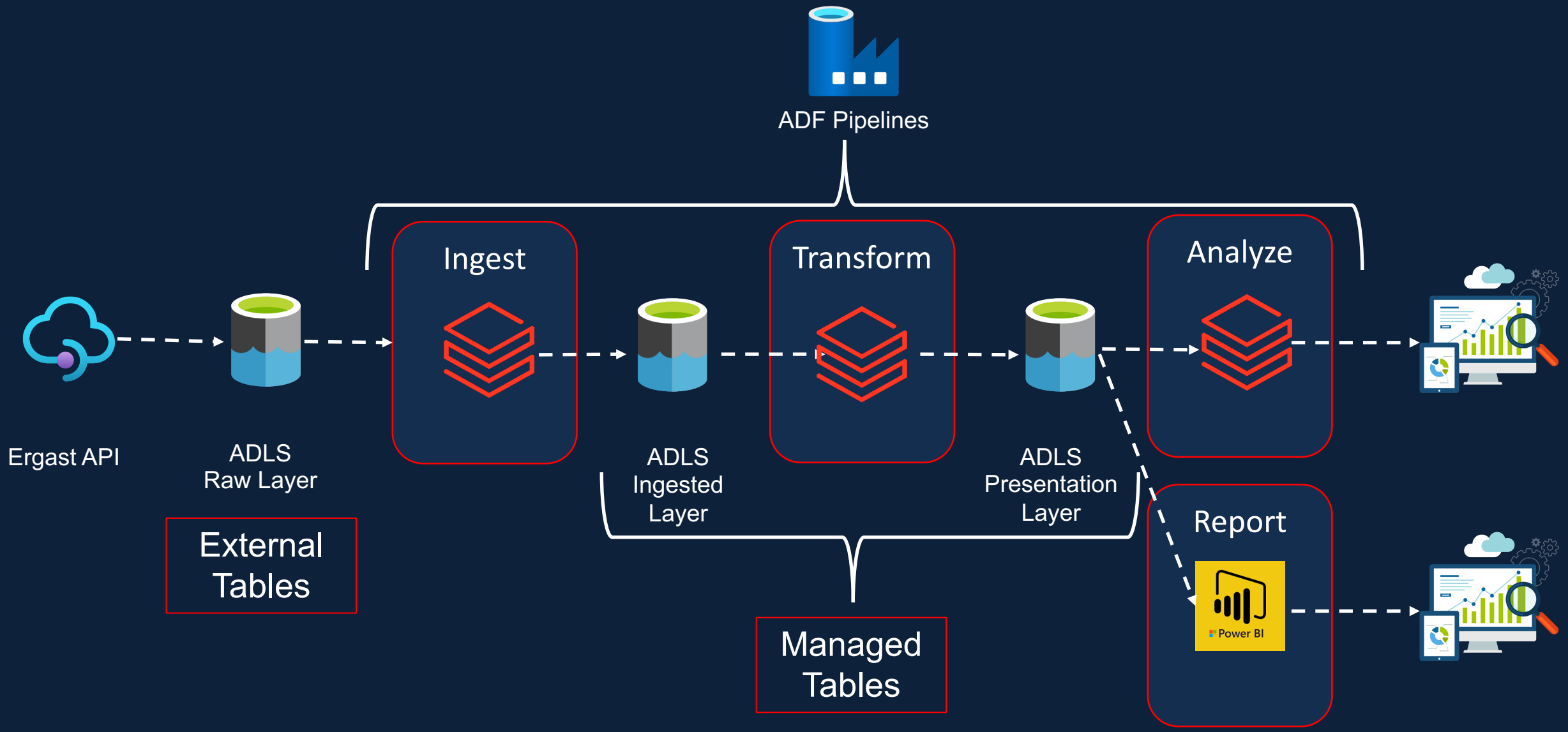


Column Name	Source
race_year	races
race_name	races
race_date	races
circuit_location	circuits
driver_name	drivers
driver_number	drivers
driver_nationality	drivers
team	constructors
grid	results
fastest_lap	results
race_time	results
points	results
created_date	current_timestamp

