HADOOP ECOSYSTEM

MODULE 5

Apache PIG

Apache Pig, developed by Yahoo researchers, simplifies the complex process of big data analysis and processing in Hadoop. Serving as an abstraction layer, Pig allows developers to work with large datasets without the need for extensive Java coding required by MapReduce.

Apache Pig stands as a valuable tool in the Hadoop ecosystem, offering developers a simplified approach to big data processing. Its versatility, ease of use, and seamless integration with MapReduce make it a preferred choice for efficient data analysis in large-scale environments.

Key Features of Apache PIG

01

Abstraction Layer

Apache Pig acts as a higher-level abstraction over MapReduce.

Developers can perform data processing without intricate Java programming.

02

Ease of Use

Provides a user-friendly interface, reducing the challenges of working with lengthy and complex Java code, Enables quick analysis and processing of large datasets.

03

MapReduce Integration

Seamlessly integrates with MapReduce, bridging the gap for data analysts unfamiliar with the complexities of MapReduce programming, Developers can leverage Pig's simplicity while still utilizing the power of MapReduce.

04

High-Level Abstraction

Built on top of Hadoop, Apache Pig offers a high-level abstraction, Reduces the development time associated with complex MapReduce programs.

Other Features

Pig Latin Language: Developers use Pig Latin, a language specifically designed for Pig, to express data transformations. Simplifies the translation of tasks into map and reduce stages

Versatility: Named after a pig for its omnivorous nature, Apache Pig can work with any type of data, regardless of structure or format.

01

Architecture

PIG Latin Interpreter + PIG Latin Language + Execution Mechanisms (User Defined Functions, Embedded executions, Grunt Shells)

Architecture

Pig Latin Interpreter:

a. Processes and analyzes massive datasets using Pig Latin scripts.
b. Provides an interface for programmers within the Hadoop environment.

Pig Latin Language:

a. Expresses various data operations such as join, filter, sort, load, and group.
b. Simplifies the creation of Pig scripts for specific data processing tasks.

Execution Mechanisms:

a. **UDFs (User Defined Functions):** Custom functions that extend Pig's functionality.
b. **Embedded Execution:** Integration into larger applications.
c. **Grunt Shells:** Interactive shell for executing Pig Latin commands.

Apache Pig Workflow Components

Parser:

Function: Handles Pig scripts initially, performing syntax checks, type checking, and miscellaneous verifications.

- Output: Generates a Directed Acyclic Graph (DAG) representing Pig Latin statements and logical operators.
- Representation: Nodes represent logical operators, and edges depict data flows within the script.

Apache Pig Workflow Components

Optimizer:

Function: Receives the DAG output from the parser and conducts optimization activities.

Optimization Activities: Split, merge, projection, pushdown, transform, reorder, etc.

Objective: Enhance query performance by eliminating unnecessary data or columns through optimization techniques

Apache Pig Workflow Components

Compiler:

Function: Compiles the optimized output into a sequence of MapReduce jobs.

Conversion: Automatically translates Pig jobs into MapReduce jobs.

Optimization: Rearranges execution order to optimize performance.

Apache Pig Workflow Components

Execution Engine:

Function: Submits the compiled MapReduce jobs to the execution engine.

Execution: Runs the jobs on the Hadoop platform to generate the desired results.

Interaction: Utilizes DUMP statements for on-screen result display and STORE statements to store results in HDFS (Hadoop Distributed File System).

Apache PIG Execution Models

1. Local Mode:

Description: Suitable for small datasets.

Execution Environment: Runs in a single JVM on the local host with a local file system.

Parallel Execution: Not feasible for parallel mapper execution, as all files run on the localhost.

Command: Use pig -x local to specify local mode.

2. MapReduce Mode:

Default Mode: Apache Pig uses MapReduce mode by default.

Execution Environment: Executes Pig Latin statements on data stored in Hadoop Distributed File System (HDFS).

