University of East London

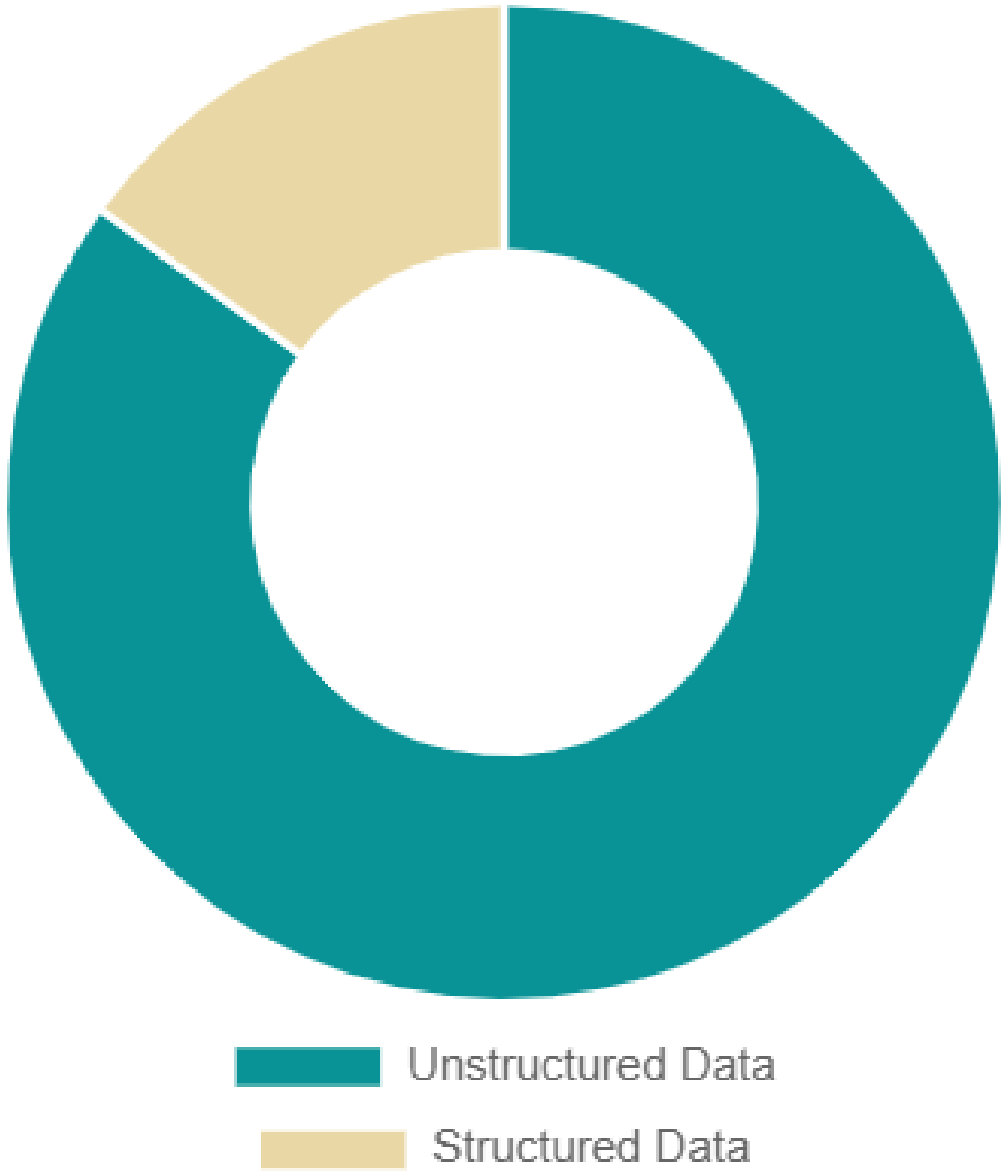
Pioneering Futures Since 1898

How Businesses Work with Unstructured Data Using Hadoop

Dr Amin Karami

Associate Professor in AI
a.karami@uel.ac.uk
www.aminkarami.com

CN7031 – Lecture
October 2025 – Week 5



Expertise: AI, Data Analytics, Learning Systems

Speaker:
Setinaz Foroudi

From Challenges to Opportunities, a Journey of Growth and Innovation

🌟 Gain insights from an **industry expert** on transitioning from academia to business

📅 Wednesday (19 Nov): MLT - 16:45-18:00

📅 Thursday (20 Nov): WB.G.02 - 16:45-18:00

University of East London
Pioneering Futures since 1898

Outline

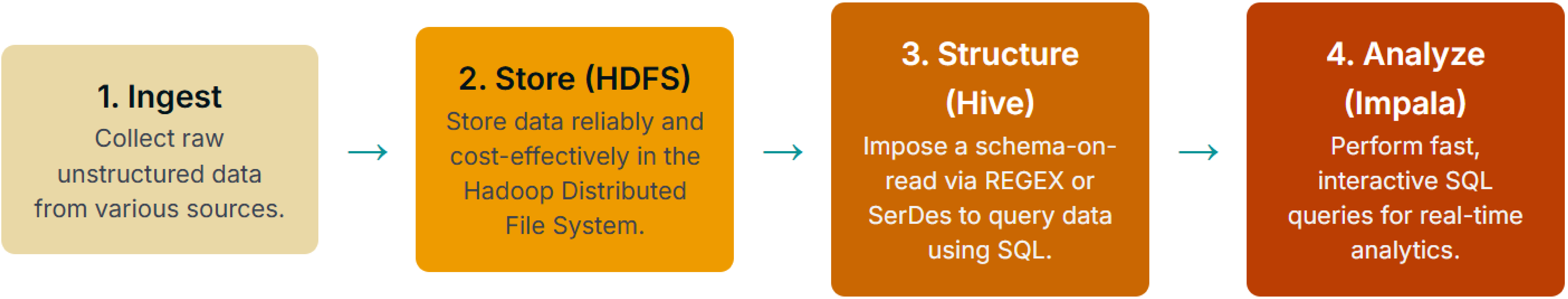
- The Hadoop Workflow for Unstructured Data
- Schema-on-Read and Schema-on-Write
- How Hive Reads Data: The "SerDe"
- Advanced Hive Optimizations

Unstructured Data

Data that does not fit into a traditional row-column database. It is complex and comes in various formats:

- Social Media posts
- Log Files
- Images & Videos
- Sensor Data
- Emails

The Hadoop Ecosystem Workflow



1. Ingest
Collect raw unstructured data from various sources.

→

2. Store (HDFS)
Store data reliably and cost-effectively in the Hadoop Distributed File System.

→

3. Structure (Hive)
Impose a schema-on-read via REGEX or SerDes to query data using SQL.

→

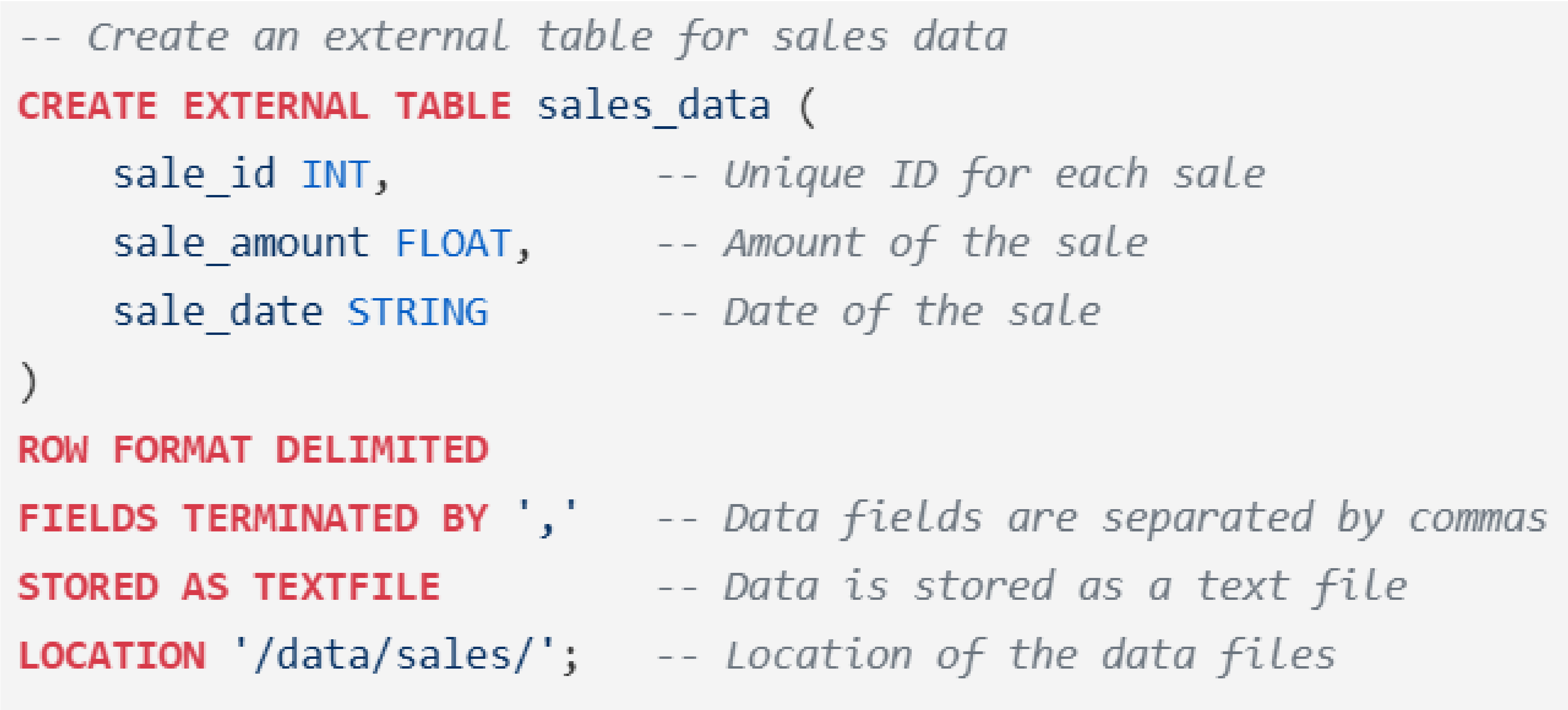
4. Analyze (Impala)
Perform fast, interactive SQL queries for real-time analytics.