Spark Databases/ Tables/ Views



Managed/ External Tables



Incremental Load

Data Loading Patterns

F1 Project Load Pattern

Implementation

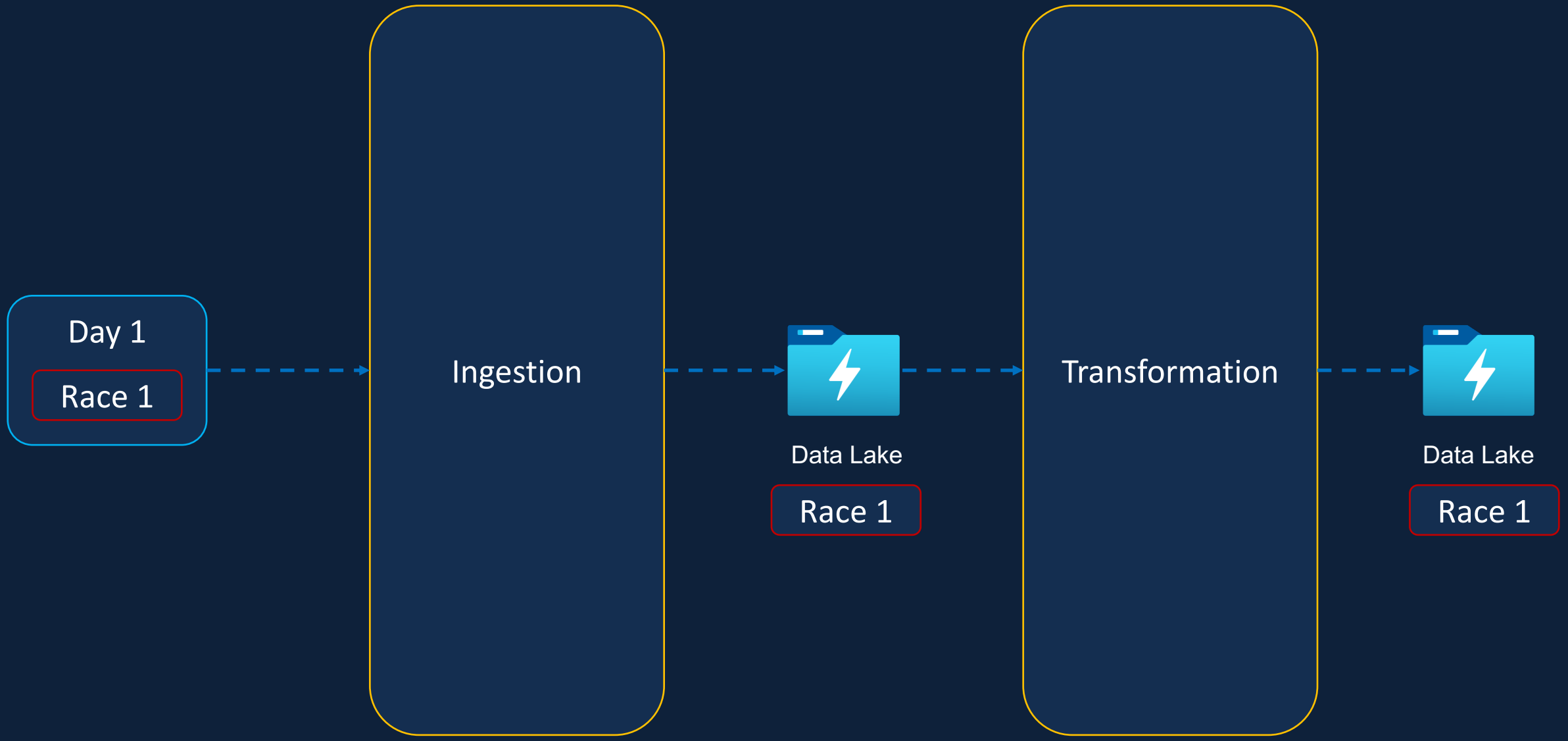
Data Load Types

Full Load

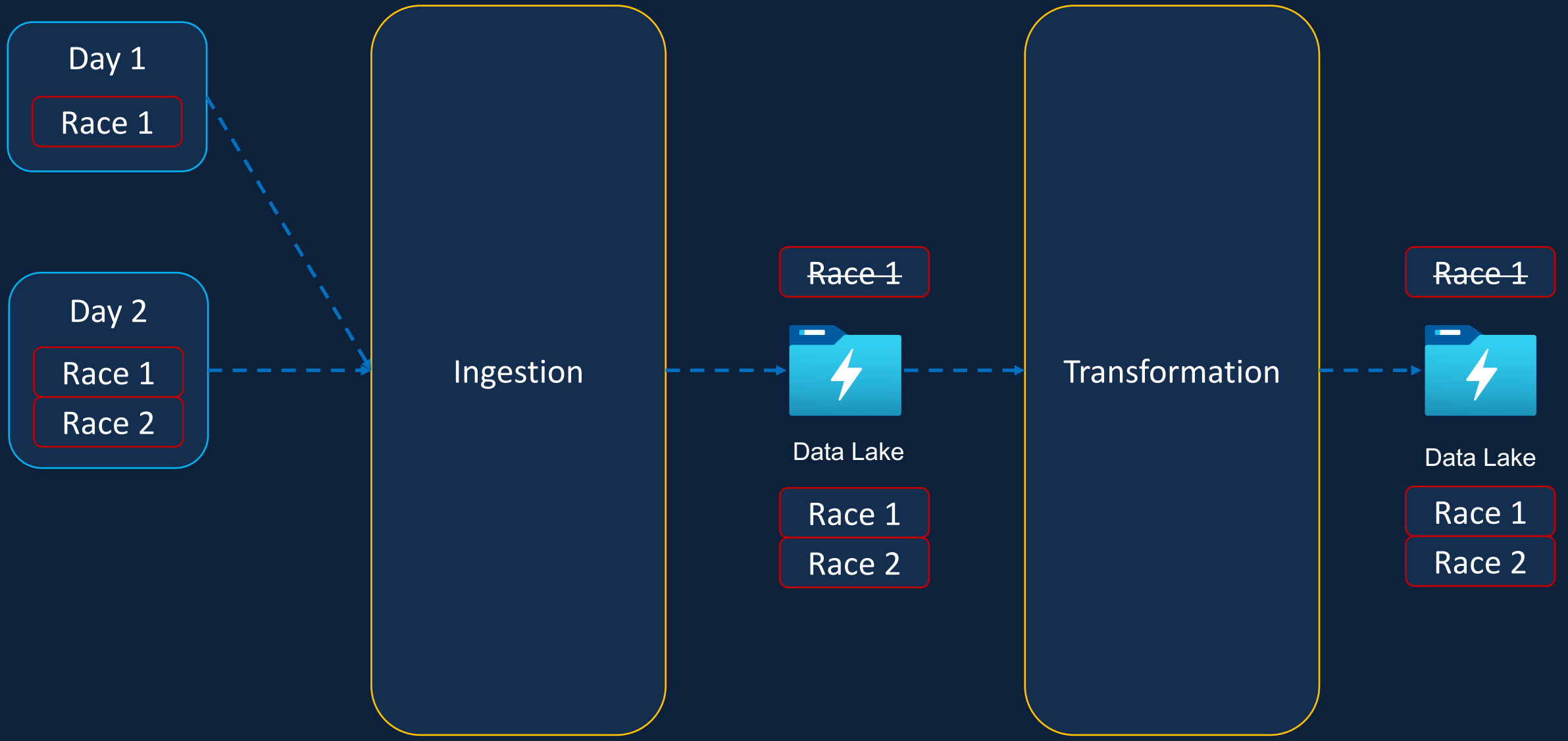
Incremental Load

Full Dataset

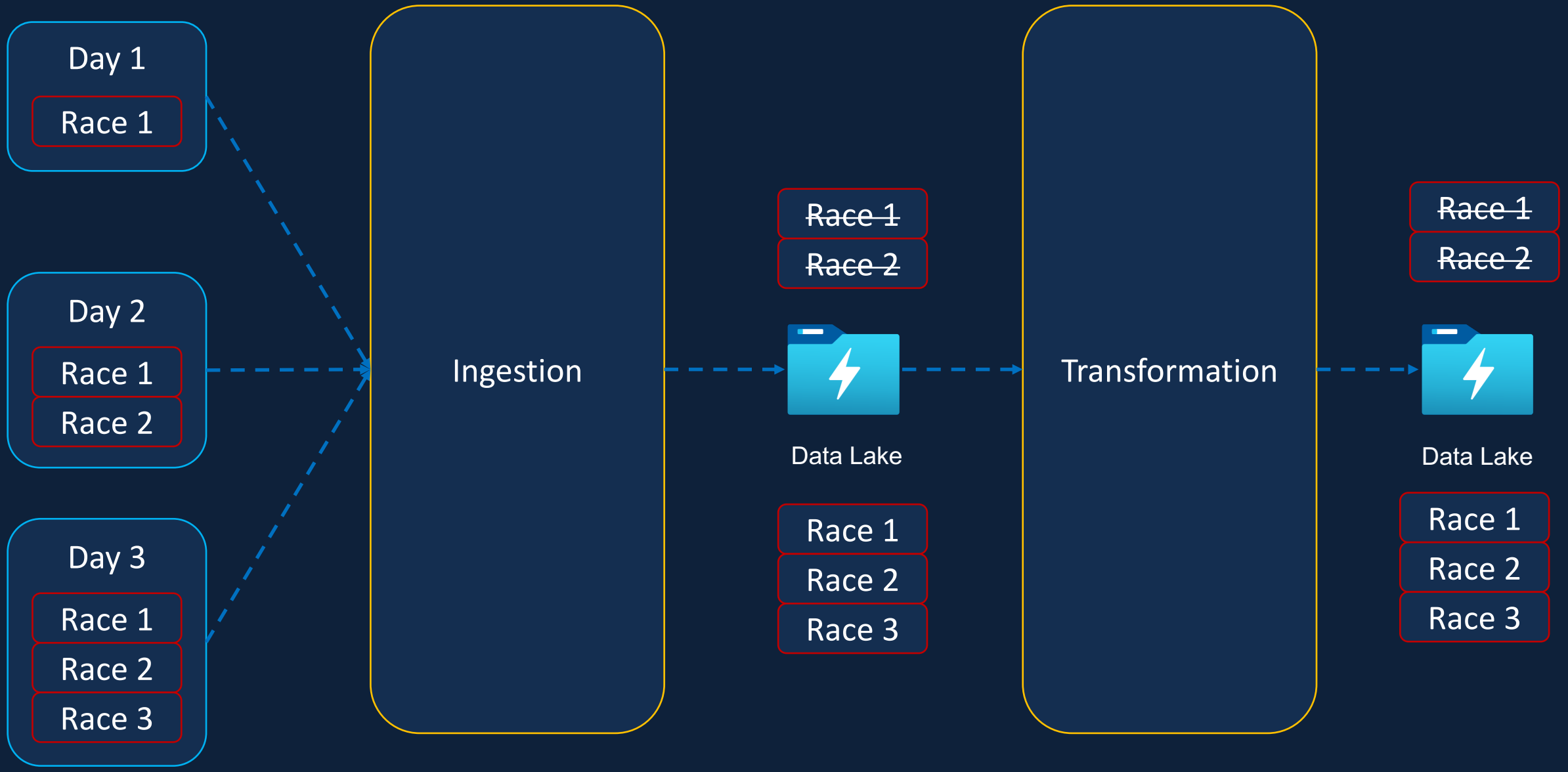
Full Refresh/ Load – Day 1



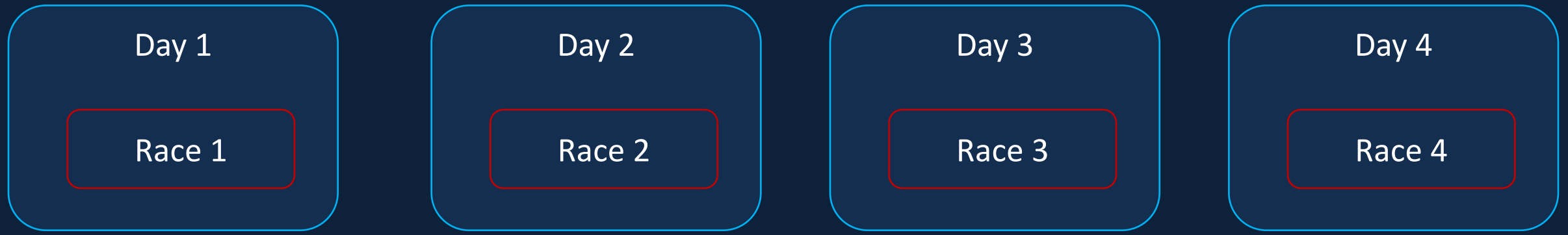