Command: Use pig -x mapreduce to specify MapReduce mode.

Pig Latin Data Model:

Atom

Definition: A single value stored in string form, serving as a number or string.

Atomic Values: Integer, double, float, byte array, and char array.

Example: "Kiara" or 27.

Bag

Definition: A collection of tuples.

Example: {(Kiara, 27), (Kehsav, 45)}.

Tuple

Definition: A record containing an ordered set of fields, similar to a row in an RDBMS.

Example: (Kiara, 27).

Map

Definition: A key-value pair set, with a unique key (char array) and a value of any kind.

Example: [name#Kiara, age#27].

Relation

Definition: A bag of tuples.

Example: Bag{(Kiara, 27), (Kehsav, 45)} is a relation.

Difference Between Apache PIG & Map Reduce

Apache Pig	Map Reduce
Scripting language	Compiled language
Provides a higher level of abstraction	Provides a low level of abstraction
Requires a few lines of code (10 lines of code can summarize 200 lines of Map Reduce code)	Requires a more extensive code (more lines of code)
Requires less development time and effort	Requires more development time and effort
Lesser code efficiency	Higher efficiency of code in comparison to Apache Pig

Execution Flow of a Pig Job:

1.Pig Script Development:

i. **Task:** The developer writes a Pig script in the Pig Latin language.

ii. **Storage:** The script is stored in the local file system.

2.Script Submission: Action: The Pig script is submitted for execution.

3.Connection with Compiler: Action: Apache Pig establishes a connection with the compiler.

4.Compiler Processing:

i. **Task:** The compiler processes the Pig script.

ii. **Output:** Generates a series of MapReduce jobs as the output.

5.MapReduce Job Generation: Action: Pig compiler generates a series of MapReduce jobs based on the Pig script.

6.Raw Data from HDFS: Source: Raw data is sourced from Hadoop Distributed File System (HDFS).

7.MapReduce Job Execution:

i. **Process:** Pig compiler executes MapReduce jobs on the raw data.

ii. **Intermediate Results:** Results are obtained after MapReduce jobs processing.

8.Storage of Results: Destination: Intermediate results are stored back into HDFS.

9.Job Completion: Status: MapReduce jobs are marked as completed.

10.Accessing Results: Output: The processed results are now accessible, either through the DUMP statement for on-screen display or STORE statements for storage in HDFS.

Data Type	Description	Comparison
int	Signed 32-bit integer	Similar to Integer in Java
long	Fully signed 64-bit number	Similar to Long in Java
float	Signed 32-bit floating-point number	Similar to Java's float
double	Floating-point 63-bit number	Similar to Java's double
chararray	List of characters in Unicode format (UTF-8)	Compatible with Java character data types
bytearray	Represents bytes by default	Default for unspecified data file type
boolean	Represents a value that is true or false	-

Data Types in PIG

These data types cover a range of numeric representations (int, long, float, double), character data (chararray), binary data (bytearray), and boolean values (boolean). This diversity allows Apache Pig to handle various types of data in a flexible manner during data processing.

Tuples, Bag and Map in PIG

A **Tuple** is an ordered collection of fields, functioning much like a row in a relational database. Each field within a Tuple can have a different data type, making it suitable for representing structured data with varying attributes. Tuples are particularly useful when dealing with data entities that have distinct yet related characteristics.

A **Bag**, on the other hand, is an unordered set of Tuples. It serves as a container for holding multiple Tuples, accommodating varying lengths and data types. Bags are analogous to mathematical sets, making them handy for scenarios where the order of elements is not a primary concern, and you need to manage collections of Tuples efficiently.

The **Map** data type in Apache Pig is a key-value pair set. It associates a unique key, usually a char array, with a corresponding value, which can be of any data type. Maps are instrumental in situations where named attributes need to be linked with values, providing a structured and flexible means of organizing information.

