Hive Data Definition:

Hive Data Definition Language (DDL) is a subset of Hive SQL statements that describe the data structure in Hive by creating, deleting, or altering schema objects such as databases, tables, views, partitions, and buckets. Most Hive DDL statements start with the keywords `CREATE`, `DROP`, or `ALTER`. The syntax of Hive DDL is very similar to the DDL in SQL.

Apache Hive supports a broad set of DDL, including (but not limited to) the following:

1.CREATE Database and DROP Database
2.CREATE Table and DROP Table
3.ALTER Table and Alter Partition statements
4.CREATE View and Drop View
5.CREATE Function and Drop Function

Data Manipulations (DML):

The Hive data manipulation language is the base for all data processing in the Hive ecosystem.

Loading Data into Tables

Processing data into information requires data to be present. The Hive environment will accept any data that can be structured in a delimited format.
Data is loaded into the platform using the following DML process.
To load data into the platform you need two components:

Data to load from (a source)
A table to load the data into (a target)

There is no transformation while loading data into tables, as Hive only performs a move/copy of the data ready for system to use.

LOADING DATA USING FILES STORED ON THE HADOOP DISTRIBUTED FILE SYSTEM

Hive supports uploading files from the Hadoop Distributed File System (HDFS). This is the most fundamental method of moving data into the Hive ecosystem.
The Hive syntax is as follows:
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename

LOADING DATA USING QUERIES

Hive supports loading data queried from existing tables into the Hive ecosystem.
The Hive syntax is as follows:

```
INSERT [OVERWRITE]
TABLE tablename1 [IF NOT EXISTS]
SELECT select_fields FROM from_statement;
```

Using an Existing Table to Create a New Table
This exercise enables you to upload a data query from a table called census.person into a table called census.personhub.
The example uses the example script Script_PersonHub.txt.
The complete script is:

```
## Use existing database
USE census;
## Create new table
CREATE TABLE personhub (
  persid        int
);
## Insert data into table, overwriting existing data in table
INSERT OVERWRITE
TABLE personhub
SELECT DISTINCT personId FROM Person;
## Check if data in table
SELECT
  persid
FROM
personhub;
```

WRITING DATA INTO THE FILE SYSTEM FROM QUERIES
Hive supports loading data queried back into the Hadoop Distributed File System.
The Hive syntax is as follows:

```
INSERT [OVERWRITE]
DIRECTORY directoryname
SELECT select_fields FROM from_statement;
```

INSERTING VALUES DIRECTLY INTO TABLES
Hive supports loading data directly into tables using a series of static values.
The Hive syntax is as follows:

```
INSERT
INTO TABLE tablename
VALUES
```

```
(row_values1),
(row_values2);
```

UPDATING DATA DIRECTLY IN TABLES
Hive supports updating data directly into tables.
The Hive syntax is as follows:

```
UPDATE tablename
SET column = value
[WHERE expression];
```

Updating Records in an Existing Table
This exercise enables you to update data directly in a table called person20.
The example uses the script Script_PersonUpdate.txt.
The complete script is:

```
USE census;
CREATE TABLE census.person20 (
  persid        int,
  lastname      string,
  firstname     string
)
CLUSTERED BY (persid) INTO 1 BUCKETS
STORED AS orc
TBLPROPERTIES('transactional' = 'true');
INSERT INTO TABLE person20 VALUES (0,'A','B'),(2,'X','Y');
```

Test if the data is updated:

```
SELECT *
FROM
  census.person20;
```

DELETING DATA DIRECTLY IN TABLES
Hive supports deleting data directly in tables.
The Hive syntax is as follows:

```
DELETE tablename
[WHERE expression];
```

Joins
USING EQUALITY JOINS TO COMBINE TABLES
Hive supports equality joins between tables to enable you to combine data from two tables.
The Hive syntax is as follows:

```
SELECT table_fields
FROM table_one
JOIN table_two
ON (table_one.key_one = table_two.key_one
AND table_one.key_two = table_two.key_two);
```

USING OUTER JOINS

Hive supports equality joins between tables using LEFT, RIGHT, and FULL OUTER joins, where keys have no match.

The Hive syntax is as follows:

```
SELECT table_fields
FROM table_one
[LEFT, RIGHT, FULL OUTER] JOIN table_two
ON (table_one.key_one = table_two.key_one
AND table_one.key_two = table_two.key_two);
```

USING LEFT SEMI-JOINS

Hive supports nested joins between tables. Consider a nested join like the following:

```
SELECT a.key, a.value
FROM a
WHERE a.key in
 (SELECT b.key
  FROM B);
```

This query will not work in Hive due to the distributed processing.

Hive can handle the query and uses a SEMI JOIN command.

The Hive syntax is as follows:

```
SELECT table_fields
FROM table_one
LEFT SEMI JOIN table_two
ON (table_one.key_one = table_two.key_one);
```

Performing a Semi-Join

Hive supports semi-joins between tables to enable you to combine data from two tables.

The example uses the script Script_SemiJoin.txt.

The complete script is:

```
USE census;
SELECT
  personname.firstname,
  personname.lastname
FROM
  census.personname
```

```
LEFT SEMI JOIN
  census.address
ON (personname.persid = address.persid);
```