Full Refresh/ Load – Day 2



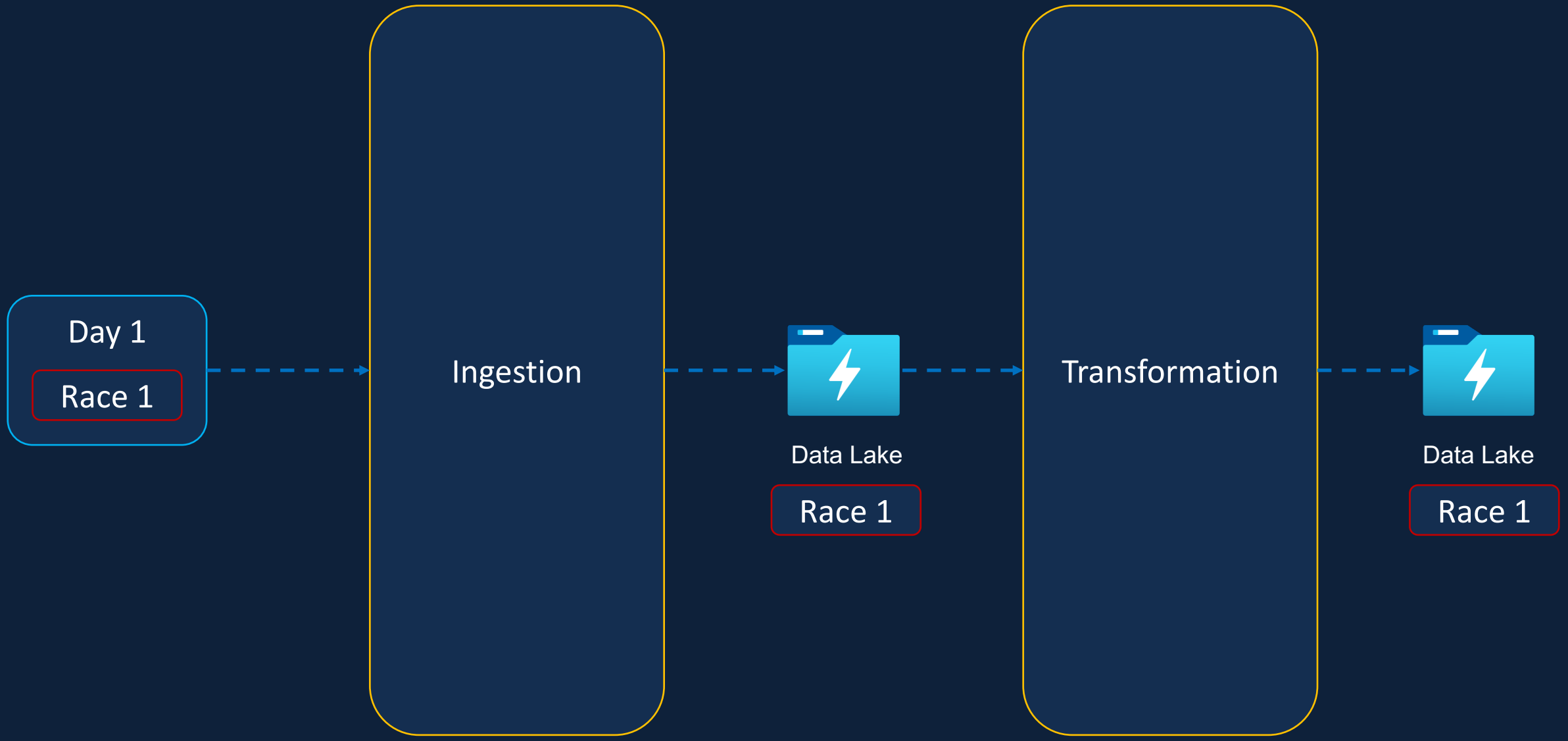
Full Refresh/ Load – Day 3



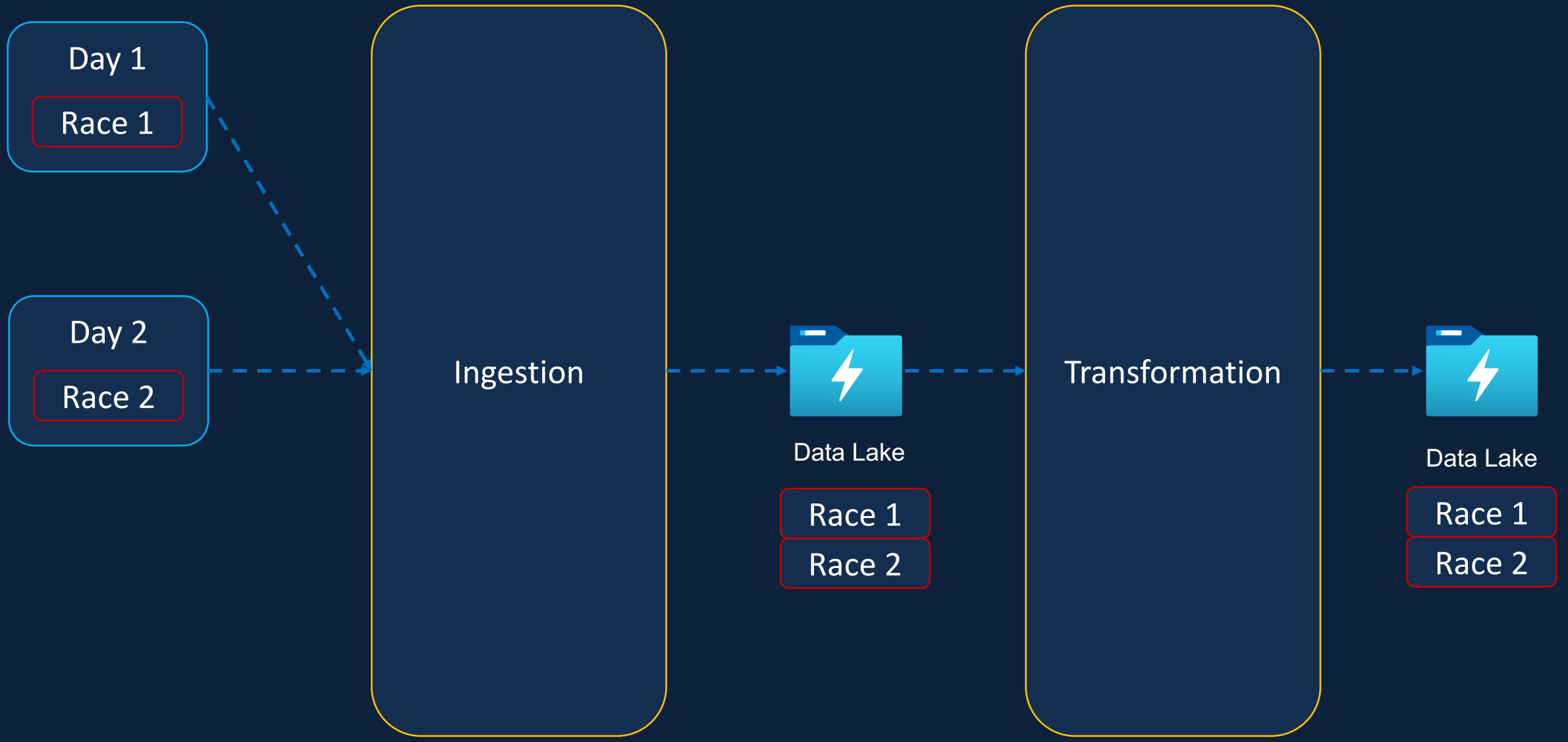
Incremental Dataset



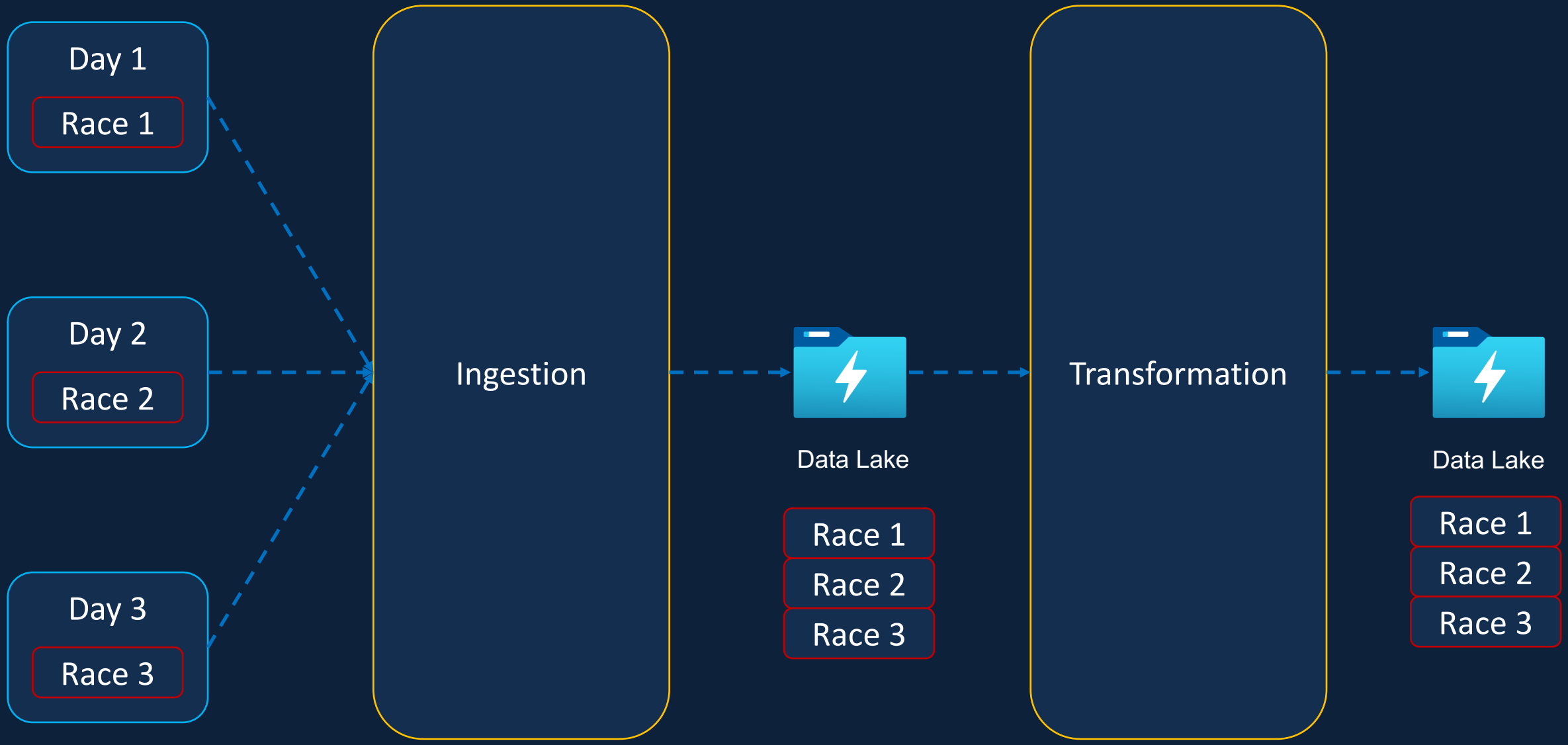
Incremental Load – Day 1



Incremental Load – Day 2



Incremental Load – Day 3



Hybrid Scenarios

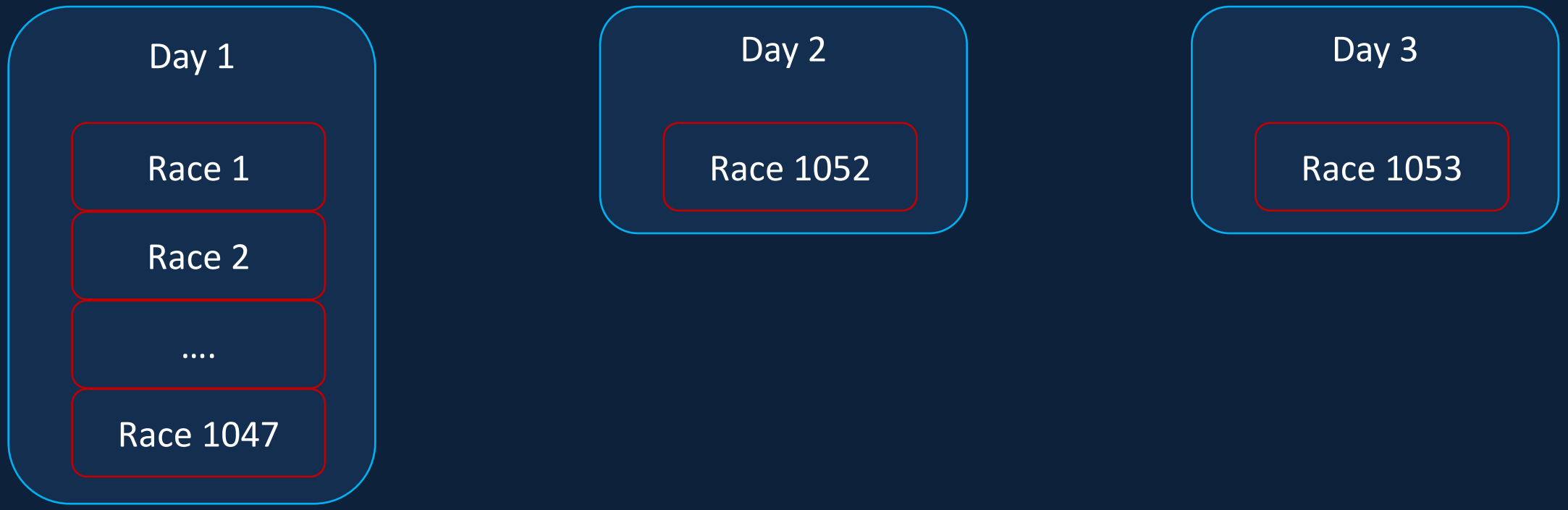
Full Dataset received, but data loaded & transformed incrementally