Relational Operators in PIG

Pig Latin Operator	Description	Example
LOAD	To load the data from the file system (local/HDFS) into a relation.	data = LOAD 'input.txt' AS (name, age);
STORE	To save a relation to the file system (local/HDFS).	STORE data INTO 'output.txt';
FILTER	To remove unwanted rows from a relation.	filtered_data = FILTER data BY age > 18;
DISTINCT	To remove duplicate rows from a relation.	unique_data = DISTINCT data;
FOREACH, GENERATE	To generate data transformations based on columns of data.	processed_data = FOREACH data GENERATE name, age * 2;
STREAM	To transform a relation using an external program.	transformed_data = STREAM data THROUGH 'external_script';

Relational Operators in PIG

Pig Latin Operator	Description	Example
JOIN	To join two or more relations.	joined_data = JOIN data1 BY id, data2 BY id;
COGROUP	To group the data in two or more relations.	cogrouped_data = COGROUP data1 BY id, data2 BY id;
GROUP	To group the data in a single relation.	grouped_data = GROUP data BY age;
CROSS	To create the cross product of two or more relations.	cross_data = CROSS data1, data2;
ORDER	To arrange a relation in a sorted order based on one or more fields (ascending or descending).	sorted_data = ORDER data BY age DESC;
LIMIT	To get a limited number of tuples from a relation.	limited_data = LIMIT data 10;

Relational Operators in PIG

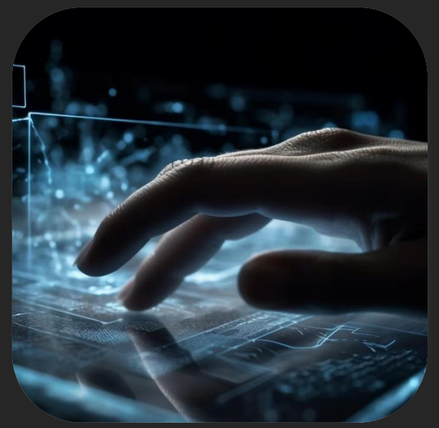
Pig Latin Operator	Description	Example
UNION	To combine two or more relations into a single relation.	merged_data = UNION data1, data2;
SPLIT	To split a single relation into two or more relations.	(subset1, subset2) = SPLIT data INTO data1 IF age > 30, data2 IF age <= 30;
DUMP	To print the contents of a relation on the console.	DUMP data;
DESCRIBE	To describe the schema of a relation.	DESCRIBE data;
EXPLAIN	To view the logical, physical, or MapReduce execution plans to compute a relation.	EXPLAIN data;
ILLUSTRATE	To view the step-by-step execution of a series of statements.	ILLUSTRATE data;

Limitations of PIG

Unhelpful Error Messages: Pig often provides unhelpful error messages, especially related to User Defined Functions (UDFs) in Python. Generic messages like "exec error in UDF" make it challenging to identify syntax or type errors.

Delayed Execution: Pig commands are not executed until an intermediate or final result is dumped or stored, leading to a delay in execution and increased iteration cycles during debugging.

Limitations of PIG



Development Stage: Despite being in use for a considerable time, Pig is still in development, which may affect its stability and maturity compared to more established tools.

Limited Community Support: Pig-related problems may not have readily available solutions on platforms like Google and StackOverflow, leading to challenges in finding support from the community.

Implicit Data Schema: Pig enforces data schema implicitly rather than explicitly, causing issues such as unexpected data structure transformations into byte arrays without explicit notice.

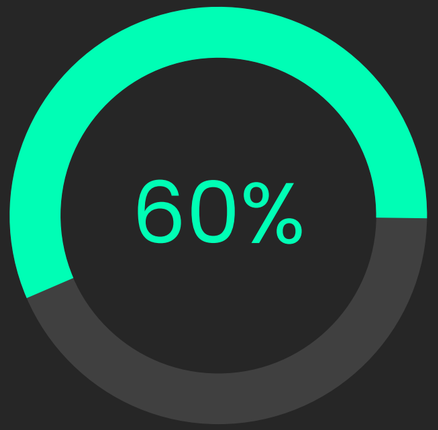
Lack of IDE/Plugin: Pig lacks a comprehensive Integrated Development Environment (IDE) or plugin support for popular text editors like Vim, limiting the development experience.

