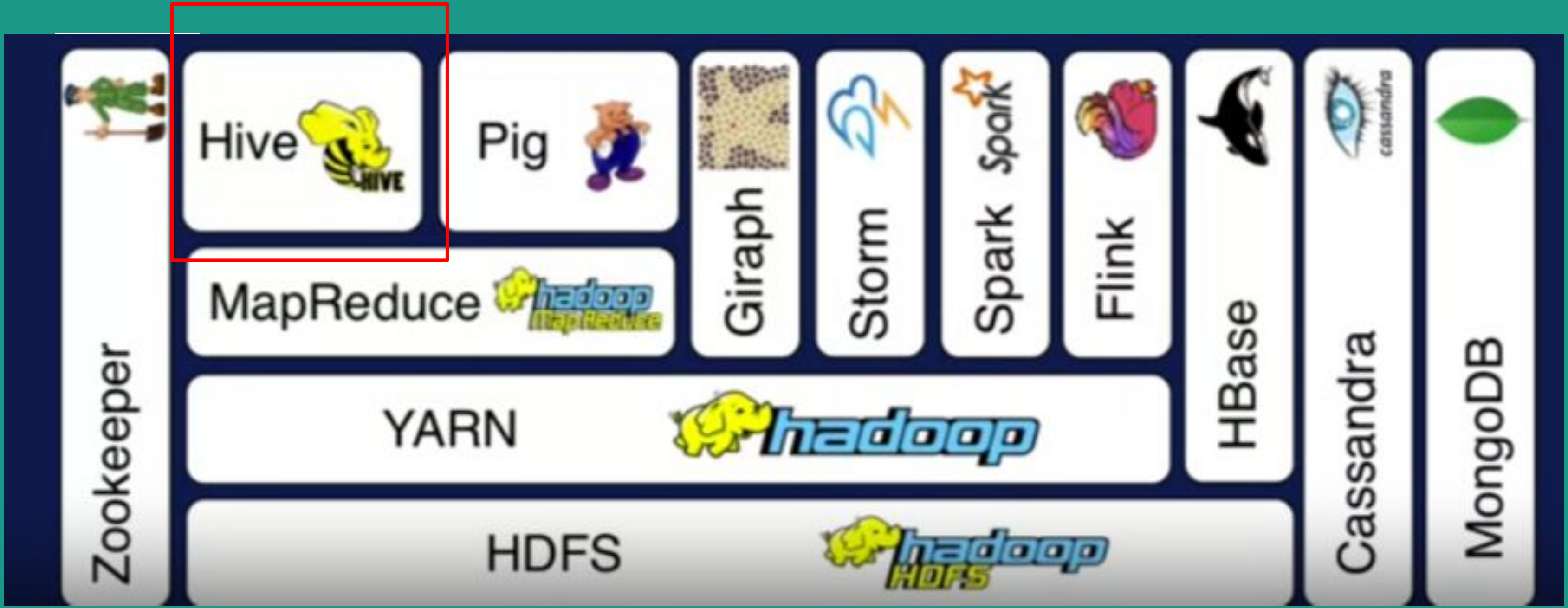
HIVE

Welcome to the Zoo!



What is Hive?

Hive: SQL for Hadoop

- Translates SQL-like queries into the world of MapReduce.
- Makes Hadoop accessible to analysts, not just programmers.

What is Hive?

Hive: SQL for Hadoop

- Translates SQL-like queries into the world of MapReduce.
- Makes Hadoop accessible to analysts, not just programmers.

SQL?

SQL - Quick Refresher

SQL
Structured Query Language

SQL: The Language of Structured Data

- SQL is how we interact with many databases.

SQL

Structured Query Language

SQL: The Language of Structured Data

Name	Age	City
Alice	25	New York
Bob	24	New York
---	---	---

Table Name : student_records

- Columns - Name, Age, City
- SELECT Name
  FROM student_records;