Imposing Structure with Hive:

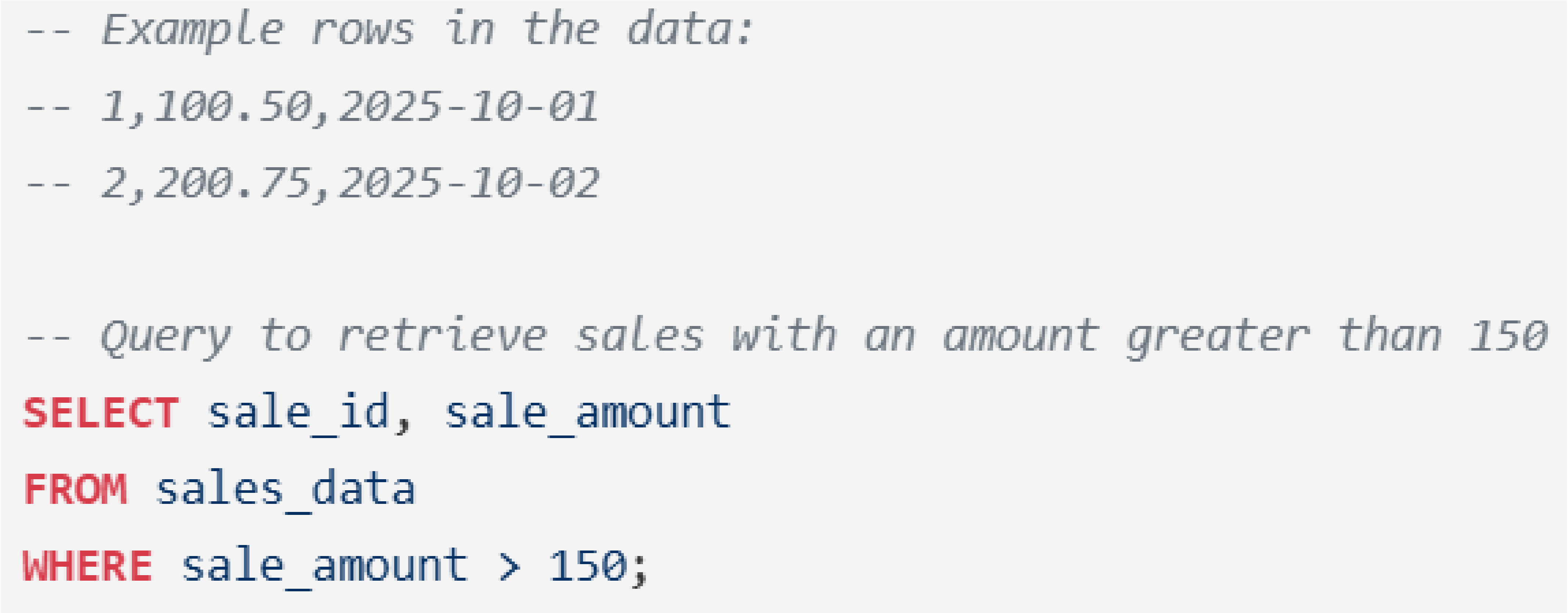
Hive's "schema-on-read" is the key to querying unstructured data with familiar SQL syntax.

Schema-on-read

- The schema is applied to the data only **at the time of query execution** (without upfront schema enforcement), allowing unstructured or semi-structured data, suitable for Big data systems and data lakes (e.g., Hive).



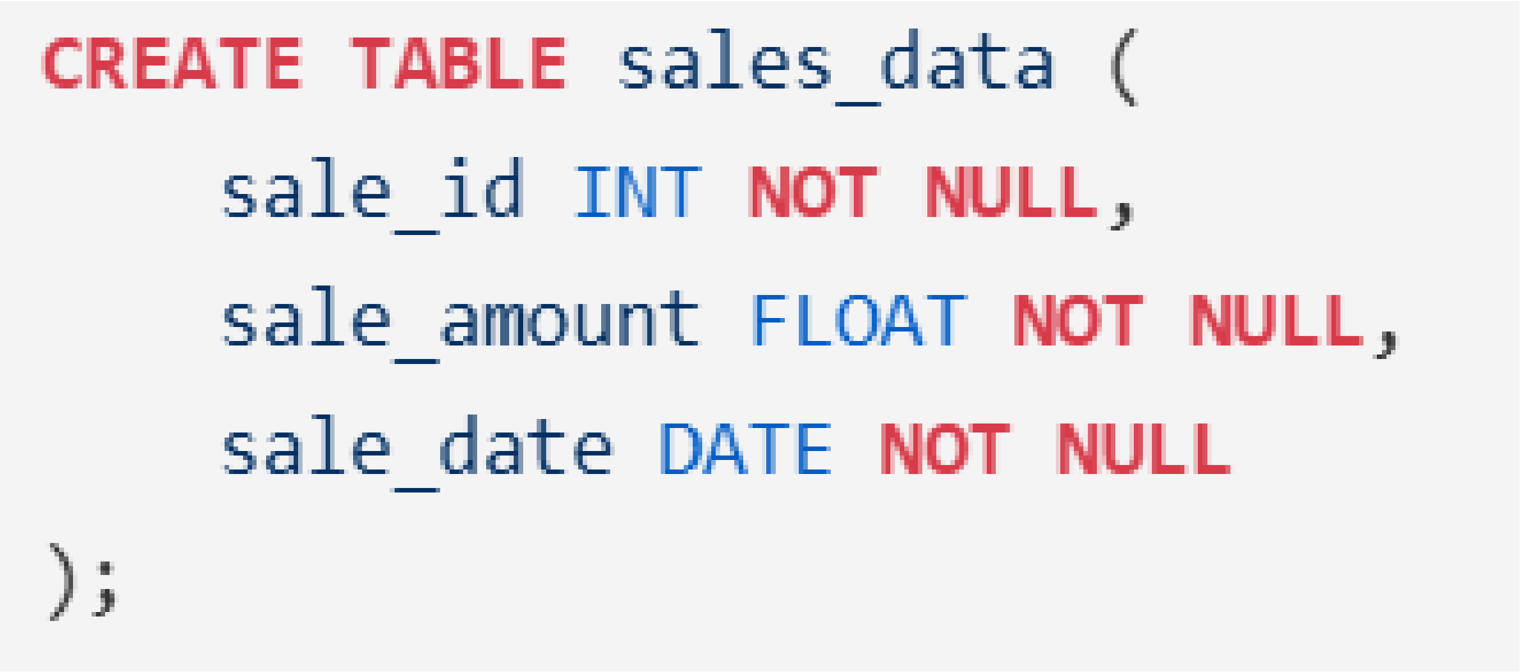
```
-- Create an external table for sales data
CREATE EXTERNAL TABLE sales_data (
    sale_id INT,            -- Unique ID for each sale
    sale_amount FLOAT,      -- Amount of the sale
    sale_date STRING        -- Date of the sale
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','    -- Data fields are separated by commas
STORED AS TEXTFILE          -- Data is stored as a text file
LOCATION '/data/sales/';    -- Location of the data files
```