02

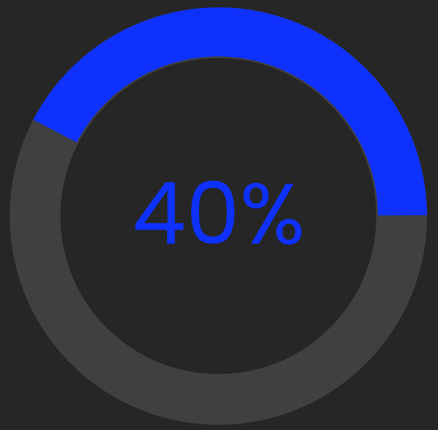
HIVE

Meaning of Hive, features, Architecture, Data types, Hive file format, Hive Query language (HQL), User Defined Functions

HIVE



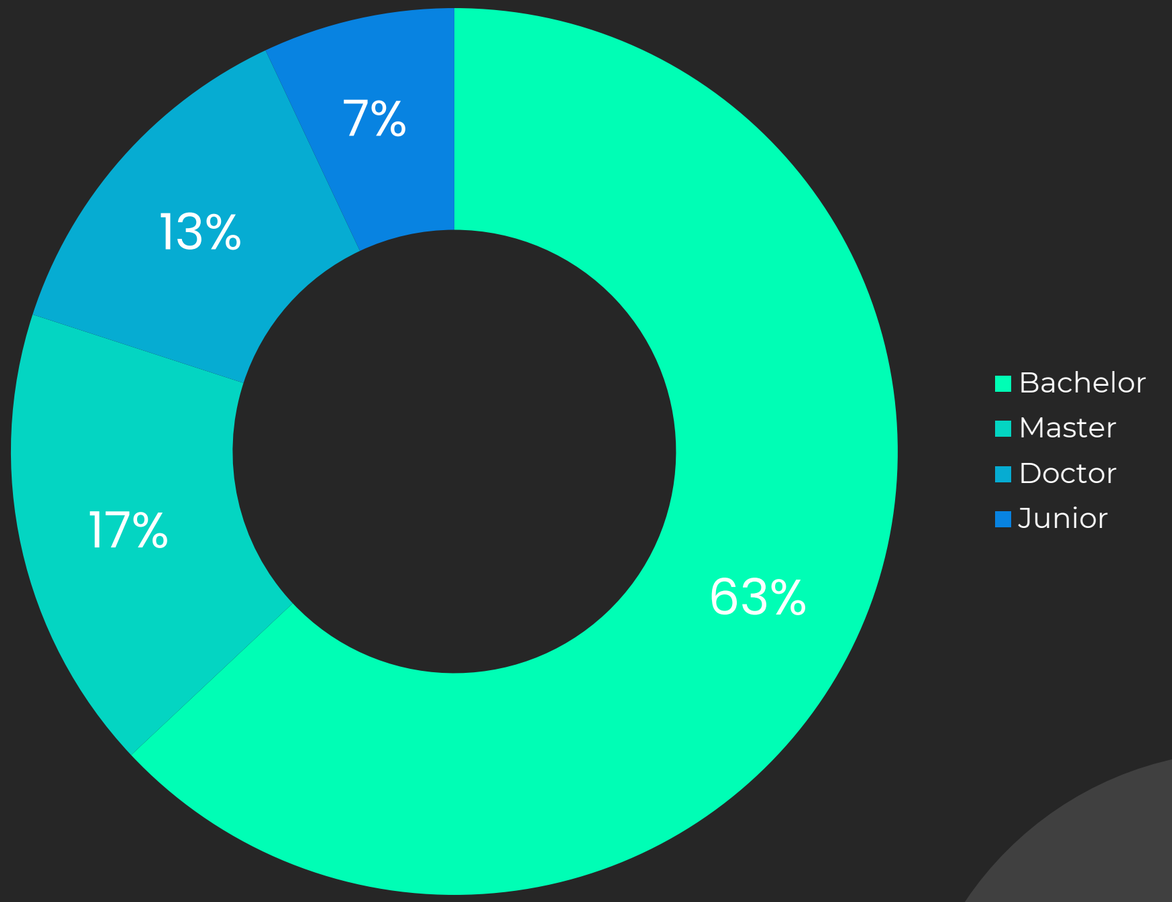
Hive is a data warehousing and SQL-like query language system built on top of Hadoop. Developed by Facebook, Hive provides a higher-level abstraction to ease the processing and analysis of large-scale datasets stored in Hadoop's distributed file system. It allows users to query data using a SQL-like language called Hive Query Language (HQL) and is particularly useful for users familiar with SQL.