Alice
Bob

SQL

Structured Query Language

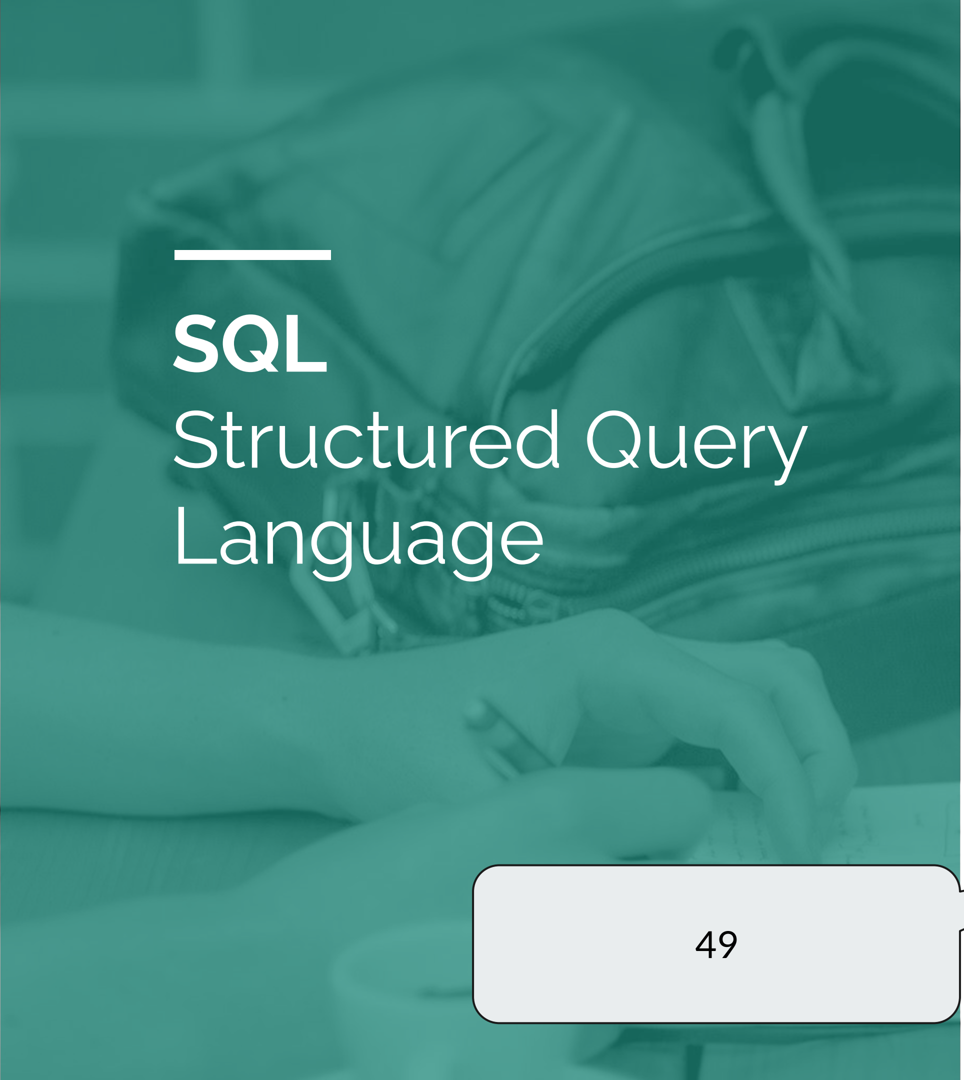
SQL: The Language of Structured Data

Name	Age	City
Alice	25	New York
Bob	24	New York
---	---	---

Table Name : student_records

- Columns - Name, Age, City
- SELECT DISTINCT City FROM student_records;

New York



SQL

Structured Query Language

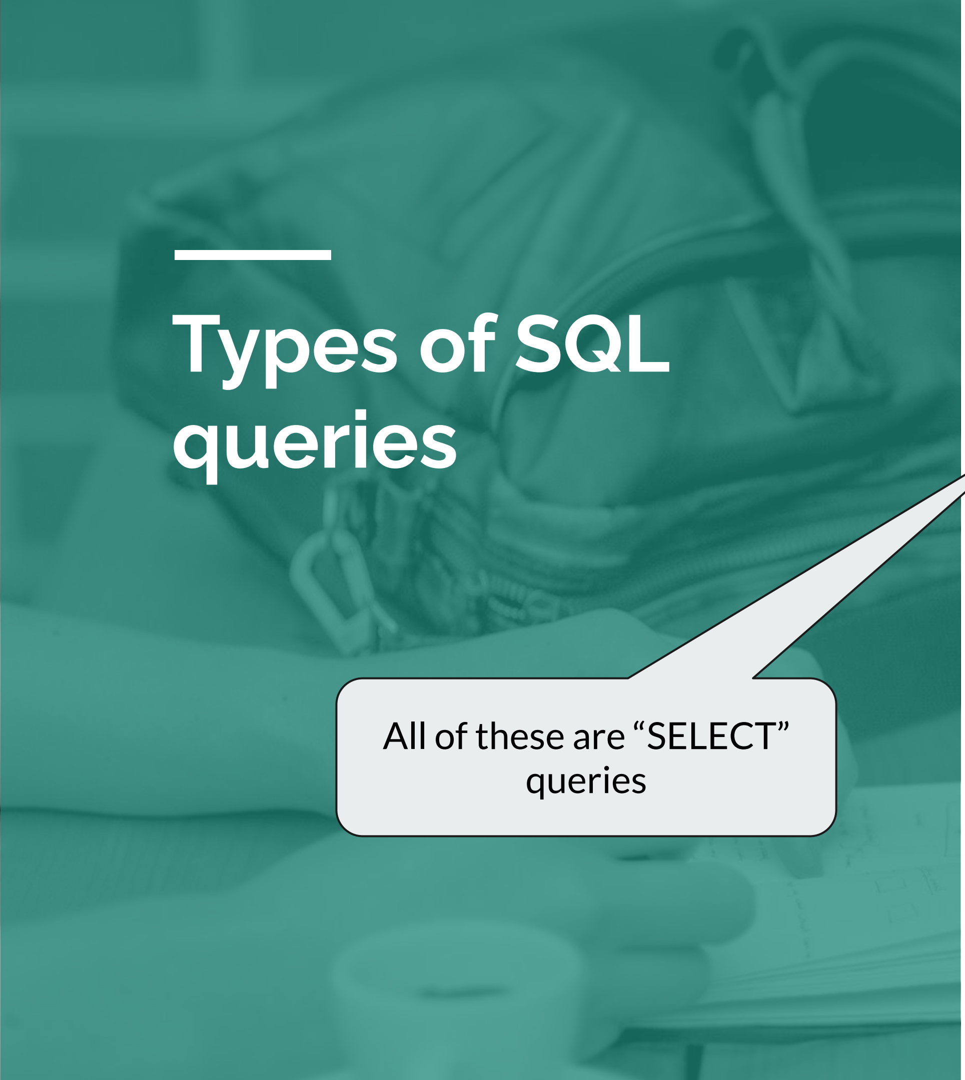
SQL: The Language of Structured Data

Name	Age	City
Alice	25	New York
Bob	24	New York
---	---	---

Table Name : student_records

- Columns - Name, Age, City
- SELECT SUM(Age)
  FROM student_records;

49



Types of SQL queries

All of these are "SELECT" queries

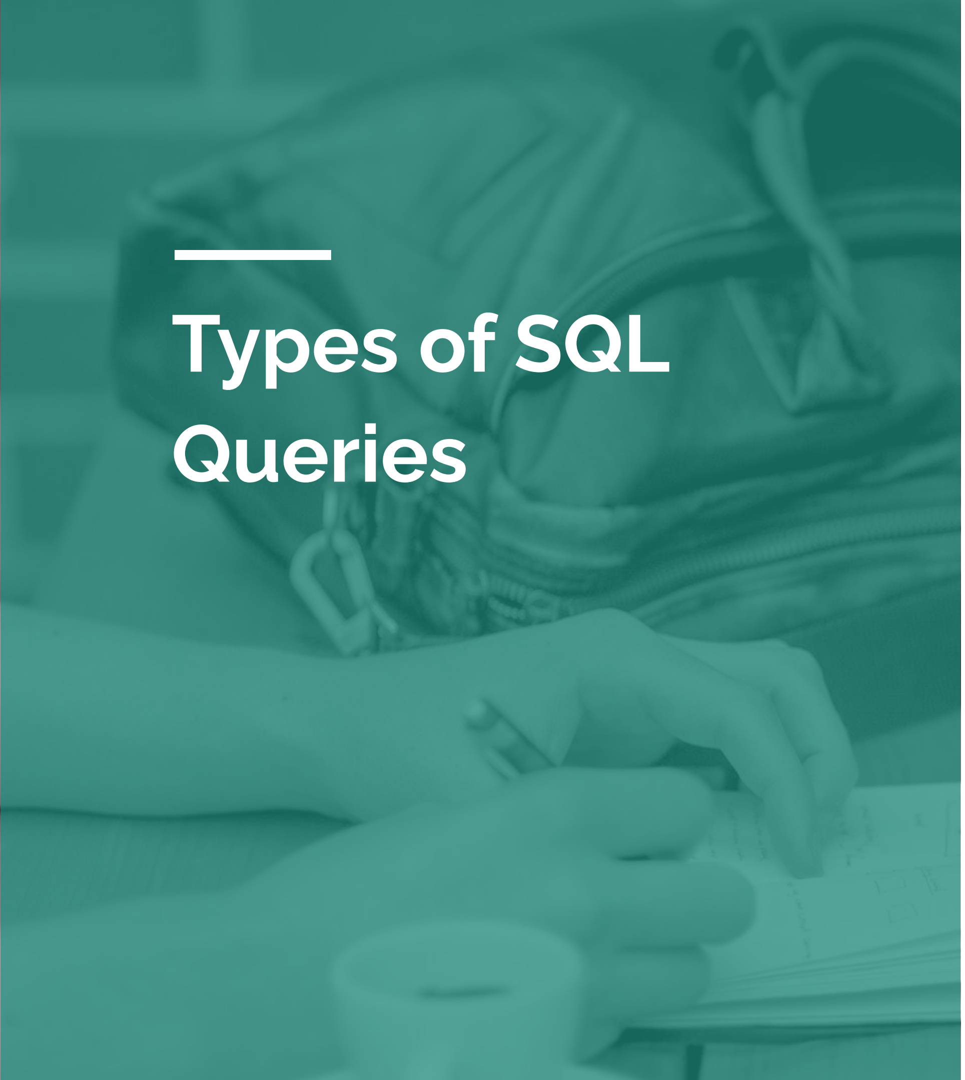
Queries we just saw ...

-
  ```
  SELECT  Name
  FROM student_records;
  ```

-
  ```
  SELECT  DISTINCT City
  FROM student_records;
  ```

