Using Complex Data Types and Table Generating Functions

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Work with complex data types in Hive:

- Array, Map, Struct

Insert into and query from tables with complex data types

Flatten complex data types using table generating operations

Complex Data Types in Hive

Complex Data Types

Array

Map

Struct

Union

Complex Data Types

Array

Map

Struct

Union

Rarely used, incomplete support in Hive

Complex Data Types

Array

Map

Struct

Complex Data Types

Array

Map

Struct

Array

Collection data type

No fixed size

Entities of the same type

Only arrays of primitive types allowed

Demo

Create a column with an array data type

Insert list data into this table using

- select

- load

Complex Data Types

Array

Map

Struct

Map

Unordered collection of pairs

No fixed size

Every entity is a key, value pair

Value is accessed using a unique key

Keys and values have their own data types

Demo

Create a column with a map data type

Insert pair data into this table from a CSV file

Query map values from the Hive table

Complex Data Types

Array

Map

Struct

Struct

Logical grouping of data

Can have different data types

Can hold any number of values

Each value referenced by a name

Demo

Create a column with a struct data type

Insert data into this table from a CSV file

Query struct values from the Hive table

Built-in Functions in Hive

Built-in Functions

UDF

User-defined Functions

UDAF

User-defined Aggregate Functions

UDTF

User-defined Table-generating Functions

UDF

UDF

User-defined Functions

Works on a single row

Outputs a single row

trim(), concat(), length()

round(), floor()

UDAF

UDAF

User-defined Aggregate Functions

Works on multiple rows

Outputs a single row

count(*), sum(), avg()

UDTF

UDTF

User-defined Table-generating Functions

Works on a single row

Outputs multiple rows

explode(), posexplode()

Table-generating Functions

explode()

Flatten the data in arrays and maps

Table-generating Functions

Manager	SubordinateList
Larry	[Sundar, Eric, Jon]
Sergey	[Ruth, Urs]
Sundar	[Susan, Alan, Lazlo]

Subordinate
Sundar
Eric
John
Ruth
Urs
Susan
Alan
Lazlo

Table-generating Functions

Employee	Details
Larry	{“office”: “271B”, “numReports”: 8, “salary”: 1}
Sergey	{“office”: “271B”, “numReports”: 5, “salary”: 1}
Sundar	{“office”: “285”, “numReports”: 12}

Key	Value
office	271B
numReports	8
salary	1
office	271B
numReports	5
salary	1
office	285
numReports	12