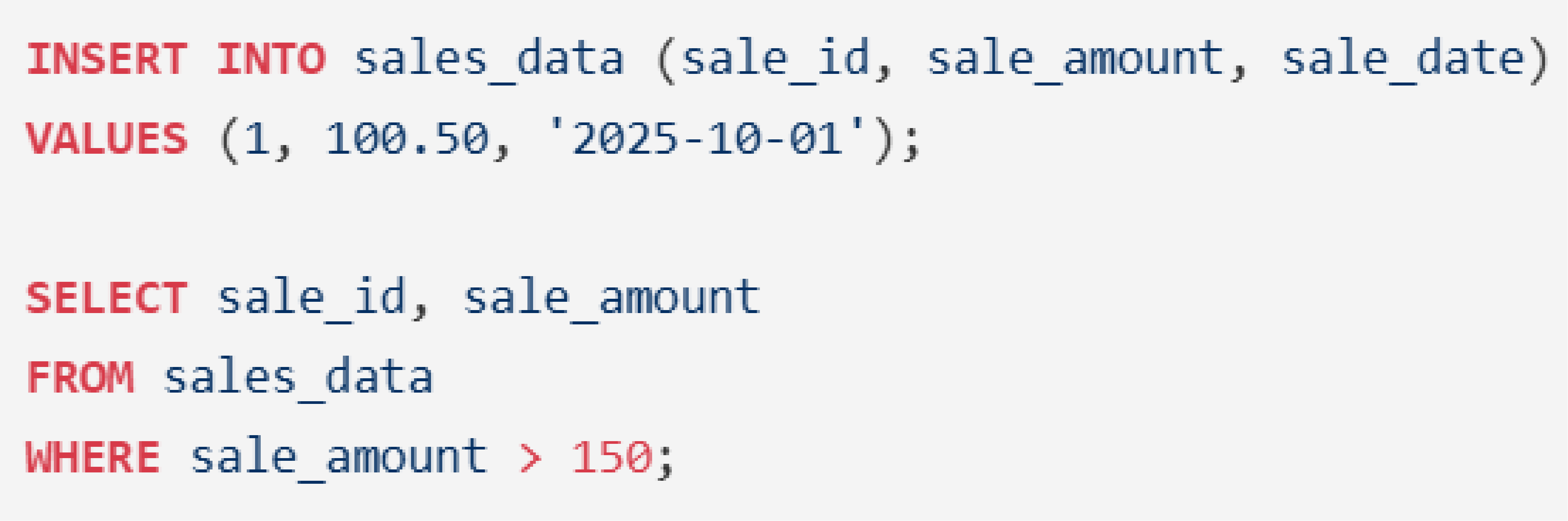
```
-- Example rows in the data:
-- 1,100.50,2025-10-01
-- 2,200.75,2025-10-02

-- Query to retrieve sales with an amount greater than 150
SELECT sale_id, sale_amount
FROM sales_data
WHERE sale_amount > 150;
```

Schema-on-write

- The schema is enforced **when data is written** to the storage system, ensuring that only data following the predefined schema is stored.
- This approach is used in traditional relational databases like MySQL or PostgreSQL, where data integrity and structure are critical.



```
CREATE TABLE sales_data (
    sale_id INT NOT NULL,
    sale_amount FLOAT NOT NULL,
    sale_date DATE NOT NULL
);
```



```
INSERT INTO sales_data (sale_id, sale_amount, sale_date)
VALUES (1, 100.50, '2025-10-01');

SELECT sale_id, sale_amount
FROM sales_data
WHERE sale_amount > 150;
```

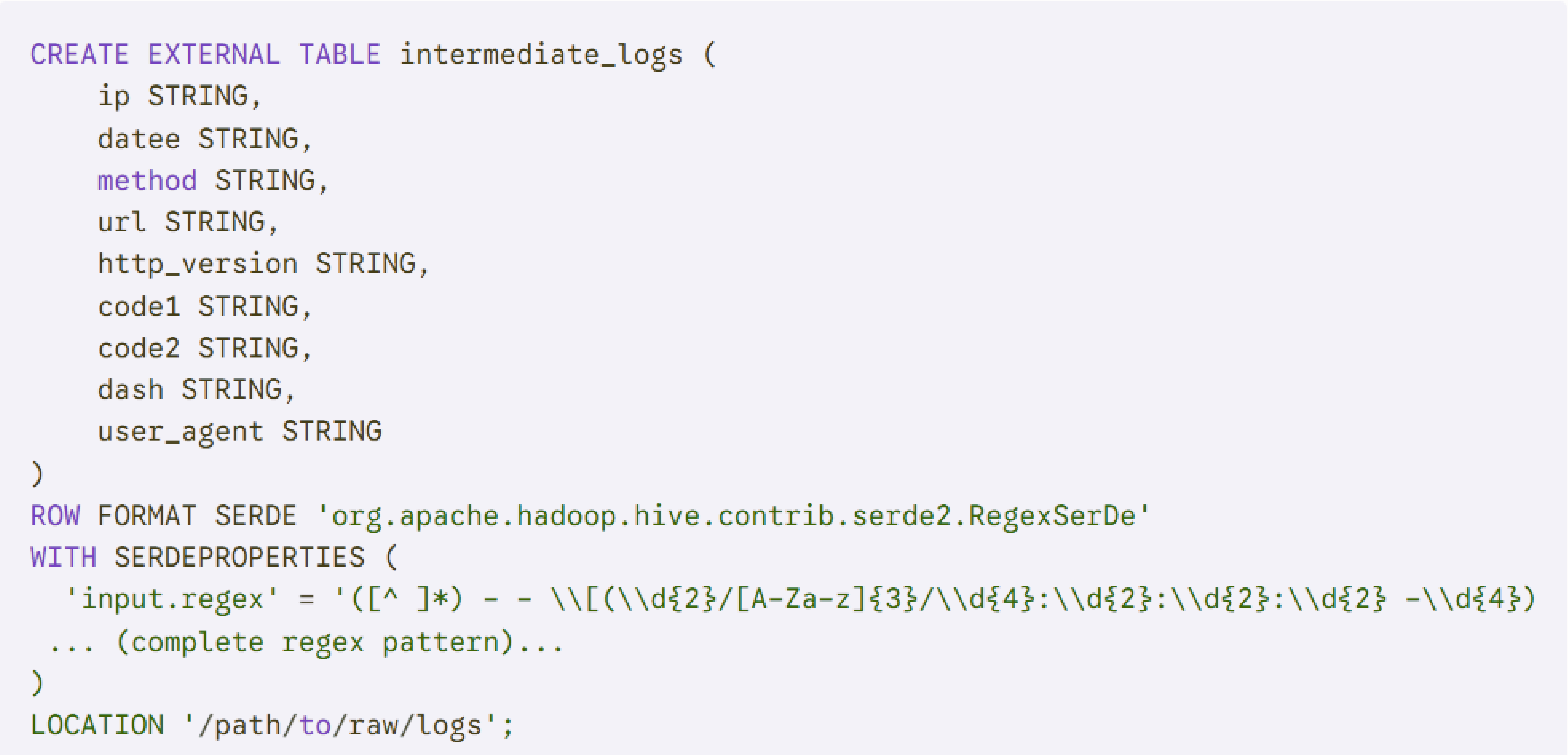
The Role of SerDe

SerDe is a core component of Hive's flexibility, enabling it to process data stored in a variety of formats. It supports the schema-on-read paradigm by dynamically interpreting data at query time.