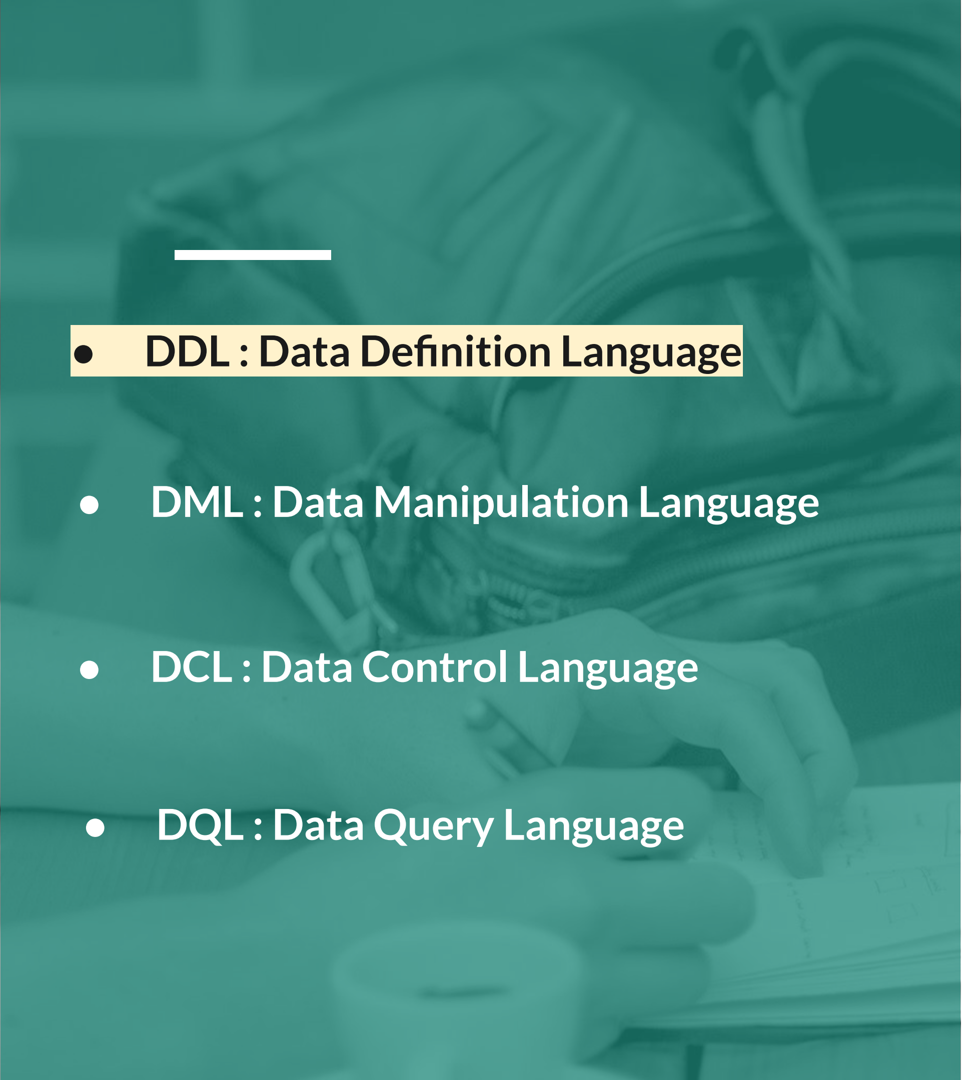
-
  ```
  SELECT SUM(Age)
  FROM student_records;
  ```

SQL - Types of SQL Queries



Types of SQL Queries

- DDL : Data Definition Language
- DML : Data Manipulation Language
- DCL : Data Control Language
- DQL : Data Query Language



- DDL : Data Definition Language
- DML : Data Manipulation Language
- DCL : Data Control Language
- DQL : Data Query Language

Data Definition Language

- Commands for building and modifying tables.
    - CREATE
    - ALTER
    - DROP

- DDL : Data Definition Language
- DML : Data Manipulation Language
- DCL : Data Control Language
- DQL : Data Query Language

Data Manipulation Language

- Working with the data inside tables
    - INSERT
    - UPDATE
    - DELETE

- DDL : Data Definition Language
- DML : Data Manipulation Language
- DCL : Data Control Language
- DQL : Data Query Language

Data Control Language

- Managing permissions and access
    - ○ **GRANT**
    - ○ **REVOKE**

- DDL : Data Definition Language
- DML : Data Manipulation Language
- DCL : Data Control Language
- DQL : Data Query Language

Data Query Language

- All about asking questions and getting answers
  - **SELECT**

HIVE == Big Data SQL

HIVE: Big Data SQL

Hive: SQL Power for Hadoop

- **HiveQL**: Hive uses a language very similar to SQL called HiveQL.
- **The Key Difference:** Hive is designed to work with massive amounts of data stored on Hadoop (HDFS).