Incremental dataset received, but data loaded & transformed in full

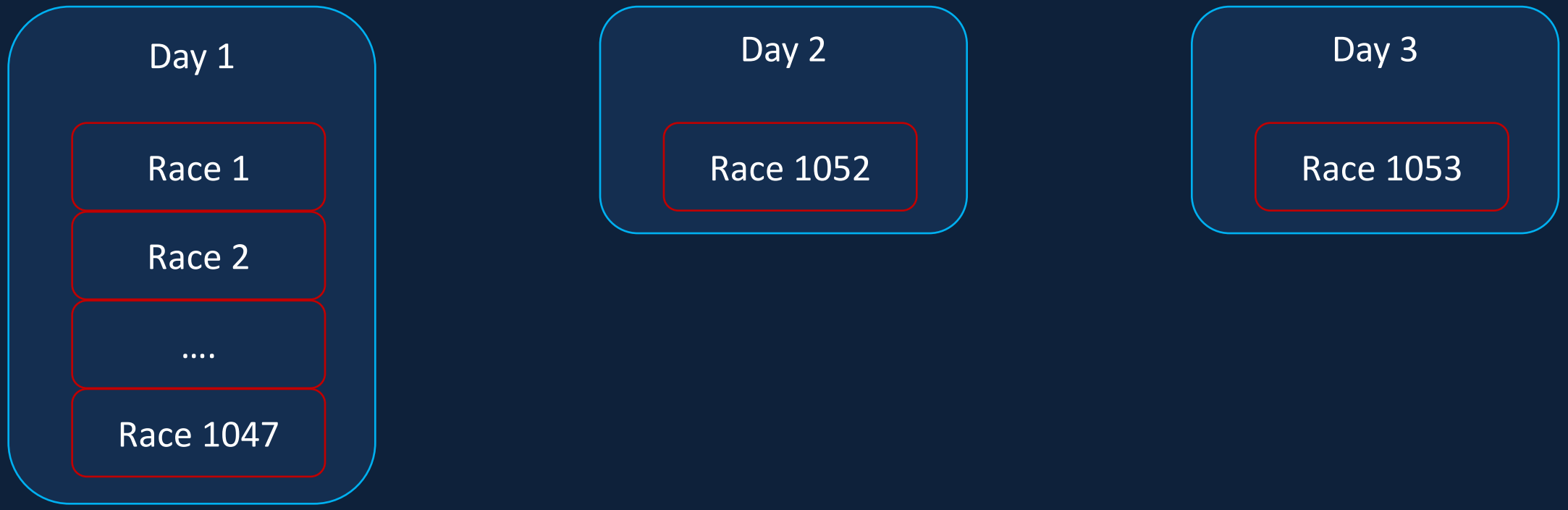
Data received contains both full and incremental files

Incremental data received. Ingested incrementally & transformed in full

Formula1 Scenario



Formula1 Scenario / Solution 1

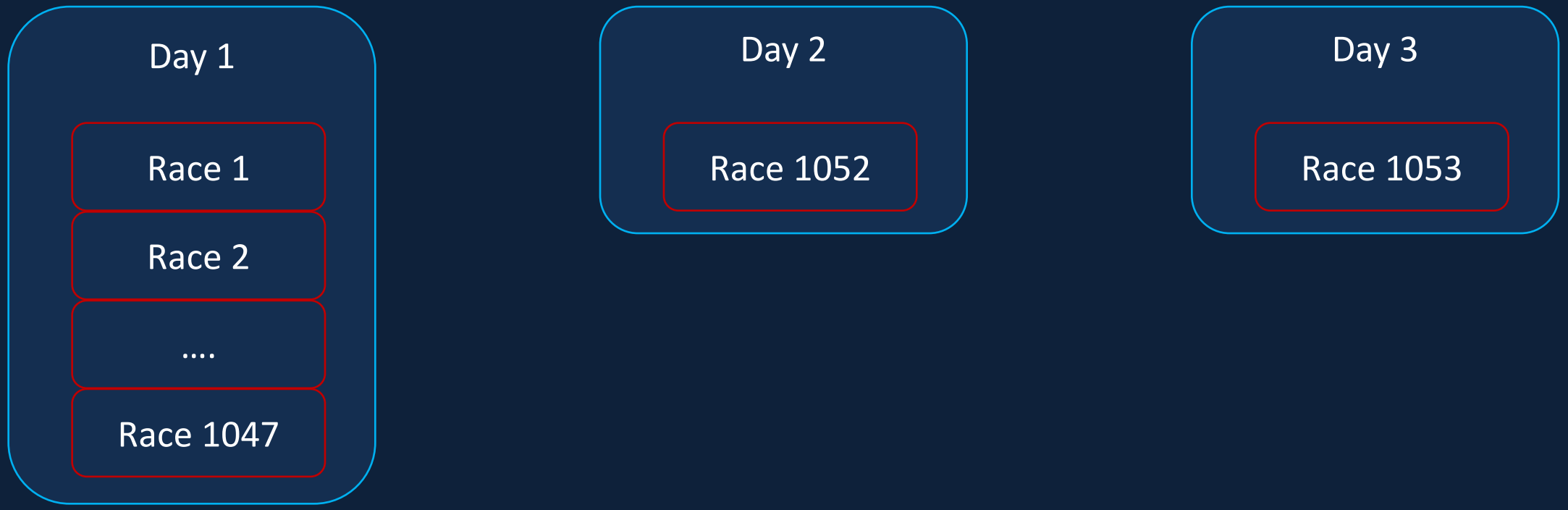


Full Refresh

Incremental Load

Incremental Load

Formula1 Scenario / Solution 2



Incremental Load

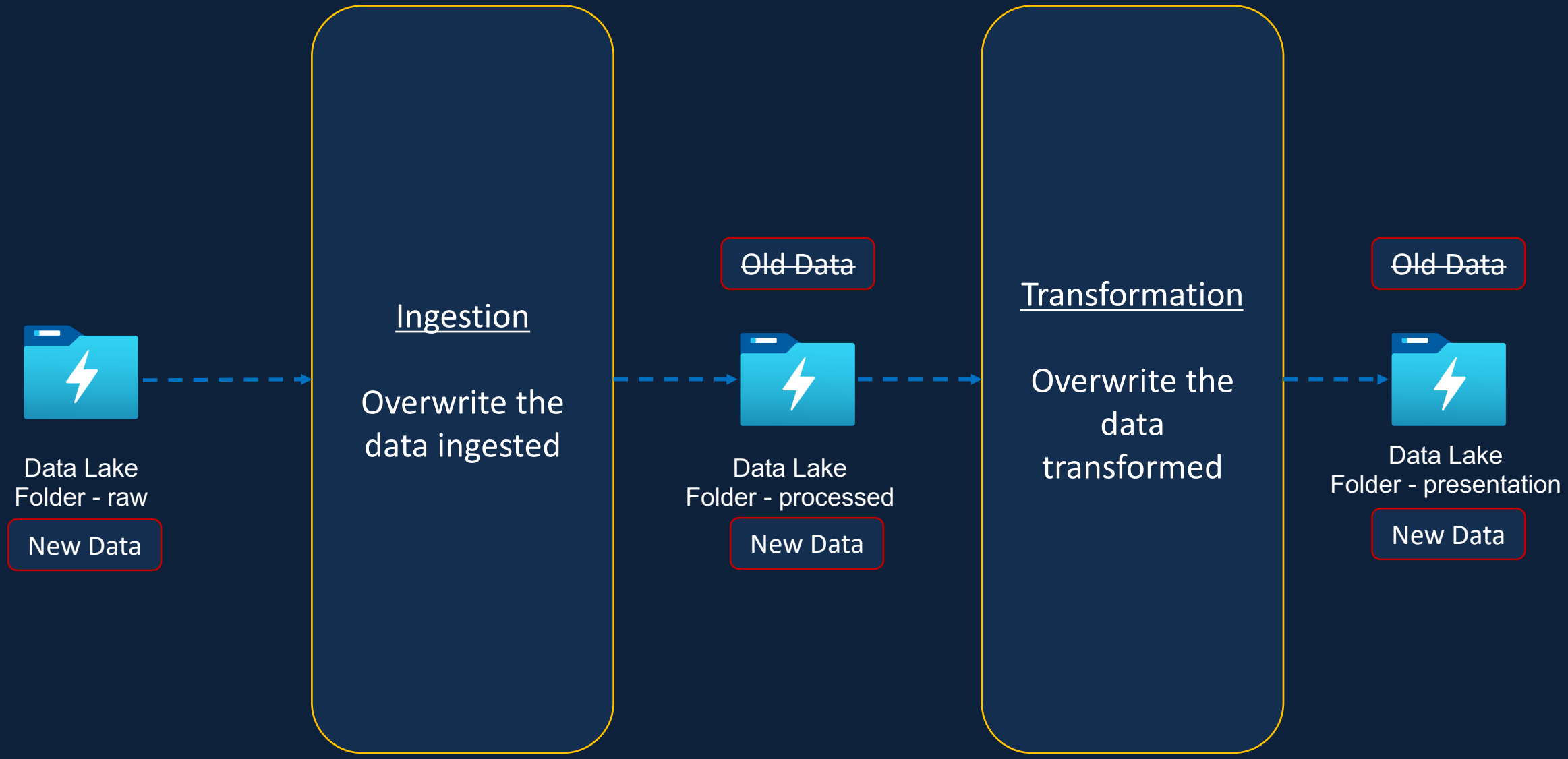
Incremental Load

Incremental Load

Formula1 Data Files

Circuits	Data from all races
Races	Data from all races
Constructors	Data from all races
Drivers	Data from all races
Results	Data only from that race
PitStops	Data only from that race
LapTimes	Data only from that race
Qualifying	Data only from that race