Popular SerDe Choices:

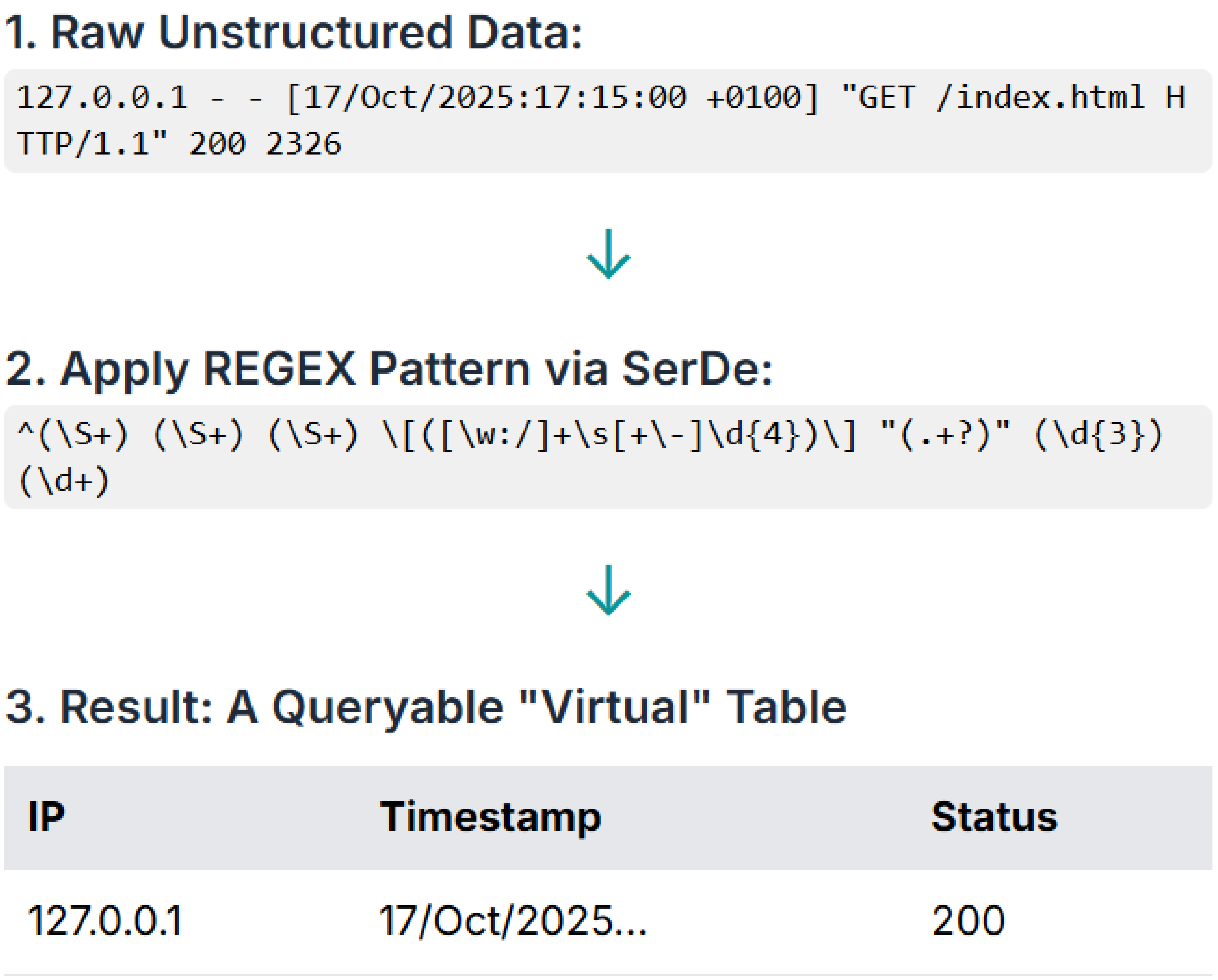
- JSON SerDe: parsing JSON files.
- Regex SerDe: applying regular expressions to unstructured text like log files.
- Avro/Parquet/ORC SerDe: efficient binary formats.
- Multimedia SerDe: multimedia files (audio, video, images)

SerDe: Example



```
CREATE EXTERNAL TABLE intermediate_logs (
    ip STRING,
    datee STRING,
    method STRING,
    url STRING,
    http_version STRING,
    code1 STRING,
    code2 STRING,
    dash STRING,
    user_agent STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  'input.regex' = '([^ ]*) - - \\[(\\d{2}/[A-Za-z]{3}/\\d{4}:\\d{2}:\\d{2}:\\d{2} -\\d{4})
 ... (complete regex pattern)...
)
LOCATION '/path/to/raw/logs';
```

REGEX in action



1. Raw Unstructured Data:

```
127.0.0.1 - - [17/Oct/2025:17:15:00 +0100] "GET /index.html H
TTP/1.1" 200 2326
```

↓

2. Apply REGEX Pattern via SerDe:

```
^(\S+) (\S+) (\S+) \[([\w:/]+\s[+\-]\d{4})\] "(.+?)" (\d{3})
(\d+)
```

↓

3. Result: A Queryable "Virtual" Table

IP	Timestamp	Status
127.0.0.1	17/Oct/2025...	200

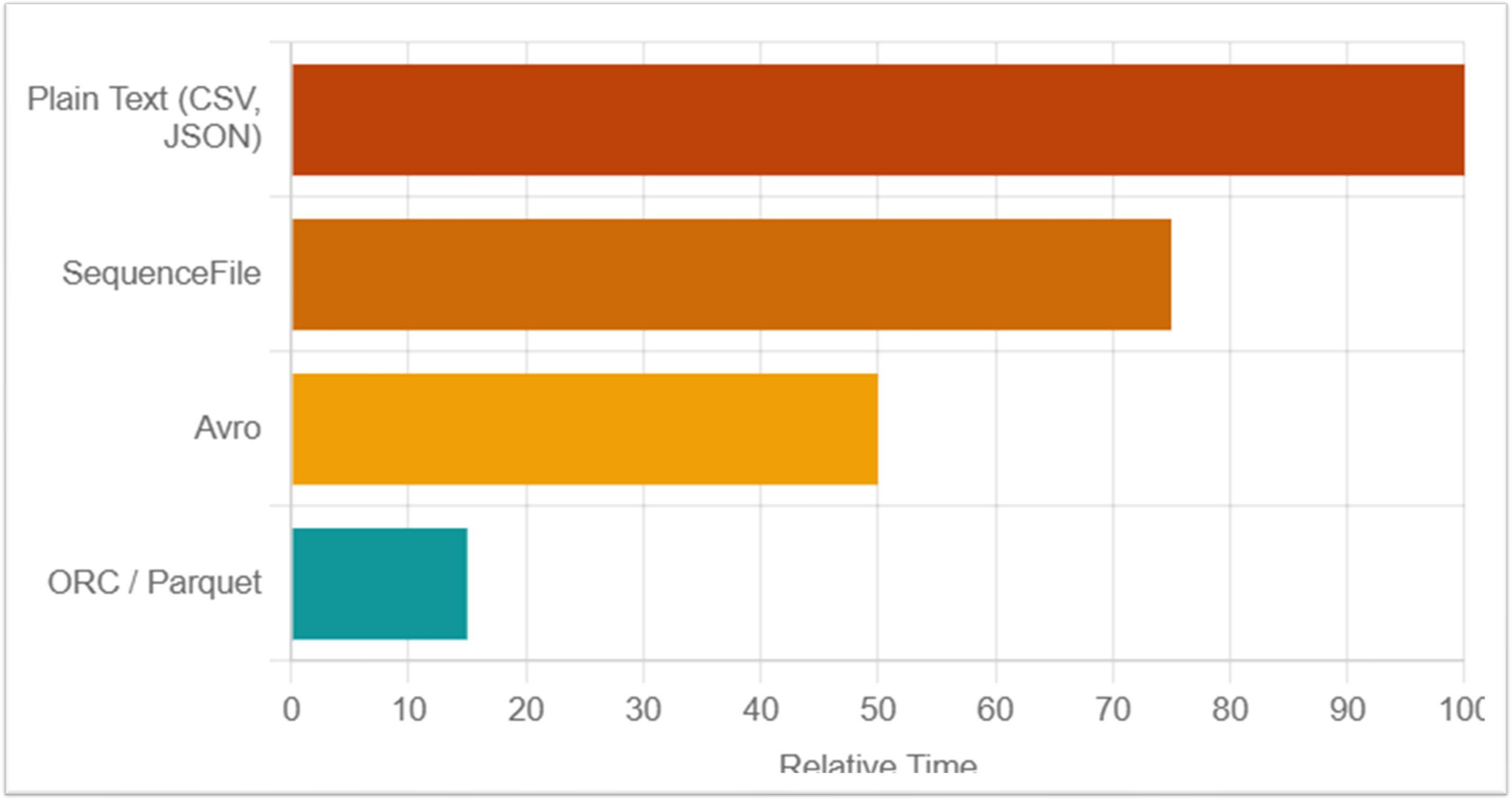
Learn REGEX:

Basic Training: https://www.w3schools.com/python/python_regex.asp
YouTube: https://youtu.be/wx-SuoZXtuQ