HIVE: Big Data SQL

Hive == SQL

https://hortonworks.com/wp-content/uploads/2016/05/Hortonworks.CheatSheet.SQLtoHive.pdf

- **Similarities**:
  - Core Structure
  - SELECT for choosing columns
  - FROM to specify the table
  - WHERE for filtering conditions

- **Concepts:** Many familiar ideas carry over:
  - **Joins (inner, outer, etc.)**
  - **Aggregations (SUM, AVG, COUNT)**
  - **Sorting and Grouping**

HIVE: Big Data SQL

Hive == SQL?

Differences:

Syntax Nuances: While similar, there are minor differences:

- How dates are handled
- Some function names might vary

Data Types: Hive supports more complex types to handle semi-structured data common in Hadoop (arrays, maps, etc.).

HIVE: Big Data SQL

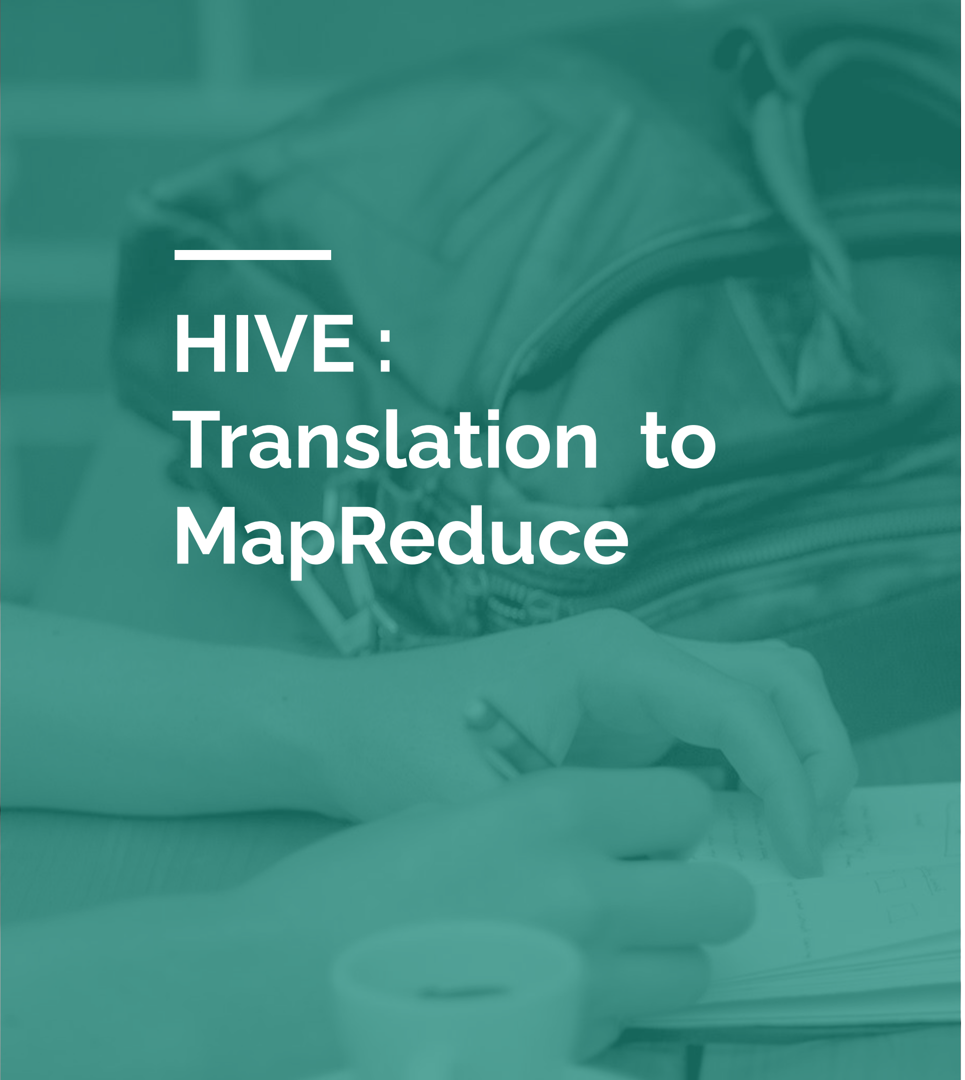
Hive == SQL?

Performance: Hive often has overhead due to translation into MapReduce.

Well-designed SQL on a database is usually faster.

Schema on Read: In Hive, you often define the structure as you query, rather than a rigid upfront schema like databases.

