**Complex Datatypes in Hive:**

* **Array :**
* A List of data which can be only same type

***Creating table with array data type***

***```***

CREATE TABLE students (

id int,

school\_name string,

students\_name **array<string>)**

**row format delimited by**

**fields terminated by ‘,’**

**collection item terminated by ‘,’**

**```**

­­

**Inserting data with array data type**

**```**

INSERT into students\_name VALUES

(1,’Gold Marry’, **array**(‘A’,’B’,’C’),

(2,’Jones’, **array**(‘D’,’E’,’F’)

```

**Querying data with array data type:**

Getting the first student only :

```

Select id, students\_name[0]

FROM students

```

* **Map** :

```

CREATE TABLE <employee\_skills>

(

id int,

name varchar(40),

skills map< string, string> # here in key we will mention employee name and in value skill.

)

row format delimited

fields terminated by ','

collection items terminated by ':'

map keys terminated by '|';

``

**Querying data with map data type:**

SELECT id, skills[‘employee\_name’]

* Same like Dictionary in Python

**Struct**

* + Logical grouping of data
  + Can have different data types.
  + Can hold any number of values
  + Each value reference by name

***```***

CREATE TABLE school (

id int,

school\_name string,

school\_detail **struct<Location:string, no\_of\_students:int, main\_course:string>)**

**row format delimited by**

**fields terminated by ‘,’**

**collection item terminated by ‘,’**

**map keys terminated by '|'**

**```**

**```**

SELECT id, school, **school\_detail.Location** #Accessing an item from struct data type.

FROM school

```

**Functions in Hive:**

* UDF (User defined functions)
* UDAF (User defined aggregate functions)
  + Count, sum, avg
* UDTF (user defined table generated functions)
  + Explode, posexplode
* Explode function flatter the array, map, or struct data type.

Ex.

Student\_table

|  |  |  |
| --- | --- | --- |
| id | School Name | students\_name |
| 1 | Gold Marry | [‘A’,’B’,’C’] |
| 2 | Jones | [‘D’,’E’,’F’] |

After applying explode function on Students column

**SELECT explode(students\_name) from students**

|  |
| --- |
| **Students** |
| ‘A’ |
| ‘B’ |
| ‘C’ |
| ‘D’ |
| ‘E’ |
| ‘F’ |

**Lateral View** : is like a join when you apply join explode function result to the main table other columns.

SELECT id, school\_name, students

FROM student\_table lateral view explode(students\_name) vitual\_explode\_table as students

Column name which will hold the data

Alias name of virtual table which we get from explode function