University of East London

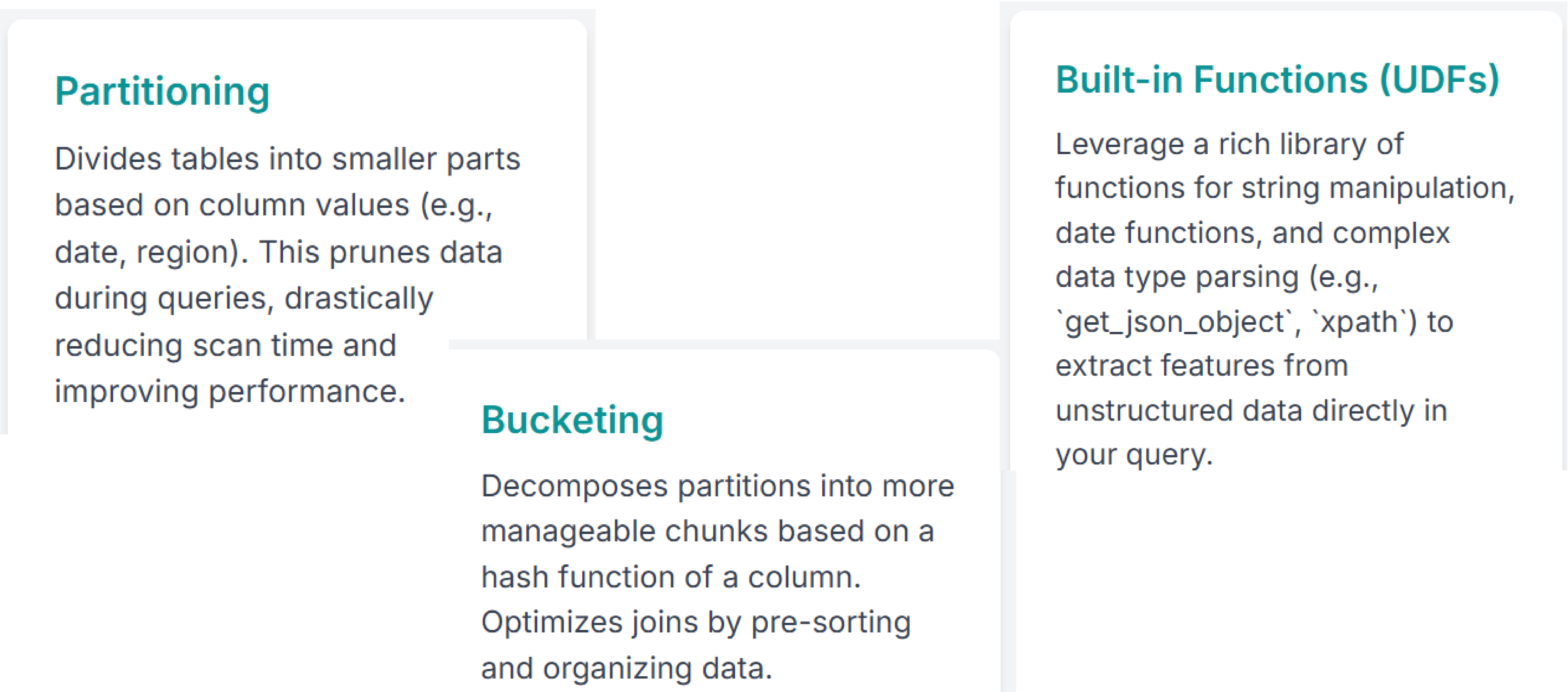
File Format Performance Impact

- After initial processing, converting to optimized columnar formats like Parquet or ORC dramatically speeds up subsequent analytical queries.
- Add **STORED AS PARQUET**; at the end of CREATE TABLE intermediate_logs_parquet()



Advanced Hive Techniques & Tricks

Optimize productivity with powerful Hive features designed for complex data.



Partitioning

Divides tables into smaller parts based on column values (e.g., date, region). This prunes data during queries, drastically reducing scan time and improving performance.

Bucketing

Decomposes partitions into more manageable chunks based on a hash function of a column. Optimizes joins by pre-sorting and organizing data.

Built-in Functions (UDFs)

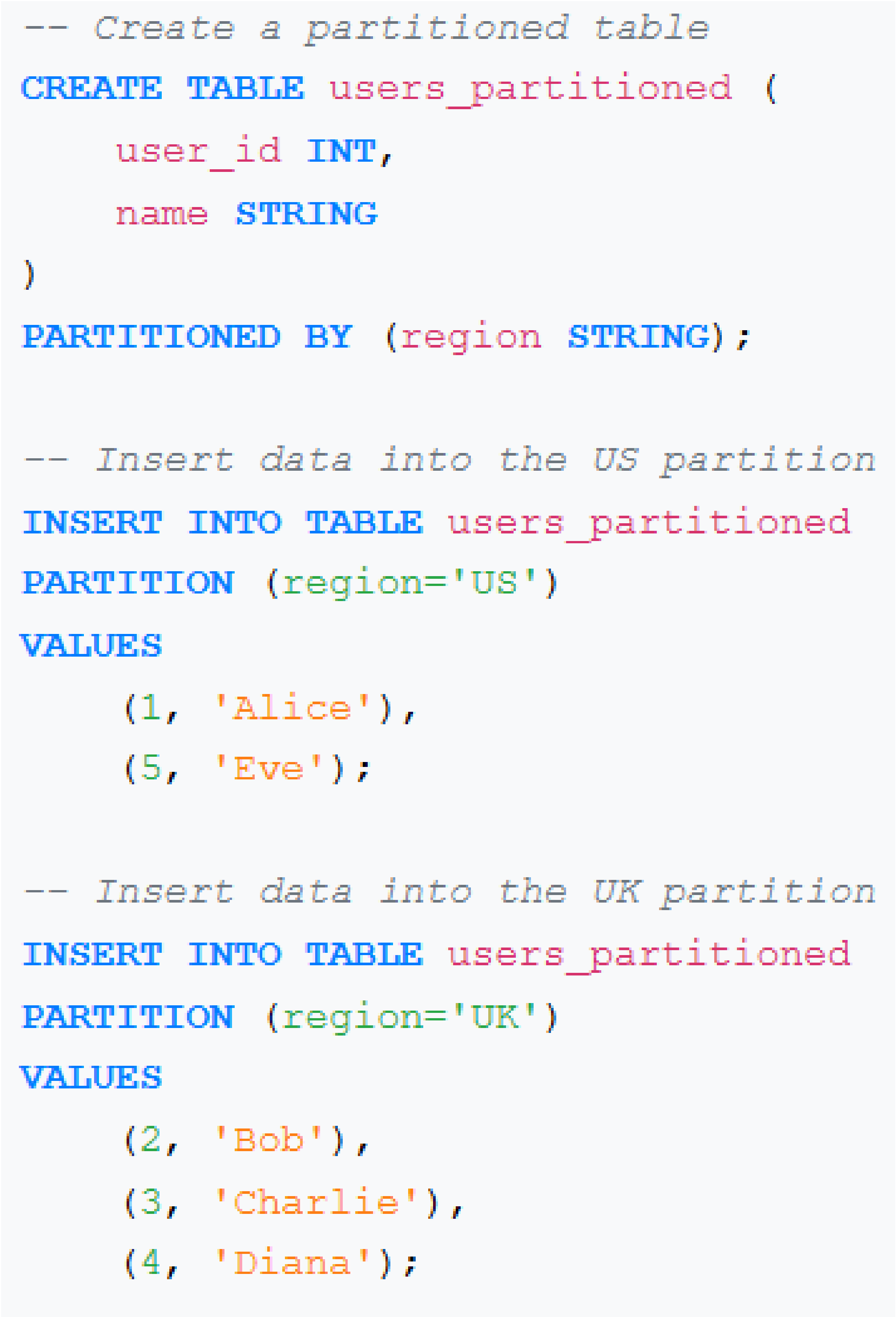
Leverage a rich library of functions for string manipulation, date functions, and complex data type parsing (e.g., `get_json_object`, `xpath`) to extract features from unstructured data directly in your query.

University of East London

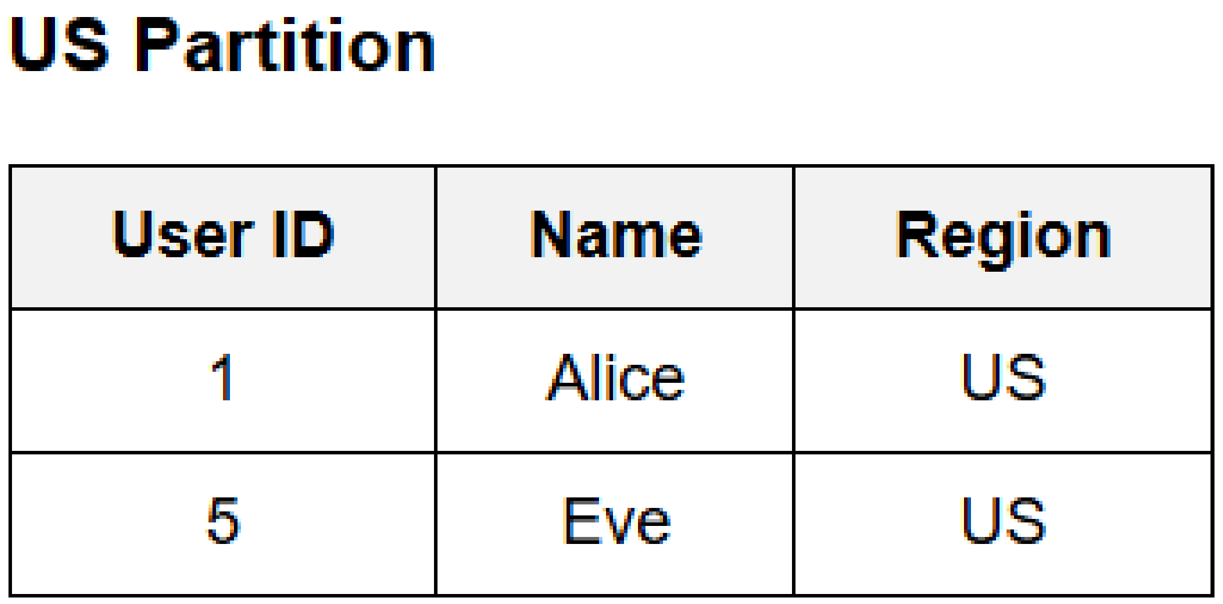
Partitioning

Divides tables into smaller parts based on column values (e.g., date, region). This reduces scan time and improves query performance.