Demo

The explode() table generating function with

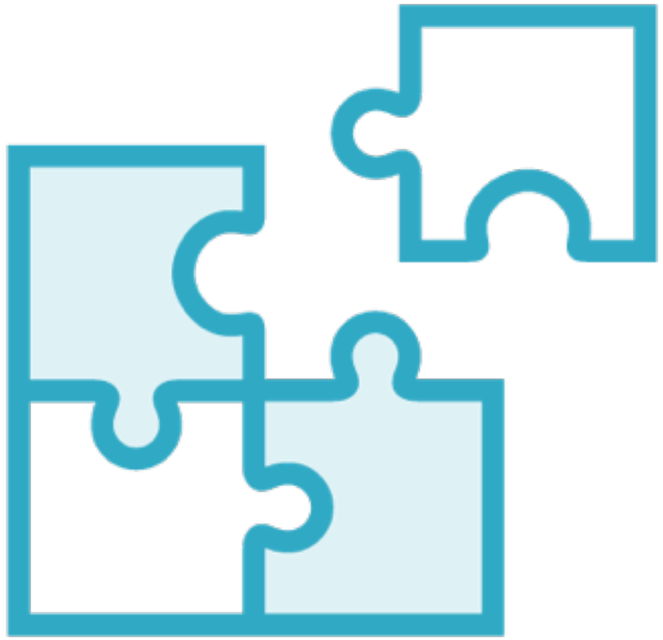
- arrays

- maps

The posexplode() table generating function

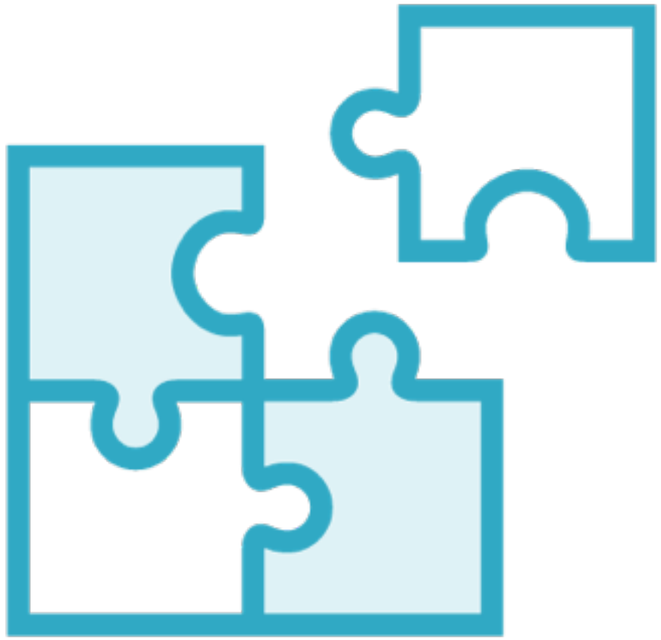
Lateral Views

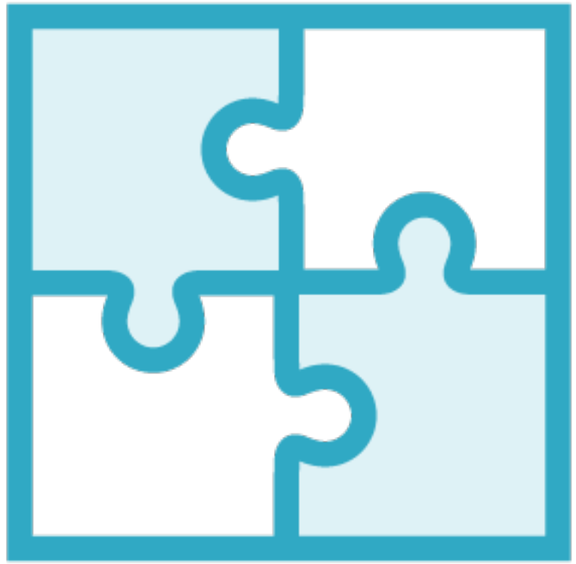
Lateral Views



A virtual table formed by the exploded view

Lateral Views





Which can be joined with the original table to allow complex queries

Lateral Views

Manager	SubordinateList
Larry	[Sundar, Eric, Jon]
Sergey	[Ruth, Urs]
Sundar	[Susan, Alan, Lazlo]

Subordinate
Sundar
Eric
John
Ruth
Urs
Susan
Alan
Lazlo

Lateral Views

This can be made a virtual table using lateral view

Subordinate
Sundar
Eric
John
Ruth
Urs
Susan
Alan
Lazlo

Lateral Views

Manager	SubordinateList	Subordinate
Larry	[Sundar, Eric, Jon]	Sundar
Larry	[Sundar, Eric, Jon]	Eric
Larry	[Sundar, Eric, Jon]	John
Sergey	[Ruth, Urs]	Ruth
Sergey	[Ruth, Urs]	Urs
Sundar	[Susan, Alan, Lazlo]	Susan
Sundar	[Susan, Alan, Lazlo]	Alan
Sundar	[Susan, Alan, Lazlo]	Lazlo

And joined with the original table

Lateral Views

Manager	SubordinateList	Subordinate
Larry	[Sundar, Eric, Jon]	Sundar
Larry	[Sundar, Eric, Jon]	Eric
Larry	[Sundar, Eric, Jon]	John
Sergey	[Ruth, Urs]	Ruth
Sergey	[Ruth, Urs]	Urs
Sundar	[Susan, Alan, Lazlo]	Susan
Sundar	[Susan, Alan, Lazlo]	Alan
Sundar	[Susan, Alan, Lazlo]	Lazlo

Query individual columns from this result

Demo

Create lateral views with exploded arrays

Join lateral views with the original table for more complex queries

Demo

Use multiple lateral views in a query

Use details from one lateral view in another lateral view

Summary

Compressed data into one column using complex data types

Created, inserted data into and queried tables with complex data types

Flattened complex data types using table generating operations

Queried flattened data using lateral views