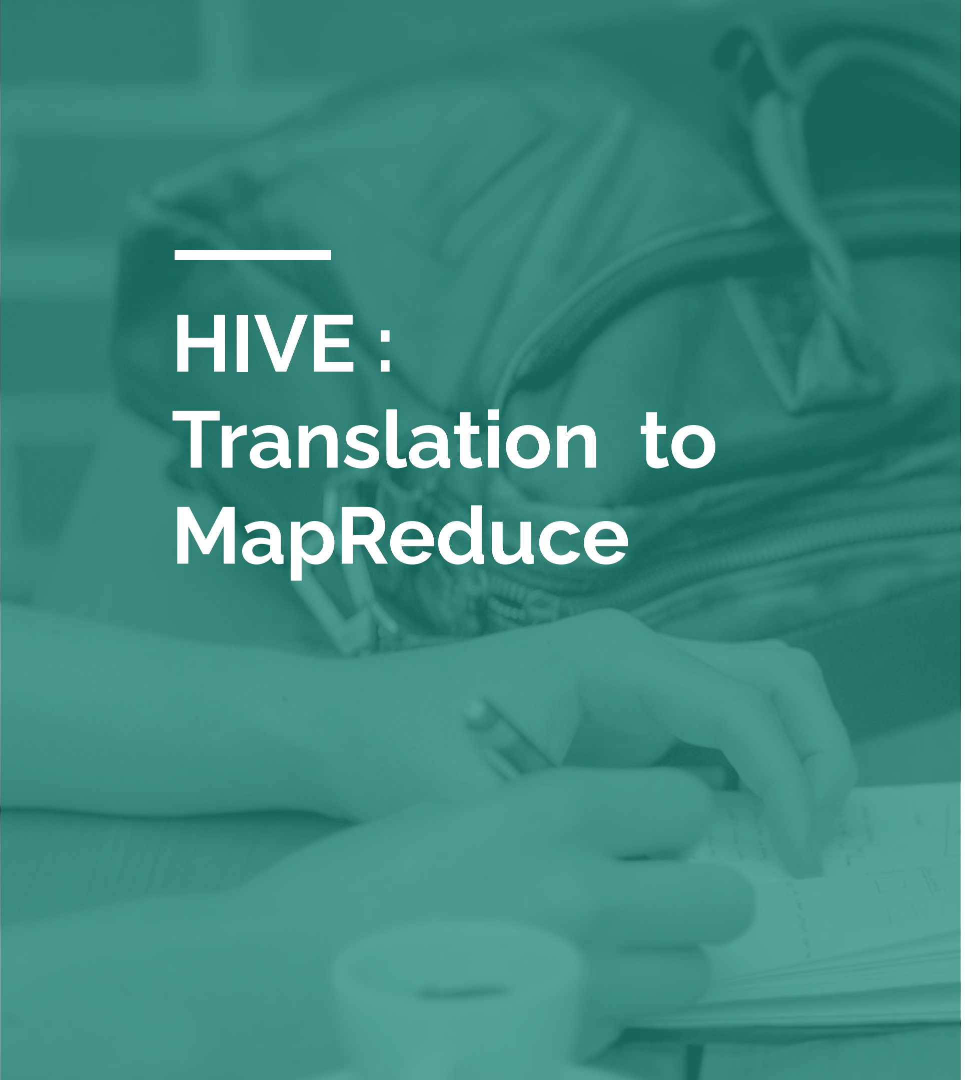
HIVE : Translation to MapReduce



HIVE : Translation to MapReduce

Our Sample Data

order_id	customer_name	city	product	price
1001	Alice	New York	Book	10
1002	Bob	Los Angeles	Shirt	20
1003	Charlie	New York	Movie	15
1004	David	Los Angeles	Game	30
1005	Alice	Seattle	Music	5



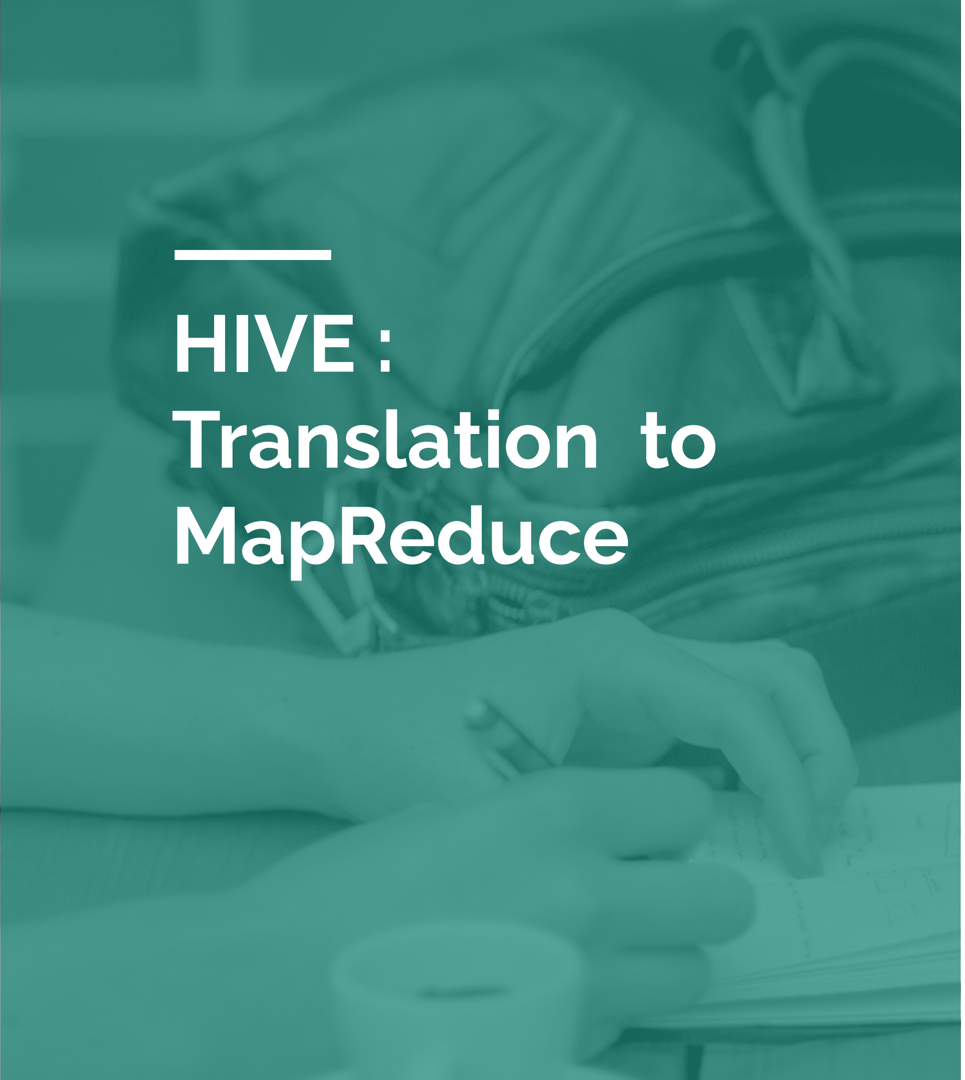
HIVE : Translation to MapReduce

Our Sample Data

order_id	customer_name	city	product	price
1001	Alice	New York	Book	10
1002	Bob	Los Angeles	Shirt	20
1003	Charlie	New York	Movie	15
1004	David	Los Angeles	Game	30
1005	Alice	Seattle	Music	5

Our Sample Query

Give me order count for each city.



