Problem Statement:

Explain Primary data types and complex data types in Hive with an example in brief.

All the data types in Hive are classified into four types, given as follows:

* Column Types
* Literals
* Null Values
* Complex Types

**1. Column Types**

Column type are used as column data types of Hive. They are as follows:

**1.1 Integral Types**

Integer type data can be specified using integral data types, INT. When the data range exceeds the range of INT, you need to use BIGINT and if the data range is smaller than the INT, you use SMALLINT. TINYINT is smaller than SMALLINT.

The following table depicts various INT data types:

|  |  |  |
| --- | --- | --- |
| **Type** | **Postfix** | **Example** |
| TINYINT | Y | 10Y |
| SMALLINT | S | 10S |
| INT | - | 10 |
| BIGINT | L | 10L |

**1.2 String Types**

String type data types can be specified using single quotes (' ') or double quotes (" "). It contains two data types: VARCHAR and CHAR. Hive follows C-types escape characters.

The following table depicts various CHAR data types:

|  |  |
| --- | --- |
| **Data Type** | **Length** |
| VARCHAR | 1 to 65355 |
| CHAR | 255 |

**1.3 Timestamp**

It supports traditional UNIX timestamp with optional nanosecond precision. It supports java.sql.Timestamp format “YYYY-MM-DD HH:MM:SS.fffffffff” and format “yyyy-mm-dd hh:mm:ss.ffffffffff”.

**1.4Dates**

DATE values are described in year/month/day format in the form {{YYYY-MM-DD}}.

**1.5 Decimals**

The DECIMAL type in Hive is as same as Big Decimal format of Java. It is used for representing immutable arbitrary precision. The syntax and example is as follows:

DECIMAL(precision, scale)

Example:

decimal(10,0)

**1.6 Union Types**

Union is a collection of heterogeneous data types. You can create an instance using **create union**.

The syntax and example is as follows:

UNIONTYPE<int, double, array<string>, struct<a:int,b:string>>

{0:1}

{1:2.0}

{2:["three","four"]}

{3:{"a":5,"b":"five"}}

{2:["six","seven"]}

{3:{"a":8,"b":"eight"}}

{0:9}

{1:10.0}

**2. Literals**

The following literals are used in Hive:

**2.1 Floating Point Types**

Floating point types are nothing but numbers with decimal points. Generally, this type of data is composed of DOUBLE data type.

**2.2 Decimal Type**

Decimal type data is nothing but floating point value with higher range than DOUBLE data type. The range of decimal type is approximately -10-308 to 10308.

**3. Null Value**

Missing values are represented by the special value NULL.

**4. Complex Types**

The Hive complex data types are as follows:

**4.1 Arrays:**

Arrays in Hive are used the same way they are used in Java.

Syntax: ARRAY<data\_type>

**Example:**

**CREATE** **TABLE** hive\_array\_table  
(name String,  
sal int,  
age array<smallint>  
)   
ROW FORMAT DELIMITED **FIELDS** TERMINATED **BY** ‘,’  
**LINES** TERMINATED **BY** ‘\n’ stored **AS** textfile;

In the above example we are using [hive](http://www.hadooptpoint.com/hadoop-hive-architecture/) array data type for storing all persons age by using that hive array in a single element. We should use only one data type in array (if data type is int all elements will int, if data type is string all elements will string).

**4.2 Maps:**

Maps in Hive are similar to Java Maps.

**Hive Map data type** is one type of Hive *complex data types example* it is an unordered collection of key-value pairs. Keys must be of primitive types. Values can be of any type.

Syntax: MAP<primitive\_type, data\_type>

**Example:**

**CREATE** **TABLE** hive\_map\_table  
(name String,  
sal int,  
age array<smallint>  
feel map<string,boolean>  
)  
ROW FORMAT DELIMITED **FIELDS** TERMINATED **BY** ‘,’  
map **KEYS** terminated **BY** ':'  
**LINES** TERMINATED **BY** ‘\n’ stored **AS** textfile;

In map function first element is string (that means primitive data types) and second element is any type (here Boolean (**Miscellaneous Data Type**)). In query we used *map keys terminated by* syntax.

**4.3 Structs:**

It is similar to STRUCT in C language. It is a collection of elements of different types. We can use any data type to specify this struct data type. Elements in STRUCT type are accessed using the DOT (.) notation.

**Example** – For a column **c** of type STRUCT {a INT; b INT} the **a** field is accessed by the expression **c.a**

**CREATE** **TABLE** hive\_struct\_table  
(name String,   
sal int,   
address struct<city:String,state:String>  
)  
ROW FORMAT DELIMITED **FIELDS** TERMINATED **BY** ‘,’  
collection items terminated **BY** ','  
**LINES** TERMINATED **BY** ‘\n’ stored **AS** textfile;

If you observed the above query in struct function first element is string and second element is also string (There is no restrictions we can use any type of data types for struct). In query we used*collection items terminated by* terminated by syntax.