Current Solution



New Solution

Data Lake
Folder - raw

Sub folders

race_date_1	Races 1-1047
race_date_2	Race 1052
race_date_3	Race 1053

Ingestion

1) Process the data from a particular sub folder
2) Delete the races for we have received data from ingested
3) Load the new data received

Data Lake
Folder - processed

Transformation

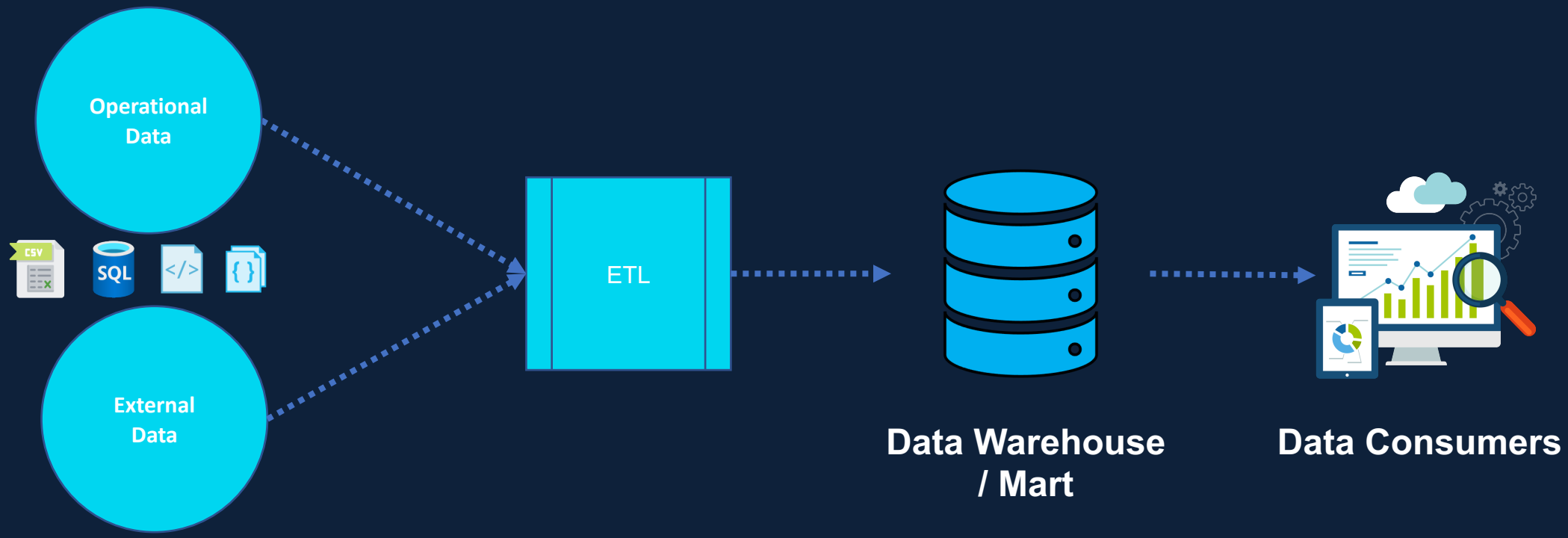
1) Identify the dates for which we have received new data
2) Delete the correspondin g races from the presentation layer
3) Process the new data received

Data Lake
Folder - presentation

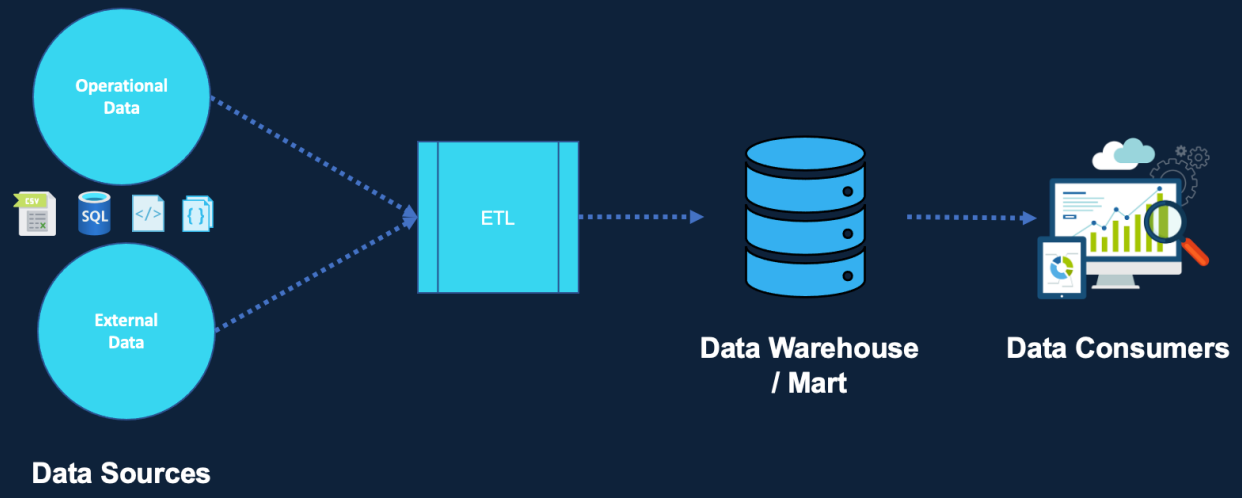
Delta Lake

- Pitfalls of Data Lakes
- Lakehouse Architecture
- Delta Lake Capabilities
- Convert F1 project to Delta Lake