HIVE : Translation to MapReduce

Our Sample Data

order_id	customer_name	city	product	price
1001	Alice	New York	Book	10
1002	Bob	Los Angeles	Shirt	20
1003	Charlie	New York	Movie	15
1004	David	Los Angeles	Game	30
1005	Alice	Seattle	Music	5

Our Sample Query

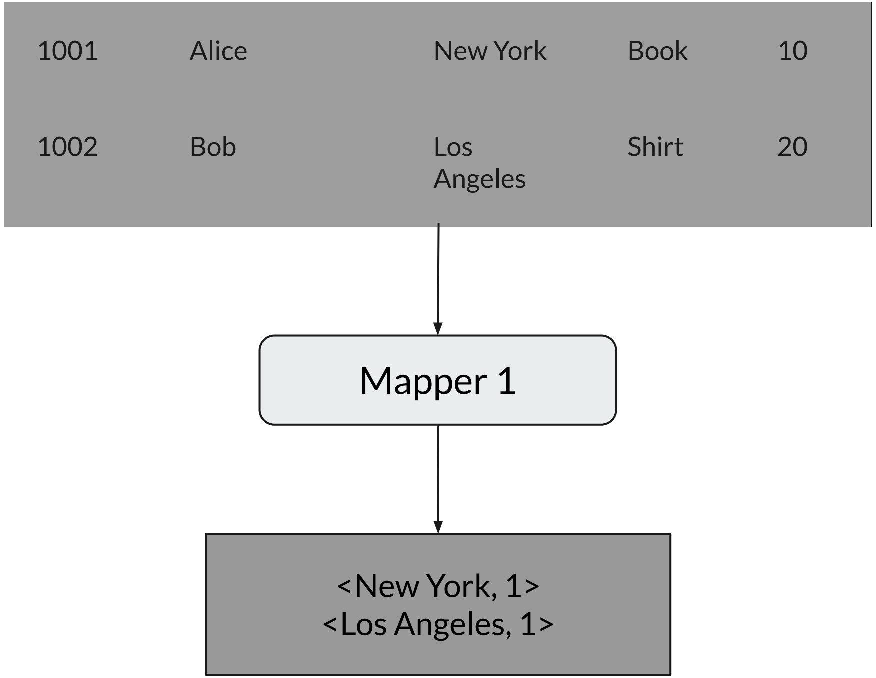
```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```

```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```

order_id	customer_name	city	product	price
1001	Alice	New York	Book	10
1002	Bob	Los Angeles	Shirt	20
1003	Charlie	New York	Movie	15
1004	David	Los Angeles	Game	30
1005	Alice	Seattle	Music	5

Mapper

Assuming our 'orders' data is **split** across multiple blocks in **HDFS**, with multiple Mappers working in parallel.



1001	Alice	New York	Book	10
1002	Bob	Los Angeles	Shirt	20

Mapper 1

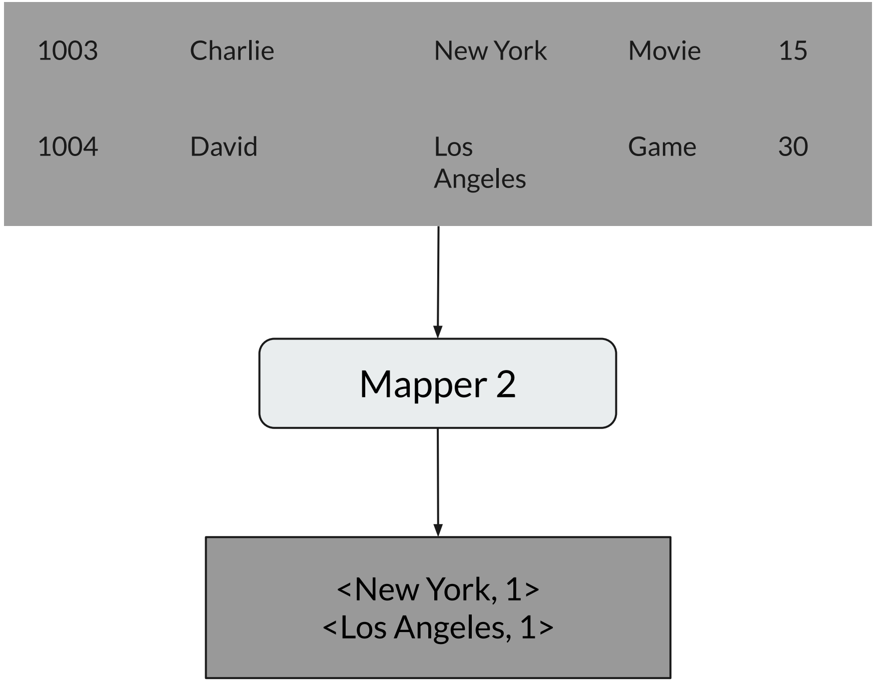
<New York, 1>
<Los Angeles, 1>

```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```

order_id	customer_name	city	product	price
1001	Alice	New York	Book	10
1002	Bob	Los Angeles	Shirt	20
1003	Charlie	New York	Movie	15
1004	David	Los Angeles	Game	30
1005	Alice	Seattle	Music	5

Mapper

Assuming our 'orders' data is **split** across multiple blocks in **HDFS**, with multiple Mappers working in parallel.



1003	Charlie	New York	Movie	15
1004	David	Los Angeles	Game	30

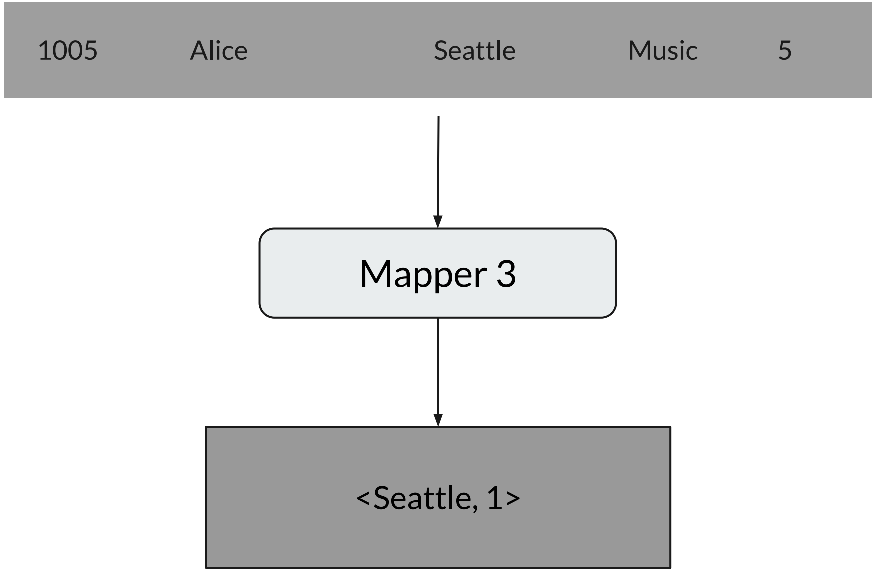
<New York, 1>
<Los Angeles, 1>

```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```

order_id	customer_name	city	product	price
1001	Alice	New York	Book	10
1002	Bob	Los Angeles	Shirt	20
1003	Charlie	New York	Movie	15
1004	David	Los Angeles	Game	30
1005	Alice	Seattle	Music	5

Mapper

Assuming our 'orders' data is **split** across multiple blocks in **HDFS**, with multiple Mappers working in parallel.