user_id	name	region
1	Alice	US
2	Bob	UK
3	Charlie	UK
4	Diana	UK
5	Eve	US

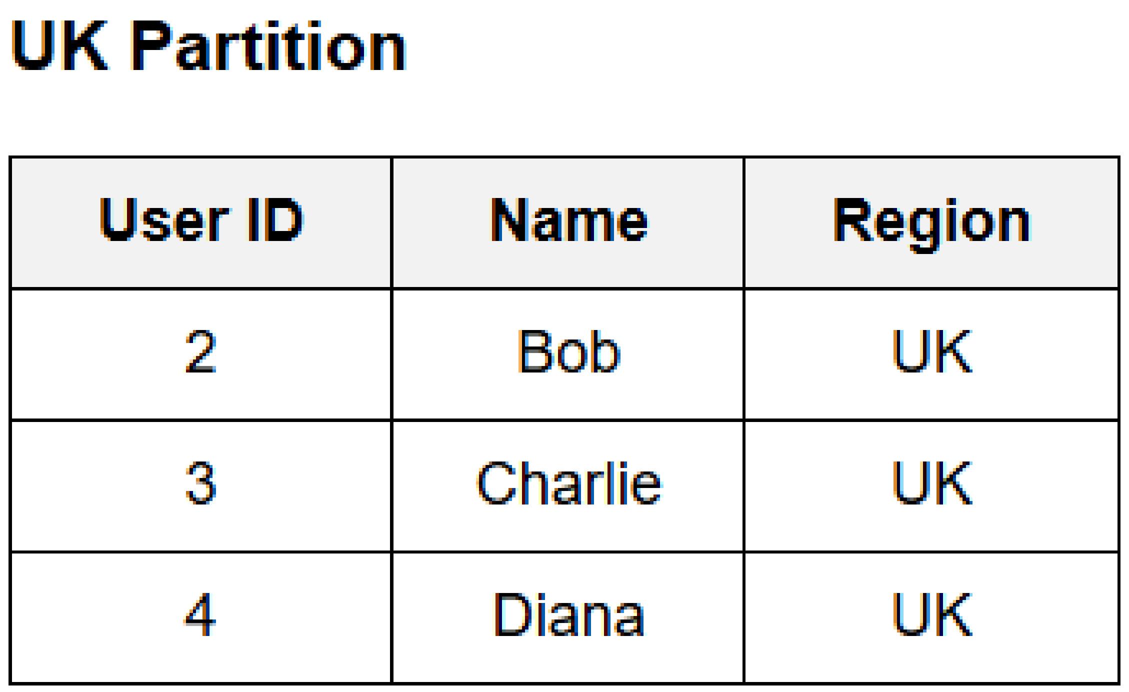


```
-- Create a partitioned table
CREATE TABLE users_partitioned (
    user_id INT,
    name STRING
)
PARTITIONED BY (region STRING);

-- Insert data into the US partition
INSERT INTO TABLE users_partitioned
PARTITION (region='US')
VALUES
    (1, 'Alice'),
    (5, 'Eve');

-- Insert data into the UK partition
INSERT INTO TABLE users_partitioned
PARTITION (region='UK')
VALUES
    (2, 'Bob'),
    (3, 'Charlie'),
    (4, 'Diana');
```



US Partition

User ID	Name	Region
1	Alice	US
5	Eve	US



UK Partition

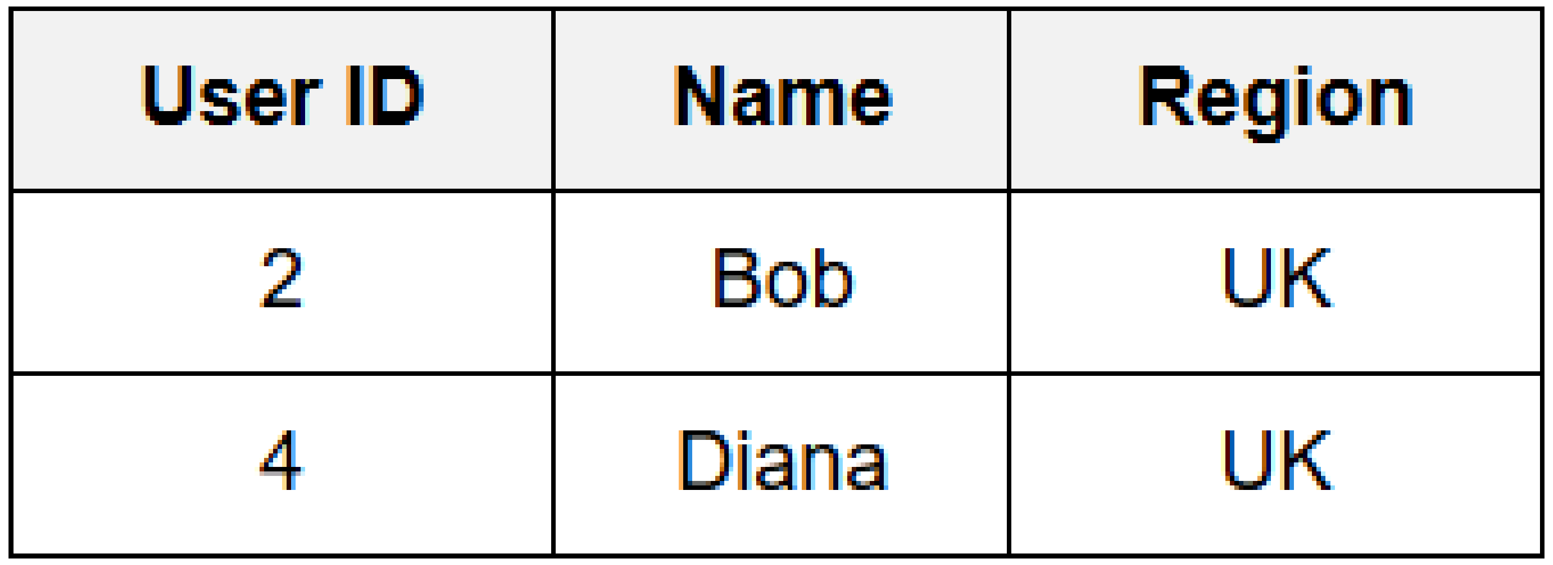
User ID	Name	Region
2	Bob	UK
3	Charlie	UK
4	Diana	UK

