# ****Data Transformation Process Using Apache Spark and Hive****

## ****Introduction****

This presentation explains how raw data from the Hive table tfl\_undergroundrecord is transformed and inserted into the target Hive table tfl\_underground\_result\_n. The process involves data cleaning, adding metadata, generating unique IDs, and ensuring data consistency.

## ****Process Overview****

The transformation process follows these key steps:

1. **Load Data from Source Table**

* Read raw data from tfl\_undergroundrecord.

1. **Data Cleaning and Transformation**

* Add an ingestion\_timestamp column to track when the data was processed.
* Remove quotes from route, delay\_time, and reason columns.
* Filter out NULL values from the route column.

1. **Generate Unique Record ID**

* Retrieve the maximum record\_id from tfl\_underground\_result\_n.
* Assign a unique ID using monotonically\_increasing\_id() starting from the last record ID.

1. **Rearrange Columns & Sort Data**

* Ensure the column order matches the target table.
* Sort the dataset by timedetails in descending order to prioritize recent data.

1. **Avoid Duplicates and Insert into Target Table**

* Use a left anti-join to remove duplicate records based on record\_id.
* Append only unique records to tfl\_underground\_result\_n.

1. **Write Transformed Data to Hive Table**

* Insert the final cleaned and transformed dataset into the target Hive table.
* Stop the Spark session.

**Flow Diagram**

Add ingestion\_timestamp

Remove Quotes & NULL Values

(route, delay\_time, reason)

Generate Unique Record ID

Rearrange Columns & Sort Data

Remove Duplicates via Join

Insert into Target Table

tfl\_underground\_result\_n

## ****Key Takeaways****

* **Data Quality:** Cleaning operations ensure accuracy and consistency.
* **Unique Identification:** record\_id prevents duplicate entries.
* **Efficient Storage:** Sorting and filtering optimise query performance.
* **Incremental Updates:** The left anti-join ensures only new data is inserted.
* **Scalability:** Using Apache Spark and Hive allows handling large datasets efficiently.

### ****Conclusion****

This transformation pipeline ensures that raw data is cleaned, structured, and stored efficiently in Hive for further analysis. The use of PySpark provides automation, scalability, and performance optimisation.