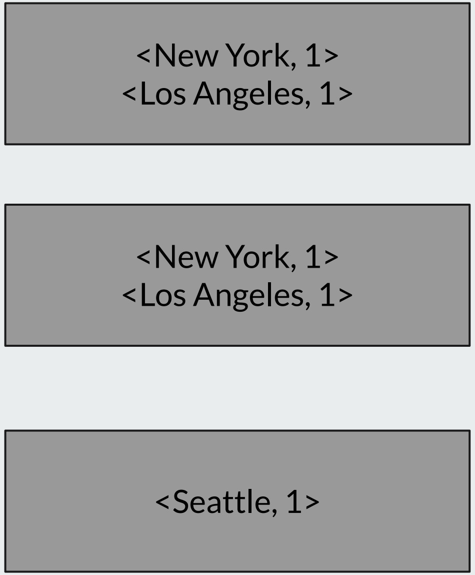


1005	Alice	Seattle	Music	5

Mapper 3

<Seattle, 1>

```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```



<New York, 1>
<Los Angeles, 1>

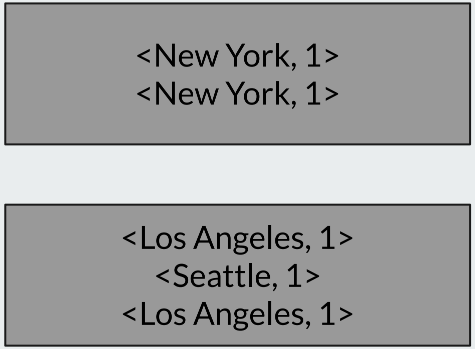
<New York, 1>
<Los Angeles, 1>

<Seattle, 1>

Shuffle & Sort

- All Key/Value pairs emitted by the Mappers are distributed across Reducers.

```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```



<New York, 1>
<New York, 1>

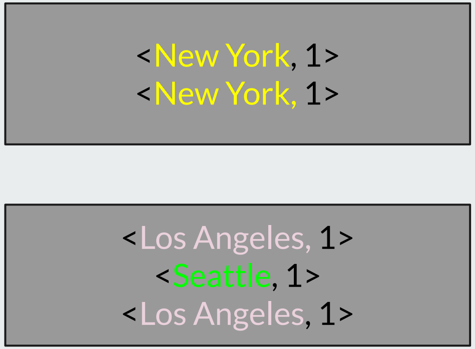
<Los Angeles, 1>
<Seattle, 1>
<Los Angeles, 1>

Shuffle & Sort

- All Key/Value pairs emitted by the Mappers are distributed across Reducers.

Shuffle & Sort

```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```



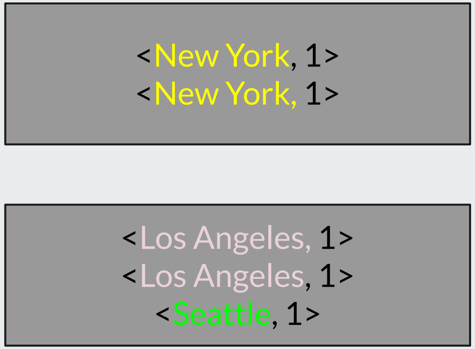
<New York, 1>
<New York, 1>

<Los Angeles, 1>
<Seattle, 1>
<Los Angeles, 1>

- All Key/Value pairs emitted by the Mappers are distributed across Reducers.
- Pairs with the same Key (e.g., <New York, 1>) are guaranteed to end up at the same Reducer.

Shuffling Done!

```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```



<New York, 1>
<New York, 1>

<Los Angeles, 1>
<Los Angeles, 1>
<Seattle, 1>

Shuffle & Sort

- All Key/Value pairs emitted by the Mappers are distributed across Reducers.
- Pairs with the same Key (e.g., <New York, 1>) are guaranteed to end up at the same Reducer.
- Within each Reducer's portion of the data, the Keys are sorted.

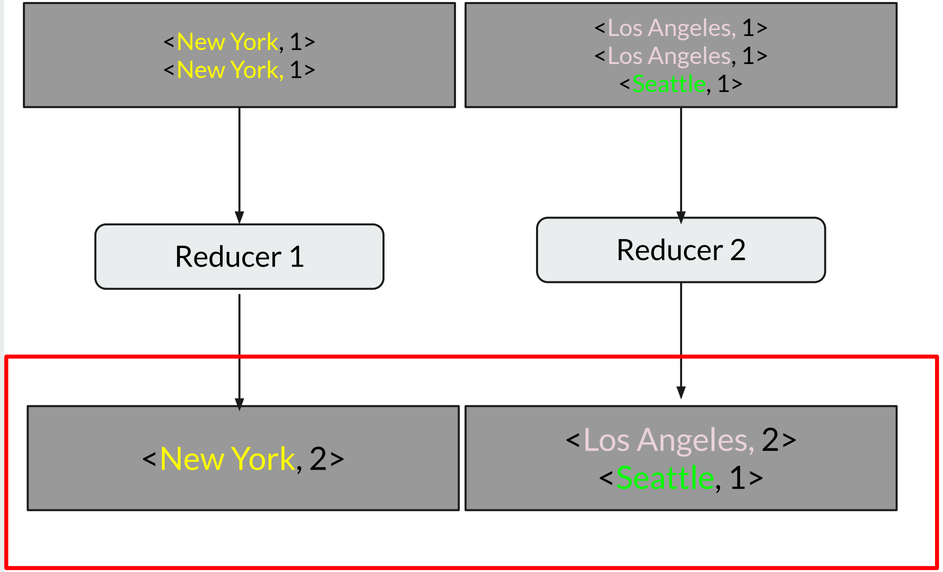
```
SELECT city, COUNT(*) as order_count
FROM orders
GROUP BY city;
```

<New York, 1>
<New York, 1>

<Los Angeles, 1>
<Los Angeles, 1>
<Seattle, 1>

Reduce

Assuming that we had 2 reducers -



<New York, 1>
<New York, 1>

<Los Angeles, 1>
<Los Angeles, 1>
<Seattle, 1>

Reducer 1

Reducer 2

<New York, 2>

<Los Angeles, 2>
<Seattle, 1>

HIVE : More than MapReduce!

Not all queries require Mappers & Reducers …

HIVE : More than MapReduce!