Data Warehouse



Data Warehouse



Lack of support for unstructured data

Longer to ingest new data

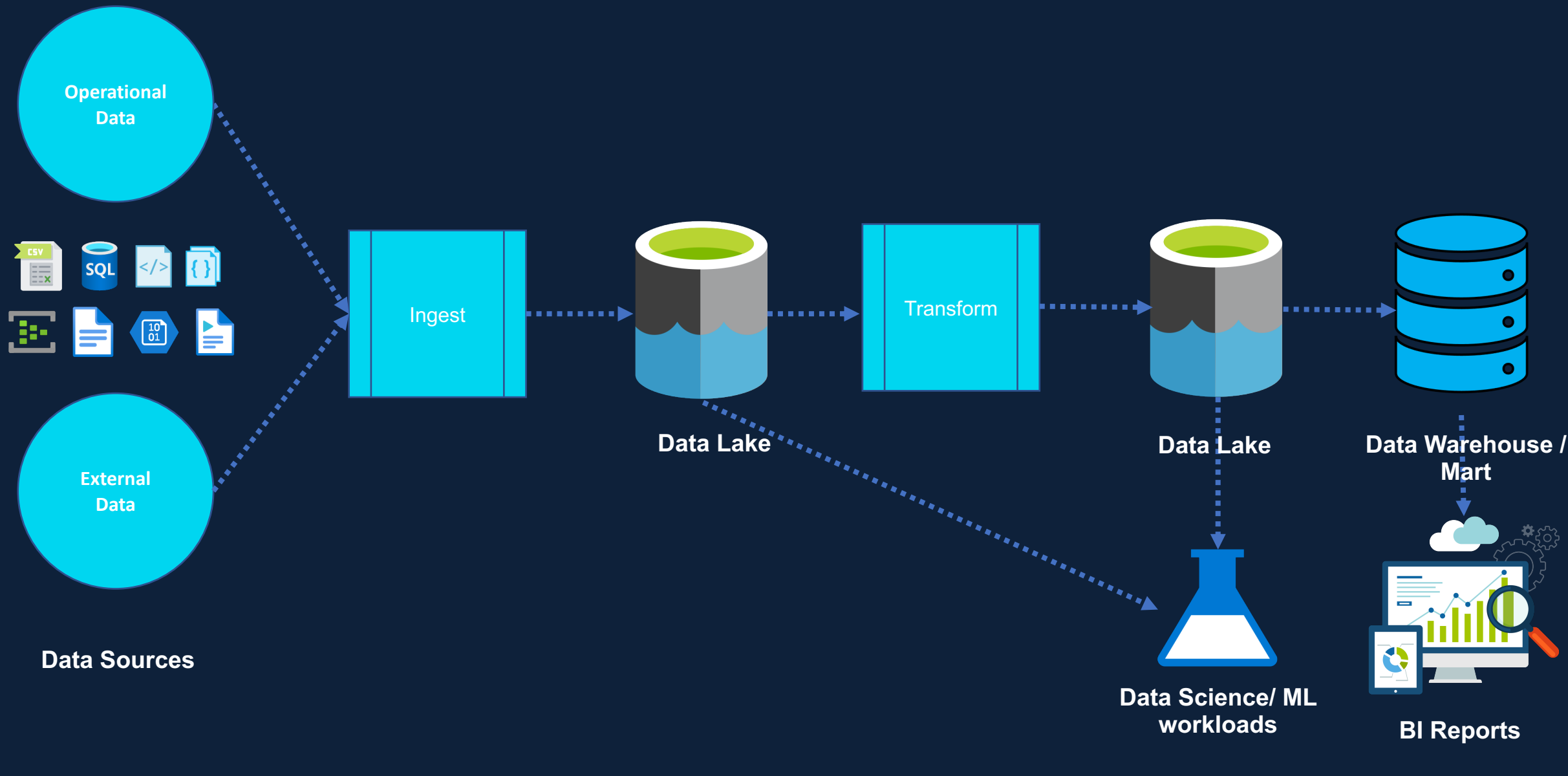
Proprietary data formats

Scalability

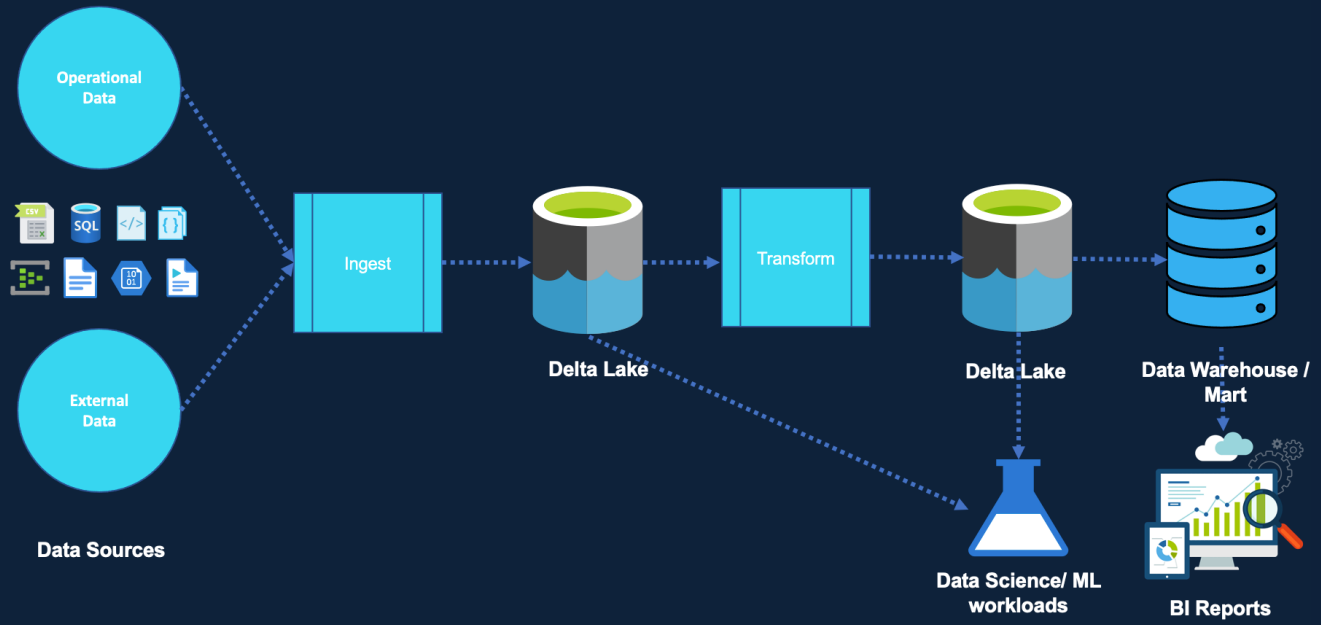
Expensive to store data

Lack of support for ML/ AI workloads

Data Lake


Operational Data
CSV
SQL
External Data
Data Sources
Ingest
Data Lake
Transform
Data Lake
Data Warehouse / Mart
Data Science/ ML workloads
BI Reports

Data Lake



No support for ACID transactions

Failed jobs leave partial files

Inconsistent reads

Unable to handle corrections to data

Unable to roll back any data.

