**Explain the below concepts with an example in brief.**

**● Hive Data Definitions**

1. **Databases:**

It’s also common to use databases to organize production tables into logical groups.

If you don’t specify a database, the default database is used.

The simplest syntax for creating a database is shown in the following example:

CREATE DATABASE acadgild\_db;

Hive will throw an error if acadgild\_db already exists. You can suppress these warnings with this variation:

CREATE DATABASE IF NOT EXISTS acadgild\_db;

SHOW DATABASES LIKE 'a.\*';

Tables in that database will be stored in subdirectories of the database directory.

The exception is tables in the default database, which doesn’t have its own directory.

The database directory is created under a top-level directory specified by the property

set hive.metastore.warehouse.dir;

You can override this default location for the new directory as shown:

CREATE DATABASE acadgild\_db

LOCATION '/user/acadgild/mydb';

By default, Hive won’t permit you to drop a database if it contains tables.

You can either drop the tables first or append the CASCADE keyword to the command,

which will cause the Hive to drop the tables in the database first:

DROP DATABASE IF EXISTS acadgild\_db CASCADE;

1. **Tables**:

create table if not exists emp\_details

(

emp\_name string,

unit string,

exp int,

location string

)

row format delimited

fields terminated by ',';

• DROP TABLE IF EXISTS emp\_details;

Renaming a Table

• Use this statement to rename the table emp\_details to employee\_details:

ALTER TABLE emp\_details RENAME TO employee\_details;

ALTER TABLE emp\_details

CHANGE COLUMN emp\_name emp\_name STRING

COMMENT 'Employee Name'

AFTER unit;

• You have to specify the old name, a new name, and the type, even if the name or type is

not changing.

1. **Views**:

When a query becomes long or complicated, a view may be used to hide the complexity

by dividing the query into smaller, more manageable pieces;

CREATE VIEW joined\_view AS

SELECT \* FROM people JOIN cart

ON (cart.people\_id=people.id)

WHERE firstname='john';

As part of Hive’s query optimization, the clauses of both the query and view may be combined

together into a single actual query.

**● Hive Data Manipulations:**

1. **Loading files into tables:**

Load operations are currently pure copy/move operations that move datafiles into locations corresponding to Hive tables.

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION

(partcol1=val1, partcol2=val2 ...)]

filepath can be:

* a relative path, such as project/data1
* an absolute path, such as /user/hive/project/data1
* a full URI with scheme and (optionally) an authority, such as hdfs://namenode:9000/user/hive/project/data1

If the OVERWRITE keyword is used then the contents of the target table (or partition) will be deleted and replaced by the files referred to by filepat

1. **Inserting data into Hive Tables from queries:**

Query Results can be inserted into tables by using the insert clause.

Syntax:

INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]] select\_statement1 FROM from\_statement;

INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select\_statement1 FROM from\_statement;

INSERT OVERWRITE will overwrite any existing data in the table or partition

INSERT INTO will append to the table or partition, keeping the existing data intact.

1. **Update:**

UPDATE tablename SET column = value [, column = value ...] [WHERE expression]

Partitioning columns cannot be updated.

Bucketing columns cannot be updated.

In Hive 0.14, upon successful completion of this operation the changes will be auto-committed.

Vectorization will be turned off for update operations.

Updates can only be performed on tables that support ACID.

1. **Delete:**

DELETE FROM tablename [WHERE expression]

1. **Merge:**

Merge allows actions to be performed on a target table based on the results of a join with a source table.

MERGE INTO <target table> AS T USING <source expression/table> AS S

ON <boolean expression1>

WHEN MATCHED [AND <boolean expression2>] THEN UPDATE SET <set clause list>

WHEN MATCHED [AND <boolean expression3>] THEN DELETE

WHEN NOT MATCHED [AND <boolean expression4>] THEN INSERT VALUES<value list>

**● HiveQL Manipulations**

Types of Built-in Operators in HIVE are:

1. **Relational Operators**

Operators such as equals, Not equals, less than, greater than …etc

The operand types are all number types in these Operators.

1. **Arithmetic Operators**

Arithmetic operations such as addition, subtraction, multiplication and division between operands we use these Operators.

The operand types all are number types in these Operators

1. **Logical Operators**

Logical operations such as AND, OR, NOT between operands we use these Operators.

The operand types all are BOOLEAN type in these Operators

1. **Complex types**

ARRAY:

Example – array (‘siva’, ‘bala’, ‘praveen’);

An Ordered sequences of similar type elements that are indexable using zero based indexing.

Map:

MAP<primitive\_type, data\_type>

Struct:

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

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

data type.

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

Expression

UNIONTYPE:

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

specified data types.

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