Hive's Capabilities Extend Beyond Basic MapReduce

- **Simple Data Retrieval:**
  - `SELECT * FROM customers LIMIT 10; (First 10 customers)`

ta

...);

ture)

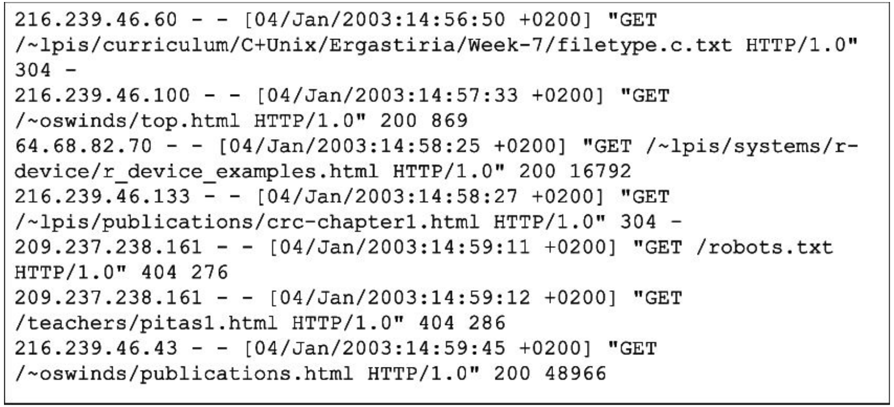
unt

Did we forget not all data is stored in tables??

HIVE : More than MapReduce!

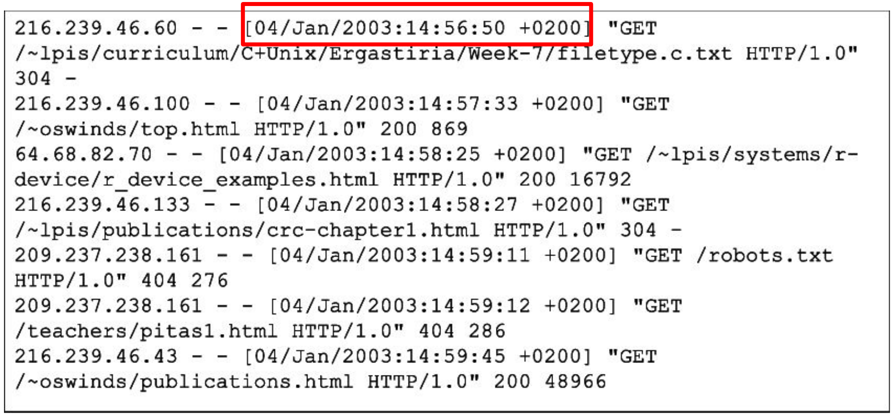
Hive: Non-Relational Data

- Table abstractions on top of raw HDFS files.
- Example -
    - Web Logs (Semi-Structured)



```
216.239.46.60 - - [04/Jan/2003:14:56:50 +0200] "GET
/~lpis/curriculum/C+Unix/Ergastiria/Week-7/filetype.c.txt HTTP/1.0"
304 -
216.239.46.100 - - [04/Jan/2003:14:57:33 +0200] "GET
/~oswinds/top.html HTTP/1.0" 200 869
64.68.82.70 - - [04/Jan/2003:14:58:25 +0200] "GET /~lpis/systems/r-
device/r_device_examples.html HTTP/1.0" 200 16792
216.239.46.133 - - [04/Jan/2003:14:58:27 +0200] "GET
/~lpis/publications/crc-chapter1.html HTTP/1.0" 304 -
209.237.238.161 - - [04/Jan/2003:14:59:11 +0200] "GET /robots.txt
HTTP/1.0" 404 276
209.237.238.161 - - [04/Jan/2003:14:59:12 +0200] "GET
/teachers/pitas1.html HTTP/1.0" 404 286
216.239.46.43 - - [04/Jan/2003:14:59:45 +0200] "GET
/~oswinds/publications.html HTTP/1.0" 200 48966
```

Hive: Non-Relational Data



```
216.239.46.60 - - [04/Jan/2003:14:56:50 +0200] "GET
/~lpis/curriculum/C+Unix/Ergastiria/Week-7/filetype.c.txt HTTP/1.0"
304 -
216.239.46.100 - - [04/Jan/2003:14:57:33 +0200] "GET
/~oswinds/top.html HTTP/1.0" 200 869
64.68.82.70 - - [04/Jan/2003:14:58:25 +0200] "GET /~lpis/systems/r-
device/r_device_examples.html HTTP/1.0" 200 16792
216.239.46.133 - - [04/Jan/2003:14:58:27 +0200] "GET
/~lpis/publications/crc-chapter1.html HTTP/1.0" 304 -
209.237.238.161 - - [04/Jan/2003:14:59:11 +0200] "GET /robots.txt
HTTP/1.0" 404 276
209.237.238.161 - - [04/Jan/2003:14:59:12 +0200] "GET
/teachers/pitas1.html HTTP/1.0" 404 286
216.239.46.43 - - [04/Jan/2003:14:59:45 +0200] "GET
/~oswinds/publications.html HTTP/1.0" 200 48966
```

```
CREATE TABLE weblogs (
  timestamp STRING,
  ip_address STRING,
  url STRING,
  extra_info STRING
)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE;
```

How did Hive came into existence?

The Birth of Hive at Facebook (circa 2007)

- **The Problem:** Facebook's data was growing massively, and traditional databases were struggling to handle the scale.

- **The Need:** Analysts and engineers needed a way to query this vast data, preferably using familiar SQL-like concepts.

- **The Solution:** Facebook engineers developed Hive internally. Its core idea: translate SQL-like queries into MapReduce jobs executable on Hadoop clusters.

What’s with the Bee Theme?

The name Hive, the honeycomb-like logo – it's a playful nod to the idea of extracting valuable insights from Big Data.

Which of the following is the primary use of Hive?

A. Real-time processing of streaming data
B. Replacing traditional relational databases (MySQL, etc.)
C. Querying and analysis of large datasets stored in Hadoop (HDFS)
D. Machine learning model development

The language used in Hive is called:

A. Python
B. Pig Latin
C. Java
D. HiveQL

Which statement adds a new column named 'total_sales' to an existing table 'orders'?

```
A. CREATE TABLE orders (total_sales DOUBLE);
B. INSERT INTO orders VALUES ('total_sales', ...);
C. UPDATE orders SET total_sales = ...;
D. ALTER TABLE orders ADD COLUMN total_sales DOUBLE;
```

True or False: Hive queries typically run faster than equivalent queries on a small relational database.

What does the 'GROUP BY' clause do in a HiveQL query?

A. Sorts the data in the result
B. Selects the columns to include in the output
C. Creates groups of rows based on common values in a column
D. Filters data based on a condition