University of East London

Bucketing

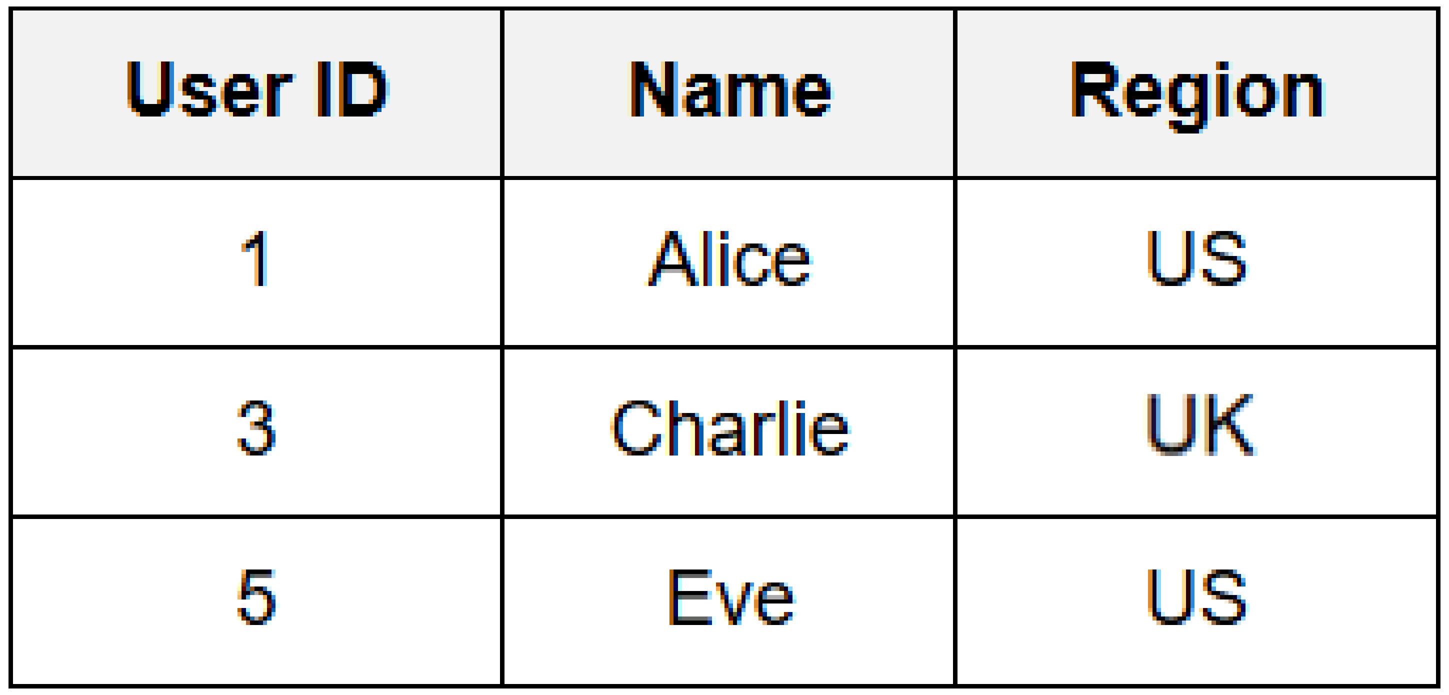
Bucketing is a method of dividing data into a fixed number of buckets (subsets) based on a hashing function or a specific range.

Bucket 1

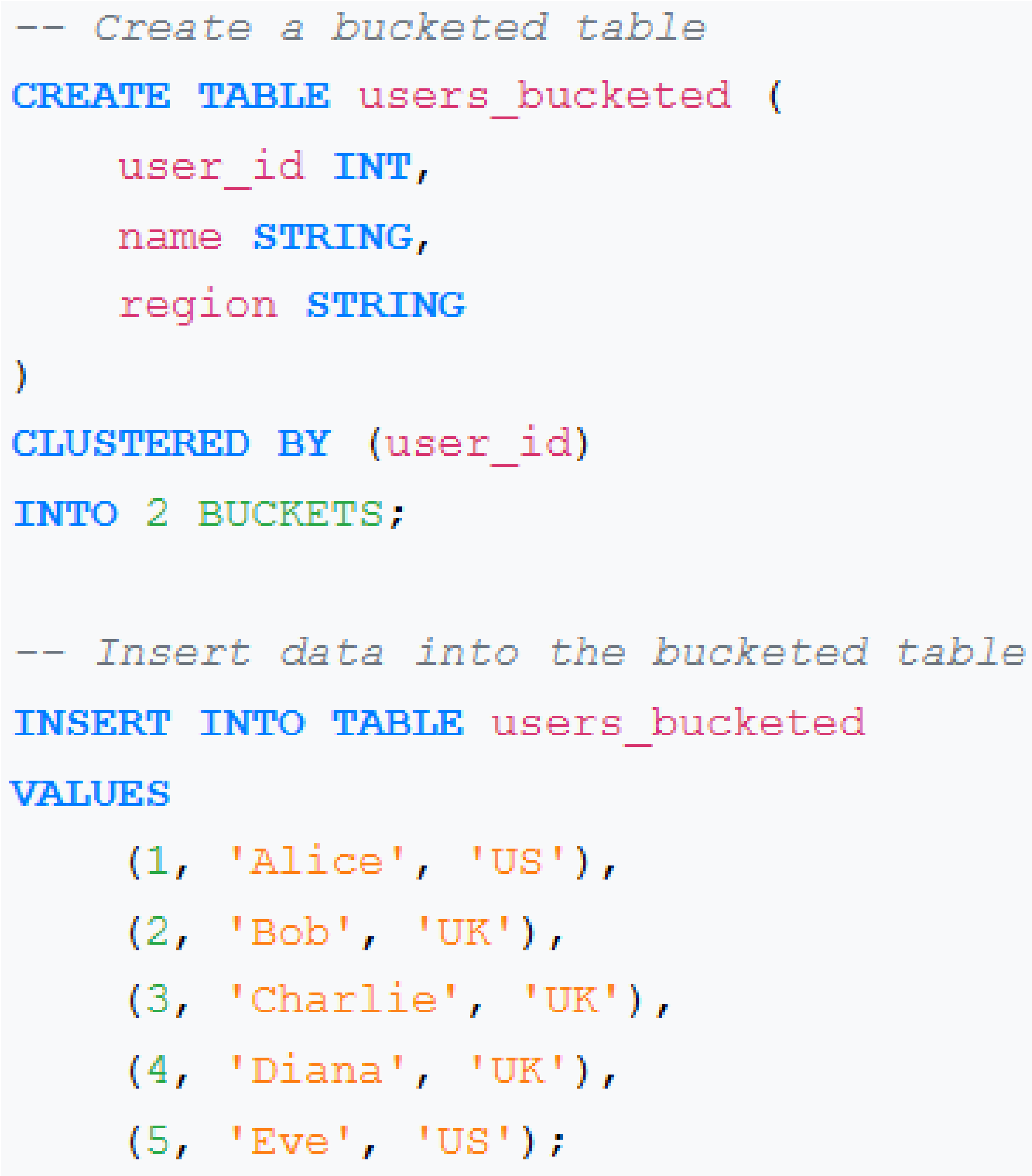


User ID	Name	Region
2	Bob	UK
4	Diana	UK

Bucket 2



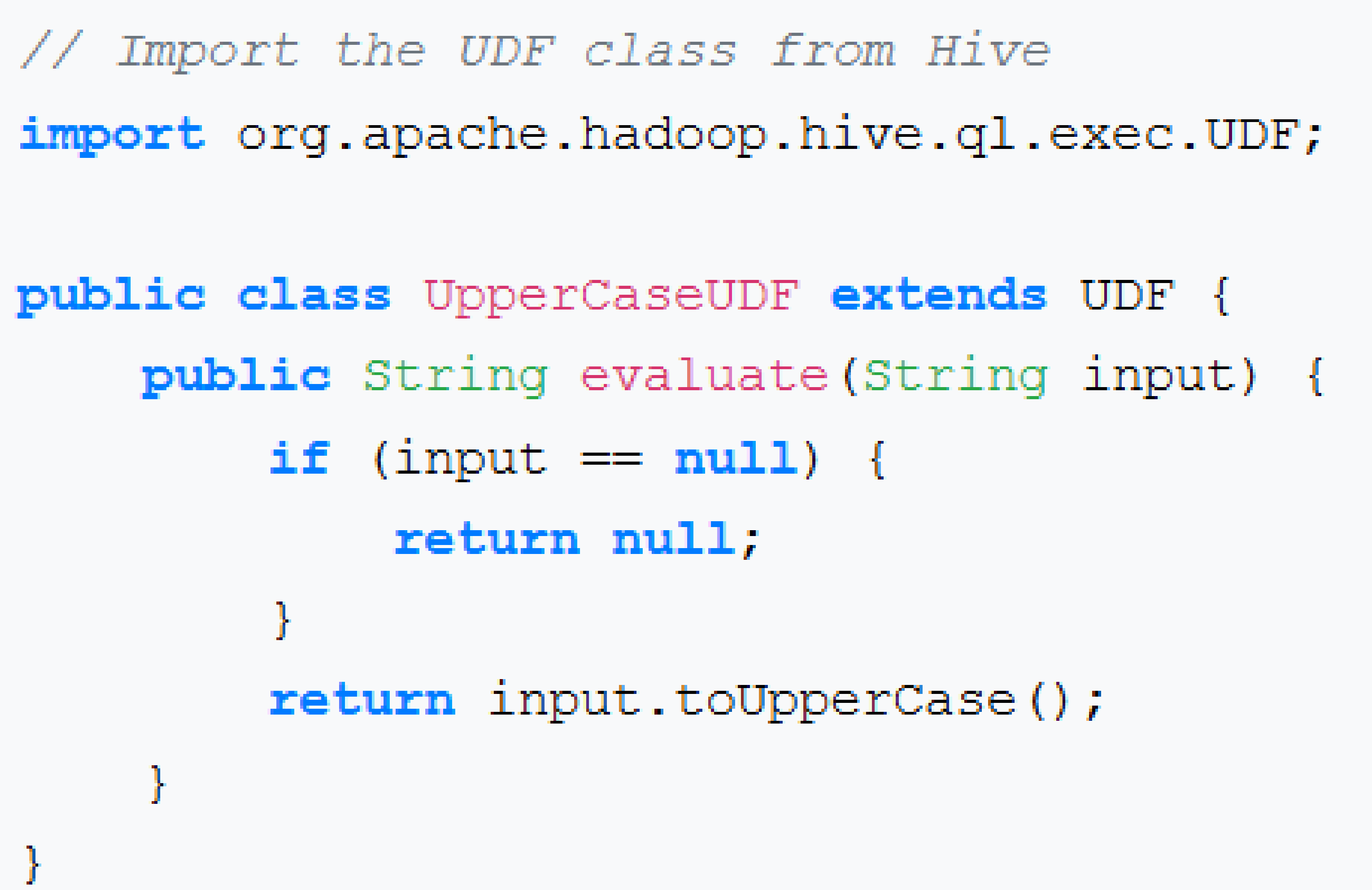
User ID	Name	Region
1	Alice	US
3	Charlie	UK
5	Eve	US



