**Q- Explain Primary and complex data types in HIVE with example.**

**Answer:**

**PRIMARY DATA TYPES:**

• Primary Data Types are further classified into four categories. They are:

• Numeric Types

• String Types

• Date/Time Types

• Miscellaneous Types

**Numeric Data Types**

• Integral types are – TINYINT, SMALLINT, INT & BIGINT

• Equivalent to Java’s byte, short, int, and long primitive types

• Floating types are – FLOAT, DOUBLE & DECIMAL.

• Equivalent to Java’s float and double, and SQL’s Decimal respectively.

• DECIMAL (5,2) represents total of 5 digits, out of which 2 are decimal digits.

**String Data Types**

**STRING**

• String literals can be expressed with either single quotes (') or double quotes (")

**VARCHAR**

• Varchar types are created with a length specifier (between 1 and 65355), which defines the

maximum number of characters allowed in the character string.

**CHAR**

• Char types are similar to Varchar but they are fixed-length meaning that values shorter than

the specified length value are padded with spaces but trailing spaces are not important during comparisons.

**Date/Time Types**

• Hive provides DATE and TIMESTAMP data types in traditional UNIX time stamp format for

date/time related fields in hive.

• DATE values are represented in the form YYYY-MM-DD. Example: DATE ‘2014-12-07’.

Date ranges allowed are 0000-01-01 to 9999-12-31.

• TIMESTAMP use the format yyyy-mm-dd hh:mm:ss[.f...].

**Miscellaneous Types**

• Hive supports two more primitive data types, BOOLEAN and BINARY. Similar to Java’s

Boolean, BOOLEAN in hive stores true or false values only.

• BINARY is an array of Bytes and similar to VARBINARY in many RDBMSs.

**COMPLEX DATA TYPES:**

**MAP**

• MAP<primitive\_type, data\_type>

• Collection of key-value pairs.

• Fields are accessed using array notation of keys (e.g., [‘key’]).

**STRUCT**

• STRUCT<col\_name : data\_type [COMMENT col\_comment], ...>

• It is similar to STRUCT in C language.

• It is a record type which encapsulates a set of named fields that can be any primitive

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

**UNIONTYPE**

• UNIONTYPE<data\_type, data\_type, ...>

• It is similar to Unions in C.

• At any point of time, an Union Type can hold any one (exactly one) data type from its specified data types.