Lack of ability remove data for GDPR etc

No history or versioning

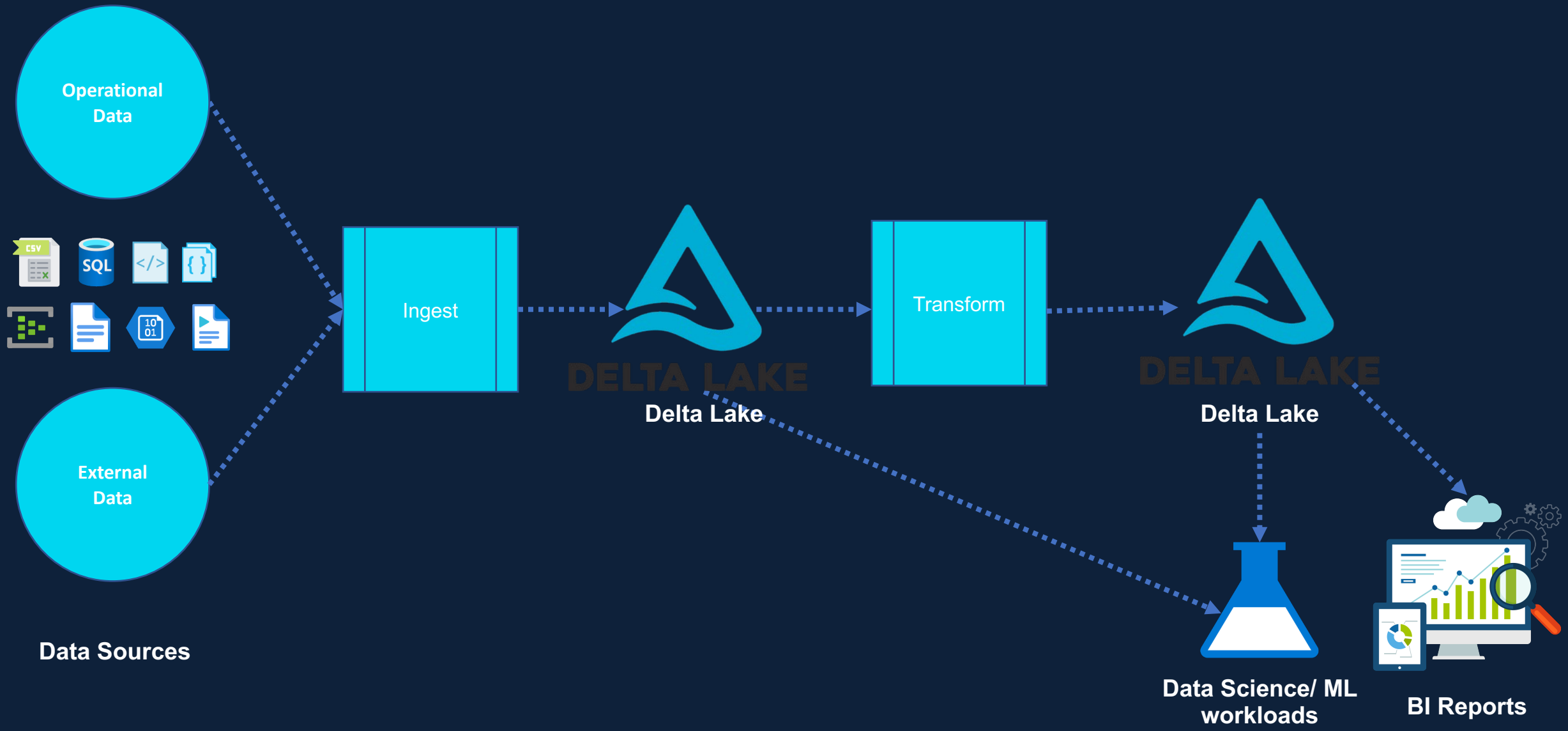
Poor performance

Poor BI support

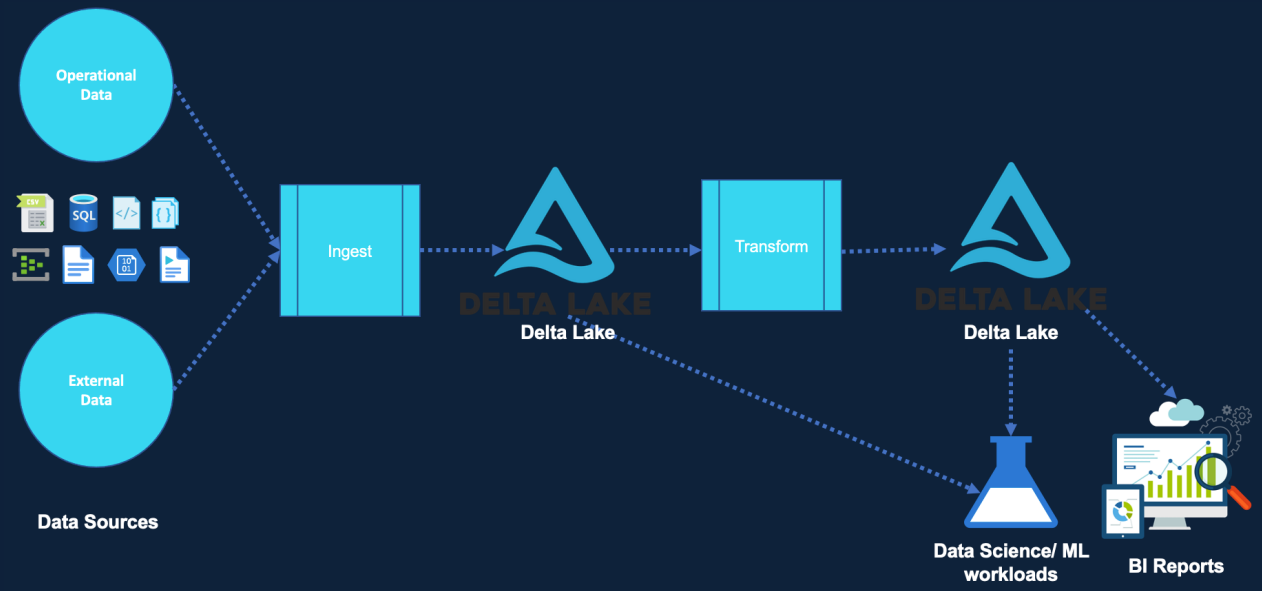
Complex to set-up

Lambda architecture for streaming workloads

Data Lakehouse

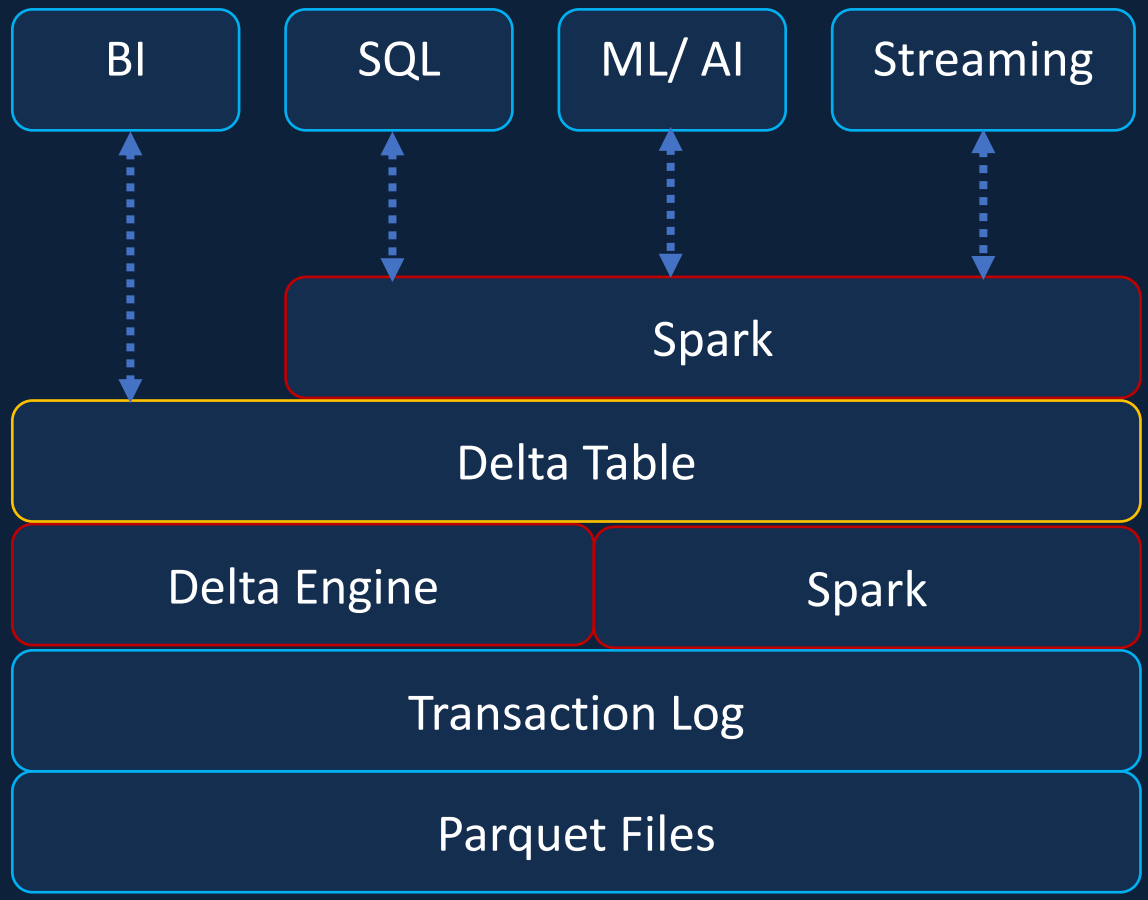


Data Lakehouse



- Handles all types of data
- Cheap cloud object storage
- Uses open source format
- Support for all types of workloads
- Ability to use BI tools directly
- ACID support
- History & Versioning
- Better performance
- Simple architecture

Delta Lake



Simple Architecture

Transformations, ML, Streaming etc.

Data security, Governance, Integrity, Perf.

Optimized engine for performance

History, Versioning, ACID, Time Travel

Open file format

Azure Data Factory

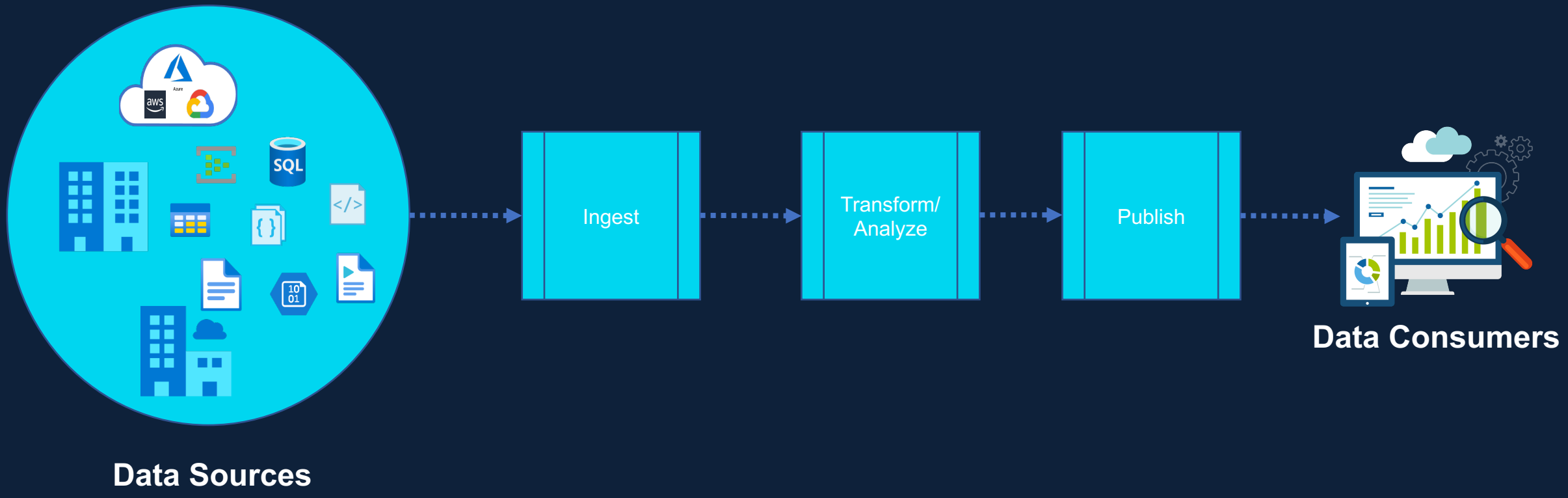
- Overview
- Creating Data Factory Service
- Data Factory Components
- Creating Pipelines
- Creating Triggers

What is Azure Data Factory

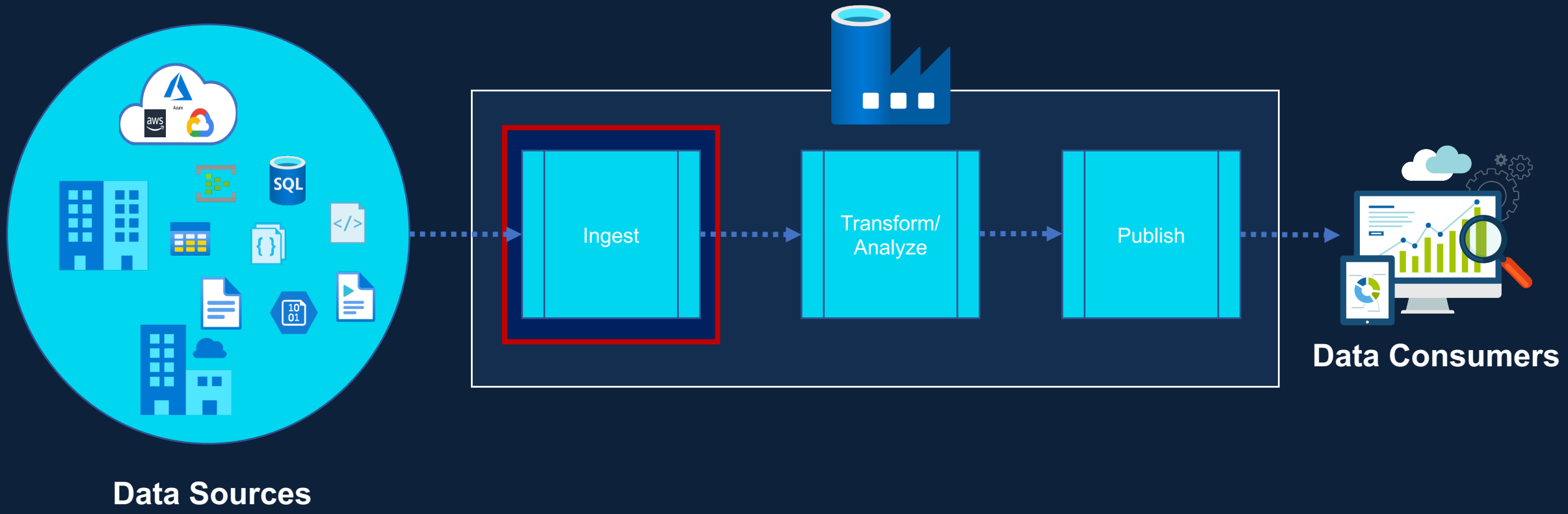
A fully managed, serverless data integration solution for ingesting, preparing and transforming all of your data at scale.