Hive offers various features, including a familiar SQL interface, support for custom scripts and User Defined Functions (UDFs), optimization through query execution plans, and extensibility through custom mappers and reducers. It enables data summarization, querying, and analysis, making it accessible to users with SQL proficiency.

HIVE’S ARCHITECTURE

Hive's architecture consists of a metastore, a compiler that translates HQL into MapReduce jobs, and execution engines like Tez or Spark. The metastore manages metadata, storing schema information in a relational database. The compiler translates queries into a series of MapReduce jobs, executed on Hadoop, providing scalability and fault tolerance.



Data Types in HIVE

Primitive Types:

- int: Represents signed 32-bit integers.
- double: Represents double-precision 64-bit floating-point numbers.
- string: Represents sequences of characters, similar to text.

Complex Types:

- Arrays: Ordered collections of elements. Elements can be of any data type, including arrays themselves.
- Maps: Key-value pairs where keys and values can be of any data type.
- Structs: Complex structures with named fields, allowing the grouping of different data types.

User-Defined Types:

- Users can define custom data types tailored to their specific needs.
- Custom types provide flexibility in representing and manipulating data.
- These types can be implemented using higher-level programming languages like Java or Python.

Hive File Format

Hive supports various file formats, with the default being TextFile. Other formats like ORCFile and Parquet offer better performance and compression, contributing to efficient storage and processing of large datasets.

