### 1. Create

To create a table in HBase, use the `create` command. For example, to create a table named `employees` with two column families `personal` and `professional`:

```
create 'employees', 'personal', 'professional'
```

### 2. List

To list all the tables in HBase, use the `list` command:

```
list
```

### 3. Describe

To describe the structure of a table, use the `describe` command. For example, to describe the `employees` table:

```
describe 'employees'
```

### 4. Disable

To disable a table, use the `disable` command. For example, to disable the `employees` table:

```
disable 'employees'
```

### 5. Disable_all

To disable all tables matching a regex pattern, use the `disable_all` command. For example, to disable all tables starting with 'emp':

```
disable_all 'emp.*'
```

### 6. Enable

To enable a table, use the `enable` command. For example, to enable the `employees` table:

```
enable 'employees'
```

### 7. Enable_all

To enable all tables matching a regex pattern, use the `enable_all` command. For example, to enable all tables starting with 'emp':

```
enable_all 'emp.*'
```

### 8. Drop

To drop (delete) a table, it must first be disabled. Use the `drop` command to delete the table. For example, to drop the `employees` table:

```
disable 'employees'
drop 'employees'
```

### 9. Drop_all

To drop all tables matching a regex pattern, use the `drop_all` command. For example, to drop all tables starting with 'emp':

```
disable_all 'emp.*'
drop_all 'emp.*'
```

### 10. Show_filters

To list all available filters in HBase, use the `show_filters` command:

```
show_filters
```

### 11. Alter

To alter the structure of a table, use the `alter` command. For example, to add a new column family `contact` to the `employees` table:

```
alter 'employees', 'contact'
```

### 12. Alter_status

To check the status of an alter command, use the `alter_status` command. For example, to check the status of the `employees` table:

```
alter_status 'employees'
```

## Complex Example

#### 1. Create a Table with Versions and Compression

Let's create a table `employees` with two column families `personal` and `professional`, specifying the number of versions and compression type.

```
create 'employees', {NAME => 'personal', VERSIONS => 3, COMPRESSION => 'SNAPPY'},
{NAME => 'professional', VERSIONS => 3, COMPRESSION => 'SNAPPY'}
```

#### 2. List Tables

List all tables to verify the creation:

```
list
```

#### 3. Describe the Table

Describe the table to see its structure and properties:

```
describe 'employees'
```

#### 4. Add Data with Timestamps

Add data to the table with specific timestamps:

```
put 'employees', '101', 'personal:name', 'John', 1633024800000
put 'employees', '101', 'personal:name', 'Johnny', 1633111200000
```

```
put 'employees', '101', 'personal:lastname', 'Doe', 1633024800000
put 'employees', '101', 'professional:role', 'Engineer', 1633024800000
```

**5. Get Data with Versions**

Retrieve data with all versions:

```
get 'employees', '101', {COLUMN => 'personal:name', VERSIONS => 3}
```

**6. Disable the Table**

Disable the table to perform schema changes:

```
disable 'employees'
```

**7. Alter the Table**

Add a new column family `contact` with specific properties:

```
alter 'employees', {NAME => 'contact', VERSIONS => 2, COMPRESSION => 'GZ'}
```

**8. Enable the Table**

Enable the table after alteration:

```
enable 'employees'
```

**9. Check Alter Status**

Check the status of the alter command:

```
alter_status 'employees'
```

**10. Scan the Table with Filters**

Scan the table using a filter to retrieve specific rows:

```
scan 'employees', {FILTER => "PrefixFilter('10')"}
```

**11. Show Filters**

List all available filters to understand what can be used:

```
show_filters
```

**12. Drop the Table**

Finally, disable and drop the table:

```
disable 'employees'
drop 'employees'
```

### Summary

In this example, we:

- Created a table with specific versions and compression.

- Added data with timestamps.
- Retrieved data with versions.
- Altered the table to add a new column family.
- Used filters in a scan operation.
- Listed available filters.
- Dropped the table.

### Data manipulation commands

#### 1. Count

The `count` command counts the number of rows in a table. For example, to count the rows in the `employees` table:

```
count 'employees'
```

#### 2. Put

The `put` command inserts data into a table. For example, to add data to the `employees` table:

```
put 'employees', '101', 'personal:name', 'John'
put 'employees', '101', 'personal:lastname', 'Doe'
put 'employees', '101', 'professional:role', 'Engineer'
```

#### 3. Get

The `get` command retrieves data from a table. For example, to get specific columns from the row with key '101':

```
get 'employees', '101', {COLUMNS => ['personal:name', 'personal:lastname']}
```

#### 4. Delete

The `delete` command removes specific cells from a row. For example, to delete the 'personal:lastname' column from row '101':

```
delete 'employees', '101', 'personal:lastname'
```

#### 5. Delete All

The `deleteall` command removes all cells from a row. For example, to delete all data for row '101':

```
deleteall 'employees', '101'
```

#### 6. Truncate

The `truncate` command disables, drops, and recreates a table. This effectively clears all data but keeps the table structure. For example, to truncate the `employees` table:

```
truncate 'employees'
```

#### 7. Scan

The `scan` command retrieves multiple rows from a table. For example, to scan the `employees` table and retrieve all rows:

```
scan 'employees'
```

You can also use filters with the `scan` command to retrieve specific rows. For example, to scan rows where the row key starts with '10':

```
scan 'employees', {FILTER => "PrefixFilter('10')"}
```

### Practical Example

Let's go through a practical example using these commands:

1. **Create a table**:

   ```
   create 'employees', 'personal', 'professional'
   ```

2. **Add data to the table**:

   ```
   put 'employees', '101', 'personal:name', 'John'
   put 'employees', '101', 'personal:lastname', 'Doe'
   put 'employees', '101', 'professional:role', 'Engineer'
   put 'employees', '102', 'personal:name', 'Jane'
   put 'employees', '102', 'personal:lastname', 'Smith'
   put 'employees', '102', 'professional:role', 'Manager'
   ```

3. **Count the rows**:

   ```
   count 'employees'
   ```

4. **Retrieve specific data**:

   ```
   get 'employees', '101', {COLUMNS => ['personal:name', 'personal:lastname']}
   ```

5. **Delete specific data**:

   ```
   delete 'employees', '101', 'personal:lastname'
   ```

6. **Delete all data for a row**:

   ```
   deleteall 'employees', '102'
   ```

7. **Truncate the table**:

   ```
   truncate 'employees'
   ```

8. **Scan the table**:

   ```
   scan 'employees'
   ```