Exercise 01.04: Clustering Columns

In this exercise, you will:

- Create a 'videos_by_tag_year' table that allows range scans and ordering by year

Background

There have been some potential issues in your job as you continue to learn how the DataStax Distribution of Apache Cassandra™ and CQL work. Although you have been able to complete the assignments to the letter, the team cannot query based on 'tag' and 'year.' Fortunately, your new understanding of clustering columns will help to improve the overall design. You decide to build a table that permits querying by 'tag' and 'added_year'.

The columns are as follows:

Column Name	Data Type
tag	text
added_year	int
video_id	uuid
added_date	timestamp
title	text
user_id	uuid

Steps

1. At the prompt, navigate to the */home/ubuntu/labwork/clustering* directory in your IDE.
    a. Launch `cqlsh`.
    b. Switch to the *killrvideo* keyspace.

```
USE killrvideo;
```

In order to demonstrate some more about clustering columns, we are first going to show you *upserts*.

2. To become an *upsert* ninja, create the following (bad) table with the (crummy) primary key:

```
CREATE TABLE bad_videos_by_tag_year (
  tag TEXT,
  added_year INT,
```

```
  added_date TIMESTAMP,
  title TEXT,
  description TEXT,
  user_id UUID,
  video_id TIMEUUID,
  PRIMARY KEY ((video_id))
);
```

3. As an aside, use DESCRIBE TABLE to view the structure of your 'bad_videos_by_tag_year' table.

NOTE: Notice the column order differs from the CREATE TABLE statement. Columns are ordered by partition key, clustering columns (which will be shown later), and then alphabetical order of the remaining columns.

4. Execute the following COPY command to import *videos_by_tag_year.csv* file.

```
COPY bad_videos_by_tag_year (tag, added_year, video_id, added_date,
description, title, user_id) FROM 'videos_by_tag_year.csv' WITH
HEADER=true;
```

NOTE: We must explicitly list the column names because this table schema no longer matches the CSV structure. Also, note the number of imported rows.

5. Perform a COUNT() on the number of rows in the 'bad_videos_by_tag_year.'

Notice that the number of rows in the 'bad_videos_by_tag_year' table does not match the number of rows imported from the file *videos_by_tag_year.csv*. Since the file *videos_by_tag_year.csv* duplicates 'video_id' for each unique 'tag' and 'year' per video, several records were *upserted* during the COPY. Therefore, the 'video_id' field is not a proper partition key for this scenario.

6. Drop this nasty table like a bad habit.

```
DROP TABLE bad_videos_by_tag_year;
```

7. Confirm the only tables present are good ones.

```
DESCRIBE TABLES;
```

Your mission is as follows: 1) to restructure your table and allow users to query on 'tag' and possible 'added_year' ranges while avoiding *upserts* during the import process, and 2) return the results in descending order of year.

Steps

1. Create a table with the columns listed above to facilitate querying for videos by 'tag' within the 'added_year' range and returning the results in descending order by year.

a. We wrote most of the CREATE TABLE for you.

b. Fill in the PRIMARY KEY and CLUSTERING ORDER BY.

c. The clustering order should show the 'added_year' in descending order with any other primary key in ascending order.

NOTE: When creating the table and when doing the later COPY command, you must list the order of the importing columns *in the same order* as those columns listed in the *.csv* file.

2. Within `cqlsh`, CREATE TABLE with the proper parameters.

```
CREATE TABLE videos_by_tag_year (
    tag text,
    added_year int,
    video_id timeuuid,
    added_date timestamp,
    description text,
    title text,
    user_id uuid,
    PRIMARY KEY ((xxxx), xxxx)
) WITH CLUSTERING ORDER BY (xxxx XXX, xxxx XXXX);
```

3. Confirm the data loaded correctly:

4. Load the data from the *videos_by_tag_year.csv* file in the directory */home/ubuntu/labwork/clustering* using the COPY command.

5. Check the number of rows in the 'videos_by_tag_year' table.

NOTE: The number of rows should match the number of rows imported by the COPY command. If not, there are again *upserts*. You will need to adjust your PRIMARY KEY. If necessary, ask your instructor for help.

6. Try running queries on the 'videos_by_tag_year' table to query on a specific tag and added year.

Example queries:

tag	added_year
trailer	2015
cql	2014
spark	2014

7. Try querying for all videos with the tag 'cql' added before the year 2015. Notice you can do range queries on clustering columns.

8. Try querying for all videos added before 2015. FYI, the query will fail. What error message does `cqlsh` report? Why did the query fail whereas the previous query worked?

Exit `cqlsh`.

END OF EXERCISE