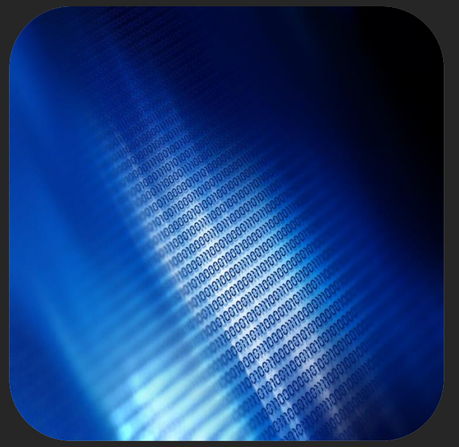
Textfile

OCRFile

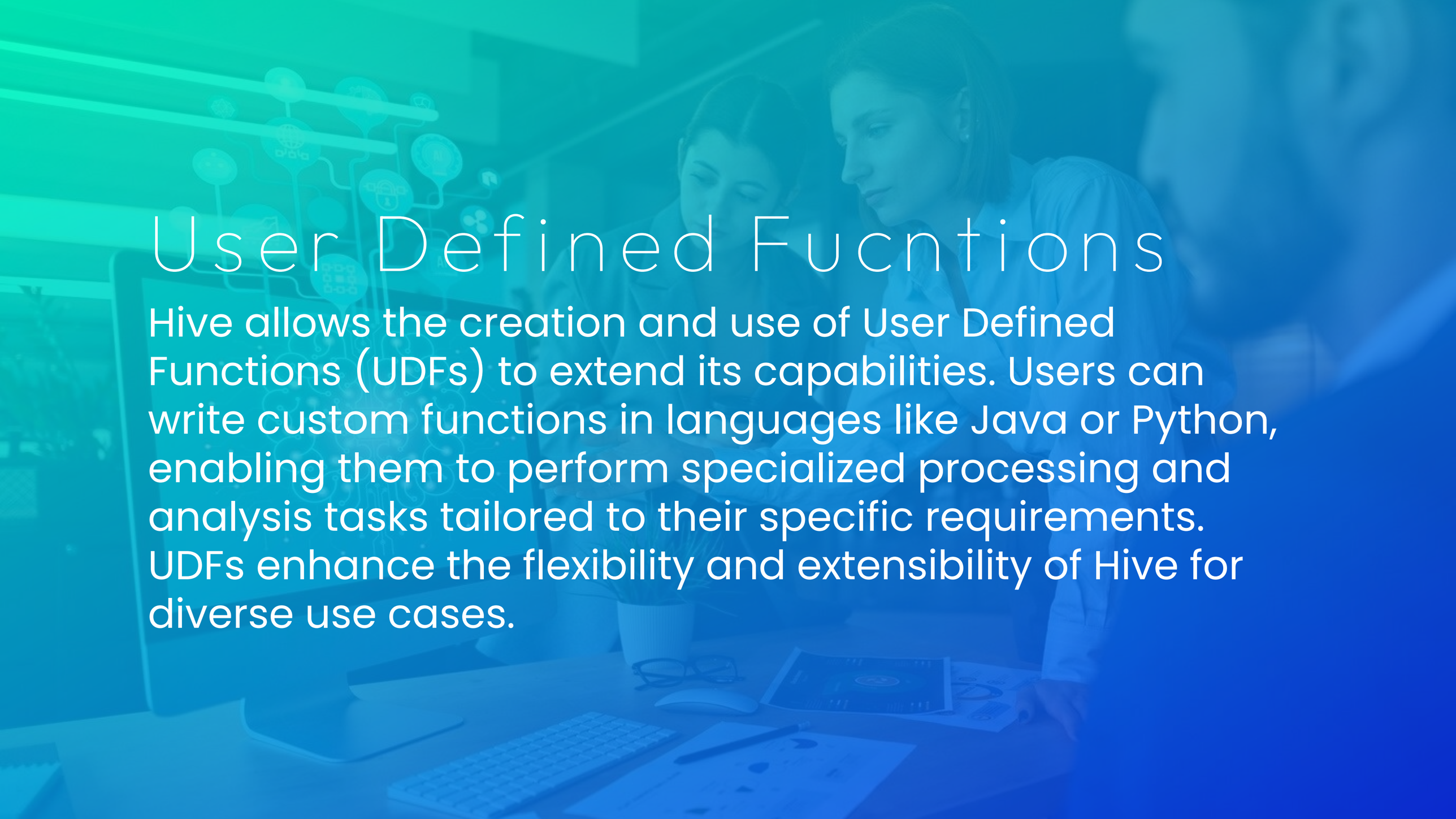
Hive Query Language

Hive Query Language (HQL) is a SQL-like language used for querying and managing Hive data. HQL allows users to express complex queries, including joins and aggregations, on large datasets. It abstracts the complexity of underlying MapReduce jobs, providing a familiar interface for users.

Slides should be simple.



Slides should be creative.



User Defined Fucntions

Hive allows the creation and use of User Defined Functions (UDFs) to extend its capabilities. Users can write custom functions in languages like Java or Python, enabling them to perform specialized processing and analysis tasks tailored to their specific requirements. UDFs enhance the flexibility and extensibility of Hive for diverse use cases.


Thanks!
Presentation
Design