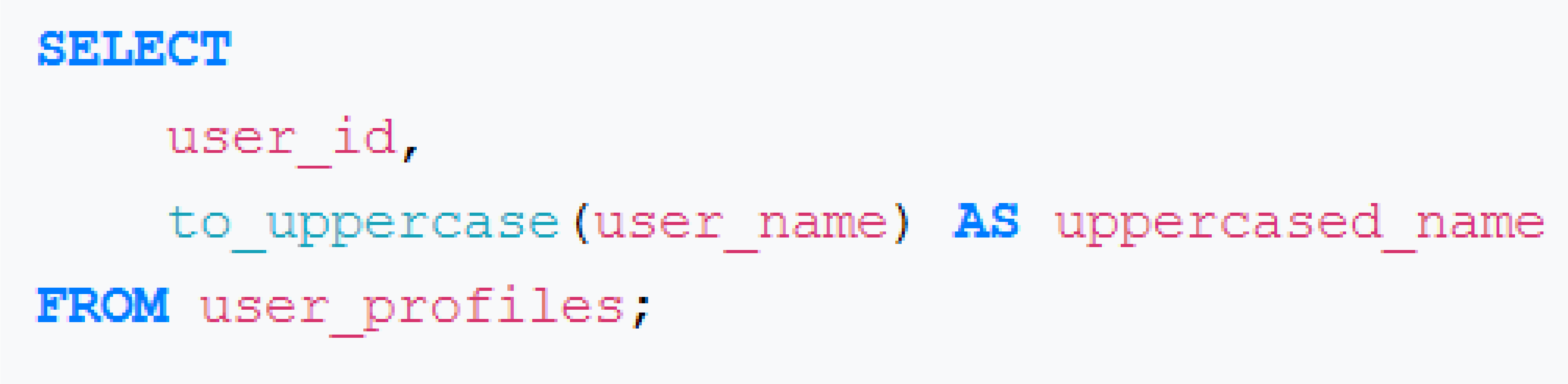
```
-- Create a bucketed table
CREATE TABLE users_bucketed (
    user_id INT,
    name STRING,
    region STRING
)
CLUSTERED BY (user_id)
INTO 2 BUCKETS;

-- Insert data into the bucketed table
INSERT INTO TABLE users_bucketed
VALUES
    (1, 'Alice', 'US'),
    (2, 'Bob', 'UK'),
    (3, 'Charlie', 'UK'),
    (4, 'Diana', 'UK'),
    (5, 'Eve', 'US');
```

UDFs

Have a rich library of functions for string manipulation, date parsing, and unstructured data processing.



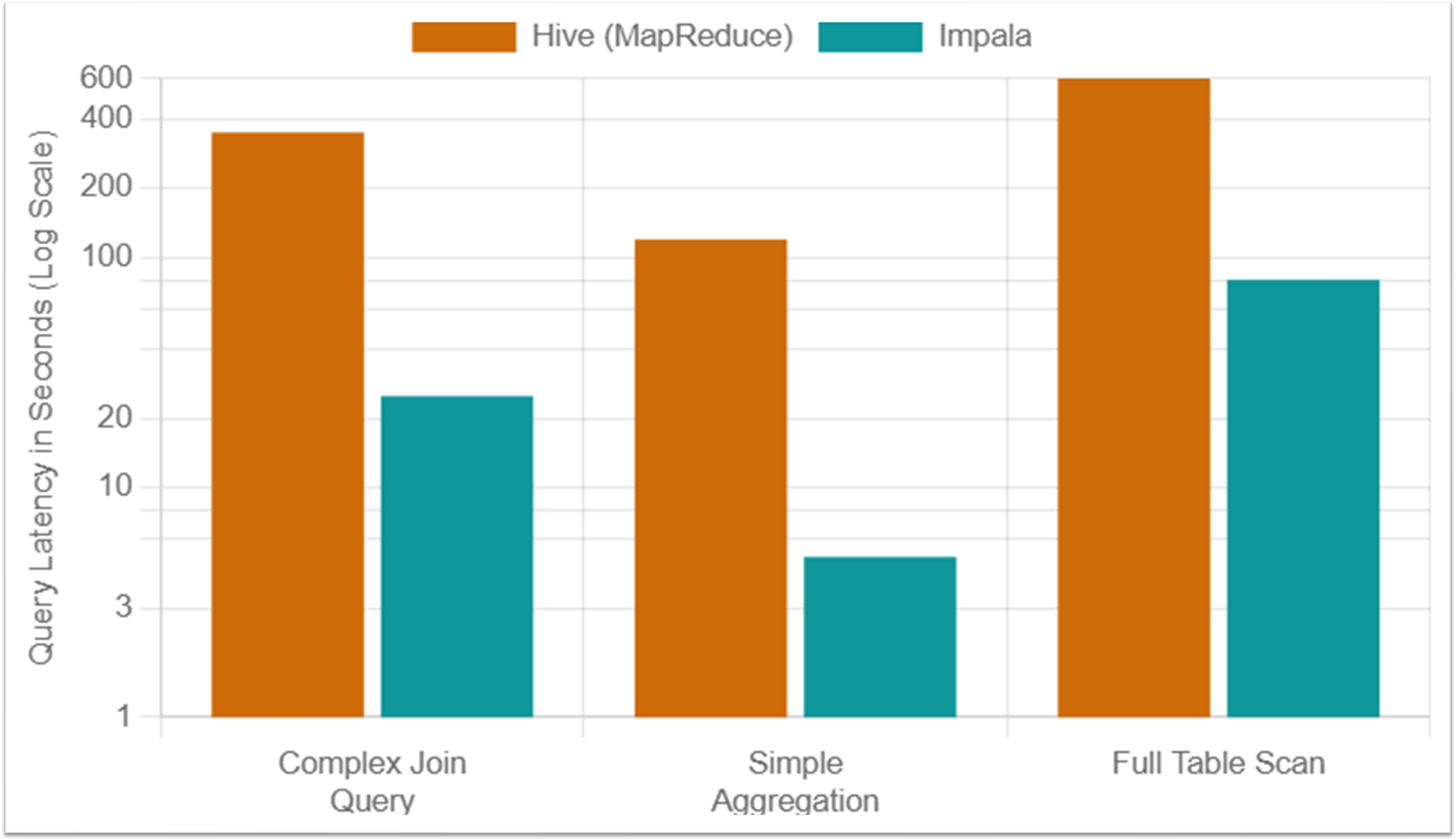
```java
// Import the UDF class from Hive
import org.apache.hadoop.hive.ql.exec.UDF;

public class UpperCaseUDF extends UDF {
    public String evaluate(String input) {
        if (input == null) {
            return null;
        }
        return input.toUpperCase();
    }
}
```



```sql
SELECT
    user_id,
    to_uppercase(user_name) AS uppercased_name
FROM user_profiles;
```

Accelerating Queries with Impala

Impala's in-memory, distributed query engine provides significantly lower latency compared to Hive's batch-oriented MapReduce approach.



Tutorial 5

Working with Unstructured Data in Hadoop

- Link to video: https://youtu.be/ZM87yiOrUZ8

Summary

- Discussed the challenges of Unstructured Data
- Discussed REGEX/Hive/Impala for Unstructured Data
- Discussed how to make queries fast
- Discussed Advanced Hive Optimizations (Partitioning, Bucketing, UDFs)

University of East London