The Data Problem

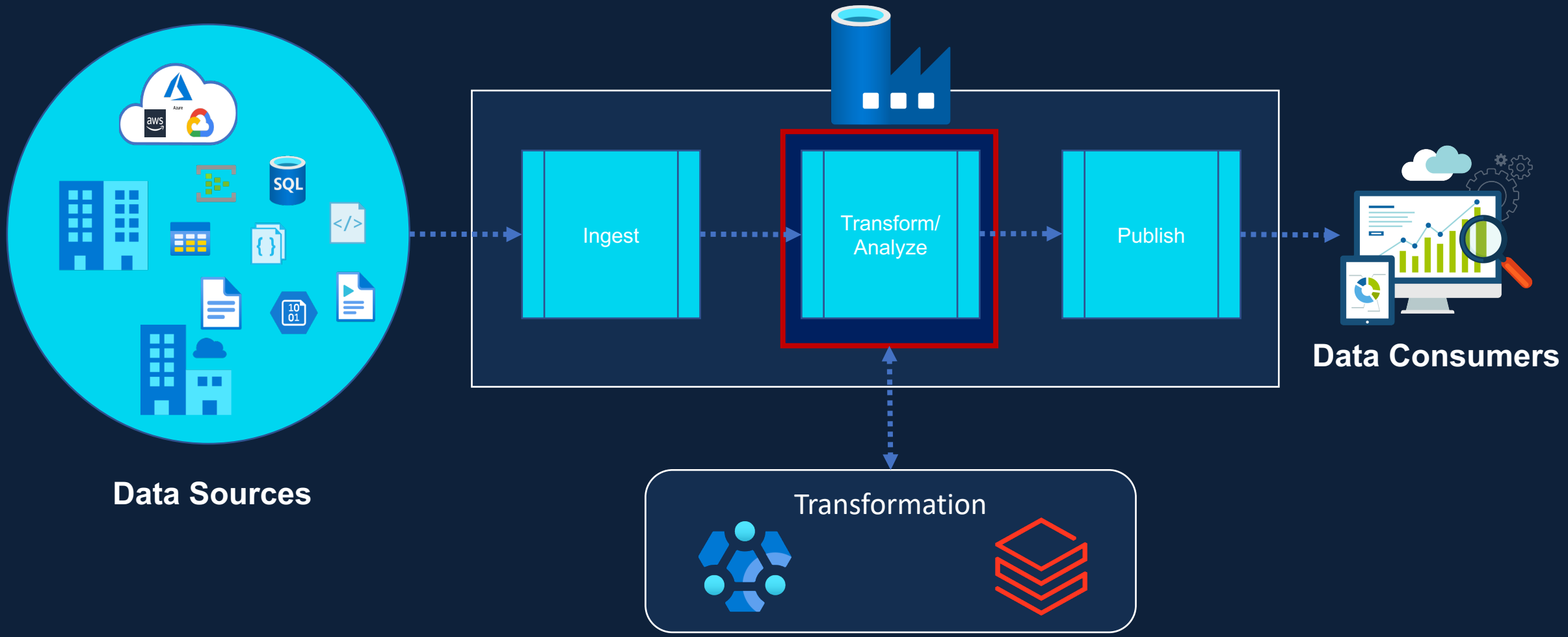
The Data Problem



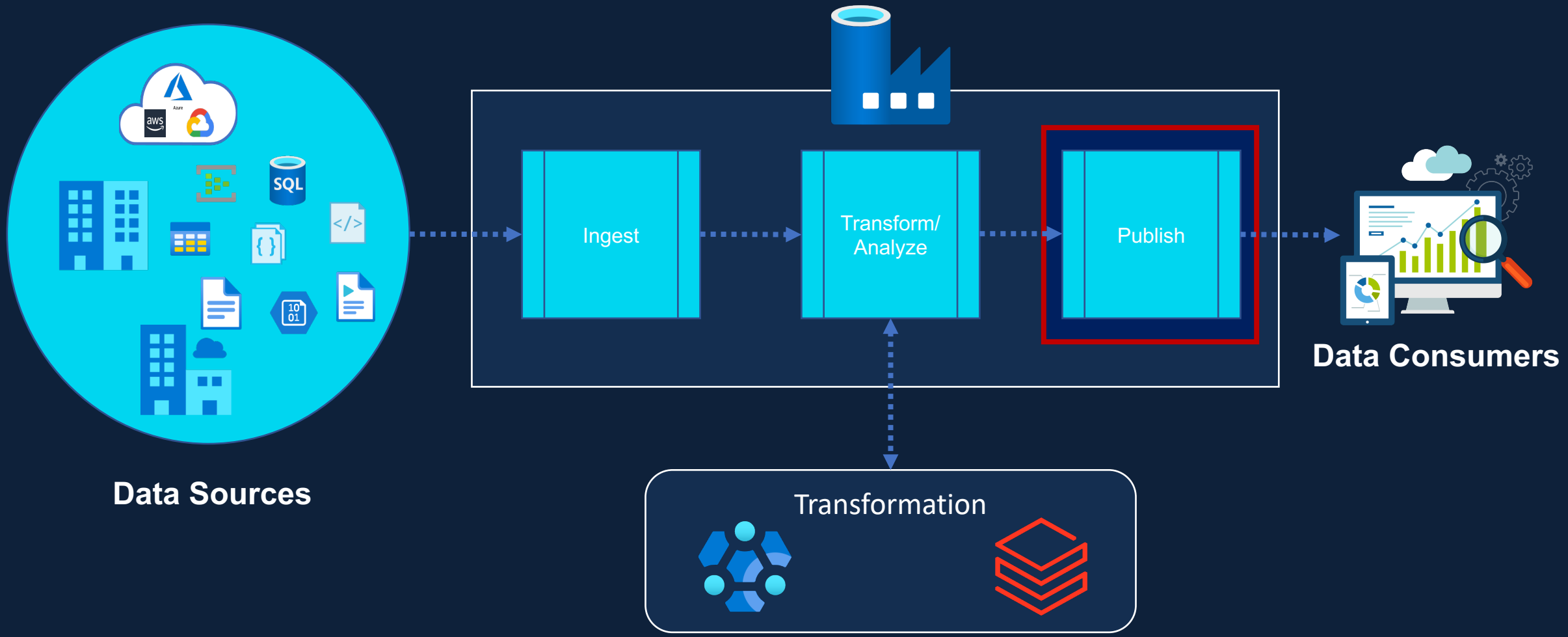
The Data Problem



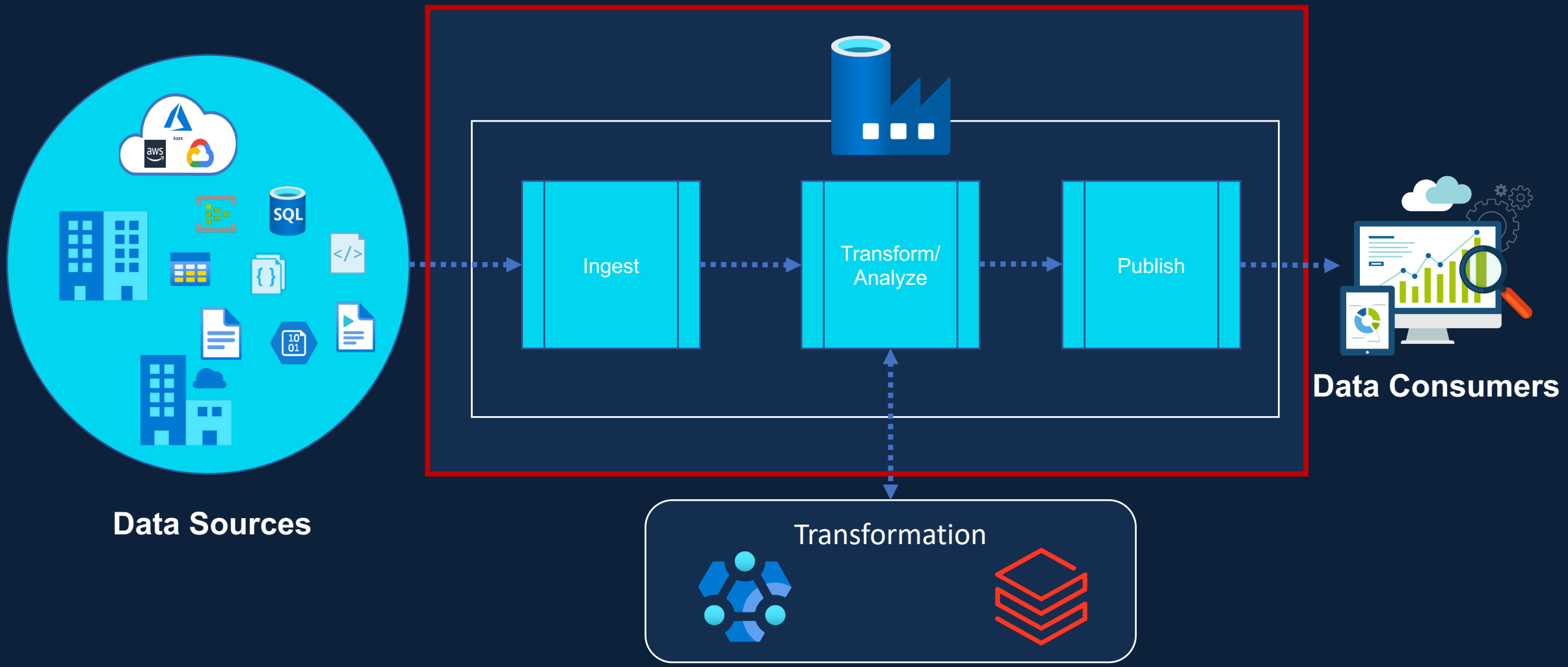
The Data Problem



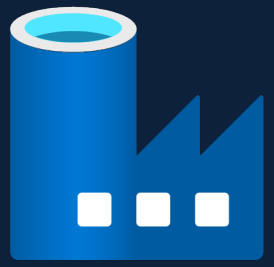
The Data Problem



The Data Problem



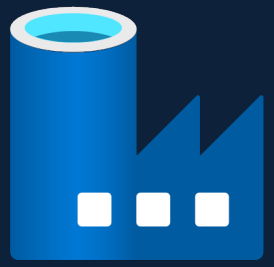
What is Azure Data Factory



- Fully Managed Service
- Serverless
- Data Integration Service
- Data Transformation Service
- Data Orchestration Service

A fully managed, serverless data integration solution for ingesting, preparing and transforming all of your data at scale.

What Azure Data Factory Is Not



- Data Migration Tool
- Data Streaming Service
- Suitable for Complex Data Transformations
- Data Storage Service

Connecting from Power BI